Modern Cryptography - An Introduction

Will Song

August 14, 2017

Who am I?

- Will Song
  - Competitive math nerd turned CTF player
  - Intern at AIS Denver
  - UIUC SIGPwny
  - 1064CBread

Topics For Today

- RSA
- Attacks on RSA and why you shouldn't roll your own crypto
- Diffie-Hellman
- Attacks on DH and why you shouldn't roll your own crypto
- Elliptic Curves
- Attacks on ECC and why you shouldn't roll your own crypto
- Elliptic Curve Diffie Hellman
- Attacks on ECDH and why you shouldn't roll your own crypto

Prerequisites - Useful Things to Know

- φ is the Euler Phi function. $\varphi(n)$ counts the positive integers $a \leq n$ such that $\gcd(a, n) = 1$

Prerequisites - Useful Things to Know

- φ is the Euler Phi function. $\varphi(n)$ counts the positive integers $a \leq n$ such that $\gcd(a, n) = 1$
- Notice $\varphi(p) = p - 1$ when p is prime.

Prerequisites - Useful Things to Know

- φ is the Euler Phi function. $\varphi(n)$ counts the positive integers $a \leq n$ such that $\gcd(a, n) = 1$
- Notice $\varphi(p) = p - 1$ when p is prime.
- (Euler's Totient Theorem) If $\gcd(a, n) = 1$, then $a^{\varphi(n)} \equiv 1 \pmod{n}$.

Prerequisites - Useful Things to Know

- φ is the Euler Phi function. $\varphi(n)$ counts the positive integers $a \leq n$ such that $\gcd(a, n) = 1$
- Notice $\varphi(p) = p - 1$ when p is prime.
- (Euler's Totient Theorem) If $\gcd(a, n) = 1$, then $a^{\varphi(n)} \equiv 1 \pmod{n}$.
- (Chinese Remainder Theorem) Let p, q be two positive integers such that $\gcd(p, q) = 1$. Then the system of modular equalities

$$x \equiv a \pmod{p}$$
$$x \equiv b \pmod{q}$$

has exactly one solution modulo pq. For the math savvy people, we say that there is an isomorphism between $\mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z}$ and $\mathbb{Z}/pq\mathbb{Z}$.

Prerequisites - Useful Things to Know

- φ is the Euler Phi function. $\varphi(n)$ counts the positive integers $a \leq n$ such that $\gcd(a, n) = 1$
- Notice $\varphi(p) = p - 1$ when p is prime.
- (Euler's Totient Theorem) If $\gcd(a, n) = 1$, then $a^{\varphi(n)} \equiv 1 \pmod{n}$.
- (Chinese Remainder Theorem) Let p, q be two positive integers such that $\gcd(p, q) = 1$. Then the system of modular equalities

$$x \equiv a \pmod{p}$$
$$x \equiv b \pmod{q}$$

  has exactly one solution modulo pq. For the math savvy people, we say that there is an isomorphism between $\mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z}$ and $\mathbb{Z}/pq\mathbb{Z}$.
- (Bézout) There exist integers x, y such that $ax + by = \gcd(a, b)$.

RSA

- Named after Rivest, Shamir, and Adleman.
- Probably the most widely used cryptosystem out there. See PGP/GnuPG.

RSA

- Named after Rivest, Shamir, and Adleman.
- Probably the most widely used cryptosystem out there. See PGP/GnuPG.
- Take two primes p, q, compute $N = pq$.

RSA

- Named after Rivest, Shamir, and Adleman.
- Probably the most widely used cryptosystem out there. See PGP/GnuPG.
- Take two primes p, q, compute $N = pq$.
- Pick a random number e such that $\gcd(e, \varphi(N)) = 1$.

RSA

- Named after Rivest, Shamir, and Adleman.
- Probably the most widely used cryptosystem out there. See PGP/GnuPG.
- Take two primes p, q, compute $N = pq$.
- Pick a random number e such that $\gcd(e, \varphi(N)) = 1$.
- Compute the number d such that $de \equiv 1 \pmod{\varphi(N)}$.

RSA

- Named after Rivest, Shamir, and Adleman.
- Probably the most widely used cryptosystem out there. See PGP/GnuPG.
- Take two primes p, q, compute $N = pq$.
- Pick a random number e such that $\gcd(e, \varphi(N)) = 1$.
- Compute the number d such that $de \equiv 1 \pmod{\varphi(N)}$.
- Your public key is (N, e), and your private key is d.

RSA

- Named after Rivest, Shamir, and Adleman.
- Probably the most widely used cryptosystem out there. See PGP/GnuPG.
- Take two primes p, q, compute $N = pq$.
- Pick a random number e such that $\gcd(e, \varphi(N)) = 1$.
- Compute the number d such that $de \equiv 1 \pmod{\varphi(N)}$.
- Your public key is (N, e), and your private key is d.
- To encrypt a message m, compute $c \equiv m^e \pmod{N}$.

RSA

- Named after Rivest, Shamir, and Adleman.
- Probably the most widely used cryptosystem out there. See PGP/GnuPG.
- Take two primes p, q, compute $N = pq$.
- Pick a random number e such that $\gcd(e, \varphi(N)) = 1$.
- Compute the number d such that $de \equiv 1 \pmod{\varphi(N)}$.
- Your public key is (N, e), and your private key is d.
- To encrypt a message m, compute $c \equiv m^e \pmod{N}$.
- To decrypt a message c, compute $m \equiv c^d \pmod{N}$.

RSA

- Named after Rivest, Shamir, and Adleman.
- Probably the most widely used cryptosystem out there. See PGP/GnuPG.
- Take two primes p, q, compute $N = pq$.
- Pick a random number e such that $\gcd(e, \varphi(N)) = 1$.
- Compute the number d such that $de \equiv 1 \pmod{\varphi(N)}$.
- Your public key is (N, e), and your private key is d.
- To encrypt a message m, compute $c \equiv m^e \pmod{N}$.
- To decrypt a message c, compute $m \equiv c^d \pmod{N}$.
- Ideally, it should be very hard to find d so we pick two very large primes such that N is approximately 2048 bits or higher.
- Often times people will take $e = 65537 = 2^{16} + 1$ to make encryption easier.

RSA - Trivial Example

- Pull up your local python interpreter so you can check that this actually works.

RSA - Trivial Example

- Pull up your local python interpreter so you can check that this actually works.
- Take $p = 13, q = 17, N = 221$. $\varphi(N) = 12 \cdot 16 = 192$.

RSA - Trivial Example

- Pull up your local python interpreter so you can check that this actually works.
- Take $p = 13, q = 17, N = 221$. $\varphi(N) = 12 \cdot 16 = 192$.
- Take $e = 5$, so $d = 77$ (check that this works!).

RSA - Trivial Example

- Pull up your local python interpreter so you can check that this actually works.
- Take $p = 13, q = 17, N = 221$. $\varphi(N) = 12 \cdot 16 = 192$.
- Take $e = 5$, so $d = 77$ (check that this works!).
- Let $m = 137$, so $c \equiv m^e \equiv 154 \pmod{N}$.

RSA - Trivial Example

- Pull up your local python interpreter so you can check that this actually works.
- Take $p = 13, q = 17, N = 221$. $\varphi(N) = 12 \cdot 16 = 192$.
- Take $e = 5$, so $d = 77$ (check that this works!).
- Let $m = 137$, so $c \equiv m^e \equiv 154 \pmod{N}$.
- We check that decryption works by computing $m \equiv c^d \equiv 137 \pmod{N}$.

RSA - Trivial Example

- Pull up your local python interpreter so you can check that this actually works.
- Take $p = 13, q = 17, N = 221$. $\varphi(N) = 12 \cdot 16 = 192$.
- Take $e = 5$, so $d = 77$ (check that this works!).
- Let $m = 137$, so $c \equiv m^e \equiv 154 \pmod{N}$.
- We check that decryption works by computing $m \equiv c^d \equiv 137 \pmod{N}$.
- Hooray!

RSA - On Your Own

- You need PyCrypto for this.
-
```
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP
k = RSA.generate(2048)
print "(%d, %d, %d, %d, %d)" % (k.n, k.e, k.d, k.p, k.q)
print k.encrypt(1337L, 0) # bad, use Crypto.Cipher
c = PKCS1_OAEP.new(k)
print c.encrypt("asdf")
```

RSA - Pitfalls

- People who make smart cards are very smart. They store d on the card and use the card to decrypt encrypted messages. Unfortunately, they pick d and compute e and d is usually very small for decryption efficiency.

RSA - Pitfalls

- People who make smart cards are very smart. They store d on the card and use the card to decrypt encrypted messages. Unfortunately, they pick d and compute e and d is usually very small for decryption efficiency.
  - (Wiener's Attack) $\dfrac{e}{N}$ has a continued fraction of the form $a_0 + \dfrac{1}{a_1 + \frac{1}{a_2 + \cdots}}$. We check convergents $x_n = \dfrac{k_n}{d_n}$ where $x_1 = \dfrac{1}{a_1}, x_2 = \dfrac{1}{a_1 + \frac{1}{a_2}}, \ldots$, and one of the d_n should be our desired d. This works for precisely $d < \dfrac{1}{3} N^{\frac{1}{4}}$.

RSA - Pitfalls

- People who make smart cards are very smart. They store d on the card and use the card to decrypt encrypted messages. Unfortunately, they pick d and compute e and d is usually very small for decryption efficiency.
  - (Wiener's Attack) $\frac{e}{N}$ has a continued fraction of the form $a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \cdots}}$. We check convergents $x_n = \frac{k_n}{d_n}$ where $x_1 = \frac{1}{a_1}, x_2 = \frac{1}{a_1 + \frac{1}{a_2}}, \ldots$, and one of the d_n should be our desired d. This works for precisely $d < \frac{1}{3}N^{\frac{1}{4}}$.
  - (Boneh-Durfee) By choosing a specific set of polynomials, we can perform Lenstra-Lenstra-Lovász (LLL) lattice reduction on a polynomial lattice to find a polynomial that contains d as a small root, overall taking polynomial time. This is possible due to a lemma by Hargrave and Graham. This works for precisely $d < N^{0.292}$, which is significantly better than Wiener's approach.

RSA - Pitfalls

- The broadcast issue. If an attacker obtains e different copies of your message encrypted to e many public moduli, all with the same public exponent e, you are screwed.

RSA - Pitfalls

- The broadcast issue. If an attacker obtains e different copies of your message encrypted to e many public moduli, all with the same public exponent e, you are screwed.
  - Assume $N_1 < N_2 < \cdots < N_e$. We have the following setup.

$$\begin{aligned} m^e &\equiv c_1 \pmod{N_1} \\ m^e &\equiv c_2 \pmod{N_2} \\ &\vdots \\ m^e &\equiv c_e \pmod{N_e} \end{aligned}$$

RSA - Pitfalls

- The broadcast issue. If an attacker obtains e different copies of your message encrypted to e many public moduli, all with the same public exponent e, you are screwed.
  - Assume $N_1 < N_2 < \cdots < N_e$. We have the following setup.

$$\begin{aligned} m^e &\equiv c_1 \pmod{N_1} \\ m^e &\equiv c_2 \pmod{N_2} \\ &\vdots \\ m^e &\equiv c_e \pmod{N_e} \end{aligned}$$

  If $\gcd(N_i, N_j) \neq 1$, then we can factor either N_i or N_j and easily compute d and thus m, so assume $\gcd(N_i, N_j) = 1$ for all i, j. But this is CRT!!! We can compute $m^e \pmod{\prod_i N_i}$, but $m < N_1$ and $\prod_i N_i > N_1^e > m^e$. This means solving CRT gives you precisely m^e, so we can take the e-th root and get back m.

RSA - Pitfalls

- Size matters. If $m < N^{\frac{1}{e}-\epsilon}$, then there is a polynomial time algorithm to solve $x^e - c \equiv 0 \pmod{N}$ for x and obtain m.

RSA - Pitfalls

- Size matters. If $m < N^{\frac{1}{e}-\epsilon}$, then there is a polynomial time algorithm to solve $x^e - c \equiv 0 \pmod{N}$ for x and obtain m.
  - (Coppersmith) Given a monic polynomial f of degree d, set $X = N^{\frac{1}{d}-\epsilon}$. There exists an efficient algorithm to find all roots $x < X$ such that $f(x) \equiv 0 \pmod{N}$.
  - This uses the LLL algorithm we mentioned before.

RSA - Pitfalls

- Size matters. If $m < N^{\frac{1}{e}-\epsilon}$, then there is a polynomial time algorithm to solve $x^e - c \equiv 0 \pmod{N}$ for x and obtain m.
  - (Coppersmith) Given a monic polynomial f of degree d, set $X = N^{\frac{1}{d}-\epsilon}$. There exists an efficient algorithm to find all roots $x < X$ such that $f(x) \equiv 0 \pmod{N}$.
  - This uses the LLL algorithm we mentioned before.
  - If m is super small, we can just take the e-th root of c and we win!

RSA - Pitfalls

- Size matters. If $m < N^{\frac{1}{e}-\epsilon}$, then there is a polynomial time algorithm to solve $x^e - c \equiv 0 \pmod{N}$ for x and obtain m.
  - (Coppersmith) Given a monic polynomial f of degree d, set $X = N^{\frac{1}{d}-\epsilon}$. There exists an efficient algorithm to find all roots $x < X$ such that $f(x) \equiv 0 \pmod{N}$.
  - This uses the LLL algorithm we mentioned before.
  - If m is super small, we can just take the e-th root of c and we win!
- There are many more ways of getting cheesed in RSA.

RSA - Pitfalls

- Size matters. If $m < N^{\frac{1}{e}-\epsilon}$, then there is a polynomial time algorithm to solve $x^e - c \equiv 0 \pmod{N}$ for x and obtain m.
  - (Coppersmith) Given a monic polynomial f of degree d, set $X = N^{\frac{1}{d}-\epsilon}$. There exists an efficient algorithm to find all roots $x < X$ such that $f(x) \equiv 0 \pmod{N}$.
  - This uses the LLL algorithm we mentioned before.
  - If m is super small, we can just take the e-th root of c and we win!
- There are many more ways of getting cheesed in RSA.
- Takeaways? Don't roll your own crypto. Use the peer-reviewed library for your language of choice, which will often times be NaCl/libsodium.

Diffie-Hellman - Introduction

- Everyone having a keypair is a pain in the butt. See PGP/GnuPG.

Diffie-Hellman - Introduction

- Everyone having a keypair is a pain in the butt. See PGP/GnuPG.
- If there is a way for two parties to agree on a shared secret, we can use symmetric encryption instead!

Diffie-Hellman - Introduction

- Everyone having a keypair is a pain in the butt. See PGP/GnuPG.
- If there is a way for two parties to agree on a shared secret, we can use symmetric encryption instead!
- Luckily, there is a way.

Diffie-Hellman - Introduction

- Everyone having a keypair is a pain in the butt. See PGP/GnuPG.
- If there is a way for two parties to agree on a shared secret, we can use symmetric encryption instead!
- Luckily, there is a way.
  - Alice and Bob agree on a prime p and some generator g.
  - Alice picks secret exponent a and Bob picks secret exponent b.
  - Alice sends Bob $g^a \pmod{p}$ and Bob sends Alice $g^b \pmod{p}$.
  - Both parties compute $k \equiv g^{ab} \equiv g^{ba} \pmod{p}$.
  - Easily extendable to more than two parties.

Diffie-Hellman - Introduction

- Everyone having a keypair is a pain in the butt. See PGP/GnuPG.
- If there is a way for two parties to agree on a shared secret, we can use symmetric encryption instead!
- Luckily, there is a way.
  - Alice and Bob agree on a prime p and some generator g.
  - Alice picks secret exponent a and Bob picks secret exponent b.
  - Alice sends Bob $g^a \pmod{p}$ and Bob sends Alice $g^b \pmod{p}$.
  - Both parties compute $k \equiv g^{ab} \equiv g^{ba} \pmod{p}$.
  - Easily extendable to more than two parties.
- You can trivially MITM this, but that is beyond the scope of this talk.

Diffie-Hellman - Trivial Example

- We can check the math in our local Python interpreter.

Diffie-Hellman - Trivial Example

- We can check the math in our local Python interpreter.
- Let's pick $p = 101, g = 3$.

Diffie-Hellman - Trivial Example

- We can check the math in our local Python interpreter.
- Let's pick $p = 101, g = 3$.
- Let's also pick $a = 42, b = 69$.

Diffie-Hellman - Trivial Example

- We can check the math in our local Python interpreter.
- Let's pick $p = 101, g = 3$.
- Let's also pick $a = 42, b = 69$.
- Compute $g^a \equiv 3^{42} \equiv 76 \pmod{p}$ and $g^b \equiv 3^{69} \equiv 73 \pmod{p}$.

Diffie-Hellman - Trivial Example

- We can check the math in our local Python interpreter.
- Let's pick $p = 101, g = 3$.
- Let's also pick $a = 42, b = 69$.
- Compute $g^a \equiv 3^{42} \equiv 76 \pmod{p}$ and $g^b \equiv 3^{69} \equiv 73 \pmod{p}$.
- Compute $g^{ab} \equiv 76^{69} \equiv 45 \pmod{p}$ and $g^{ba} \equiv 73^{42} \equiv 45 \pmod{p}$.

Diffie-Hellman - Pitfalls

- Sometimes $p-1$ is composed entirely of small prime factors (we say $p-1$ is smooth).

Diffie-Hellman - Pitfalls

- Sometimes $p-1$ is composed entirely of small prime factors (we say $p-1$ is smooth).
  - (Pohlig-Hellman) Write $p-1 = p_1^{e_1} \cdots p_k^{e_k}$. Given that the p_i are small and two integers g, h modulo p, there exists an efficient algorithm to compute an x such that $g^x \equiv h \pmod{p}$. Indeed, we can write $g_i = g^{n/p_i^{e_i}}, h_i = h^{n/p_i^{e_i}}$ and there exists an efficient algorithm to compute an x such that $g_i^{x_i} = h_i \pmod{p_i^{e_i}}$, and we can solve $x \equiv x_i \pmod{p_i^{e_i}}$ with CRT.

Diffie-Hellman - Pitfalls

- Sometimes $p-1$ is composed entirely of small prime factors (we say $p-1$ is smooth).
  - (Pohlig-Hellman) Write $p-1 = p_1^{e_1} \cdots p_k^{e_k}$. Given that the p_i are small and two integers g, h modulo p, there exists an efficient algorithm to compute an x such that $g^x \equiv h \pmod{p}$. Indeed, we can write $g_i = g^{n/p_i^{e_i}}, h_i = h^{n/p_i^{e_i}}$ and there exists an efficient algorithm to compute an x such that $g_i^{x_i} = h_i \pmod{p_i^{e_i}}$, and we can solve $x \equiv x_i \pmod{p_i^{e_i}}$ with CRT.
  - We avoid this by choosing $p = 2q+1$, where p, q are both prime and q is large.

Diffie-Hellman - Pitfalls

- Sometimes $p-1$ is composed entirely of small prime factors (we say $p-1$ is smooth).
  - (Pohlig-Hellman) Write $p-1 = p_1^{e_1} \cdots p_k^{e_k}$. Given that the p_i are small and two integers g, h modulo p, there exists an efficient algorithm to compute an x such that $g^x \equiv h \pmod{p}$. Indeed, we can write $g_i = g^{n/p_i^{e_i}}, h_i = h^{n/p_i^{e_i}}$ and there exists an efficient algorithm to compute an x such that $g_i^{x_i} = h_i \pmod{p_i^{e_i}}$, and we can solve $x \equiv x_i \pmod{p_i^{e_i}}$ with CRT.
  - We avoid this by choosing $p = 2q+1$, where p, q are both prime and q is large.
- If you're lazy, sometimes it's just not big enough.
  - In general, Diffie-Hellman on $2n$-bit p offers n bits of security.

Diffie-Hellman - Pitfalls

- Sometimes $p-1$ is composed entirely of small prime factors (we say $p-1$ is smooth).
  - (Pohlig-Hellman) Write $p-1 = p_1^{e_1} \cdots p_k^{e_k}$. Given that the p_i are small and two integers g, h modulo p, there exists an efficient algorithm to compute an x such that $g^x \equiv h \pmod{p}$. Indeed, we can write $g_i = g^{n/p_i^{e_i}}, h_i = h^{n/p_i^{e_i}}$ and there exists an efficient algorithm to compute an x such that $g_i^{x_i} = h_i \pmod{p_i^{e_i}}$, and we can solve $x \equiv x_i \pmod{p_i^{e_i}}$ with CRT.
  - We avoid this by choosing $p = 2q+1$, where p, q are both prime and q is large.
- If you're lazy, sometimes it's just not big enough.
  - In general, Diffie-Hellman on $2n$-bit p offers n bits of security.
  - (Baby Step Giant Step) We store (j, g^j) for $0 \le j \le \sqrt{p}$. We can check if some function f satisfies $f^k(h) = g^j$ for some $0 \le k \le \sqrt{p}$ and j in the table and if found, we can produce the correct x.

Diffie-Hellman - Pitfalls

- Sometimes $p-1$ is composed entirely of small prime factors (we say $p-1$ is smooth).
  - (Pohlig-Hellman) Write $p-1 = p_1^{e_1} \cdots p_k^{e_k}$. Given that the p_i are small and two integers g, h modulo p, there exists an efficient algorithm to compute an x such that $g^x \equiv h \pmod{p}$. Indeed, we can write $g_i = g^{n/p_i^{e_i}}, h_i = h^{n/p_i^{e_i}}$ and there exists an efficient algorithm to compute an x such that $g_i^{x_i} = h_i \pmod{p_i^{e_i}}$, and we can solve $x \equiv x_i \pmod{p_i^{e_i}}$ with CRT.
  - We avoid this by choosing $p = 2q + 1$, where p, q are both prime and q is large.
- If you're lazy, sometimes it's just not big enough.
  - In general, Diffie-Hellman on $2n$-bit p offers n bits of security.
  - (Baby Step Giant Step) We store (j, g^j) for $0 \le j \le \sqrt{p}$. We can check if some function f satisfies $f^k(h) = g^j$ for some $0 \le k \le \sqrt{p}$ and j in the table and if found, we can produce the correct x.
  - Not usually a problem unless you're too lazy to make/use a multiprecision integer library.

Diffie-Hellman - Pitfalls

- Sometimes people aren't nice.
  - Occasionally the attacker will choose a g such that $g^x \pmod{p}$ does not have many values, and can brute force the shared secret.

Diffie-Hellman - Pitfalls

- Sometimes people aren't nice.
  - Occasionally the attacker will choose a g such that $g^x \pmod p$ does not have many values, and can brute force the shared secret.
- And other ways of attacking the Discrete Log Problem if you aren't careful. Validate your inputs and use big enough numbers and you shouldn't have any problems.

Prerequisites - Algebra

- A group is a set G equipped with an closed associative binary operation $*$ such that
  - There exists $e \in G$ such that $e * g = g * e = g$ for all $g \in G$.
  - For each $g \in G$, there exists $g^{-1} \in G$ such that $g * g^{-1} = g^{-1} * g = e$.
- We used groups in Diffie-Hellman. Can anyone spot what group was used?

Prerequisites - Algebra

- A group is a set G equipped with an closed associative binary operation $*$ such that
  - There exists $e \in G$ such that $e * g = g * e = g$ for all $g \in G$.
  - For each $g \in G$, there exists $g^{-1} \in G$ such that $g * g^{-1} = g^{-1} * g = e$.
- We used groups in Diffie-Hellman. Can anyone spot what group was used?
  - The $p - 1$ non-zero remainders modulo p under multiplication form a group because $\gcd(a, p) = 1$ implies the existence of integers x, y such that $ax + py = 1$, or $ax \equiv 1 \pmod{p}$.

Prerequisites - Algebra

- A group is a set G equipped with an closed associative binary operation $*$ such that
  - There exists $e \in G$ such that $e * g = g * e = g$ for all $g \in G$.
  - For each $g \in G$, there exists $g^{-1} \in G$ such that $g * g^{-1} = g^{-1} * g = e$.
- We used groups in Diffie-Hellman. Can anyone spot what group was used?
  - The $p - 1$ non-zero remainders modulo p under multiplication form a group because $\gcd(a, p) = 1$ implies the existence of integers x, y such that $ax + py = 1$, or $ax \equiv 1 \pmod{p}$.
- We don't require the group operation to commute, but when it does we say the group is abelian, or commutative.

Prerequisites - Algebra

- A ring R is a set equipped with two closed associative binary operations, $+$ and $\cdot$, such that:
  - There exists $0 \in R$ such that $r + 0 = 0 + r = r$ for all $r \in R$.
  - For each $r \in R$, there exists $-r \in R$ such that $r + (-r) = 0$.
  - There exists $1 \in R$ such that $r \cdot 1 = 1 \cdot r = r$ for all $r \in R$.
  - $r_1(r_2 + r_3) = r_1r_2 + r_1r_3$ for all $r_1, r_2, r_3 \in R$.
- A field F is a ring but every non-zero element $f \in F$ has an inverse f^{-1} such that $ff^{-1} = f^{-1}f = 1$.
- Can anyone give any examples of rings and or fields?

Prerequisites - Algebra

- A ring R is a set equipped with two closed associative binary operations, $+$ and $\cdot$, such that:
  - There exists $0 \in R$ such that $r + 0 = 0 + r = r$ for all $r \in R$.
  - For each $r \in R$, there exists $-r \in R$ such that $r + (-r) = 0$.
  - There exists $1 \in R$ such that $r \cdot 1 = 1 \cdot r = r$ for all $r \in R$.
  - $r_1(r_2 + r_3) = r_1r_2 + r_1r_3$ for all $r_1, r_2, r_3 \in R$.
- A field F is a ring but every non-zero element $f \in F$ has an inverse f^{-1} such that $ff^{-1} = f^{-1}f = 1$.
- Can anyone give any examples of rings and or fields?
- We will focus on $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$, or the field/ring of numbers modulo p where p is prime.

Elliptic Curves - Definition

- An elliptic curve over the field F, $\mathcal{E}(F)$, is defined as the set of points in $(x, y) \in F^2$ that satisfy
$$y^2 = x^3 + ax + b$$
for some $a, b \in F$.
In particular, we also want this thing to be non-singular, or some value called the discriminant (very similar to the discriminant of a quadratic equation) to be non-zero.

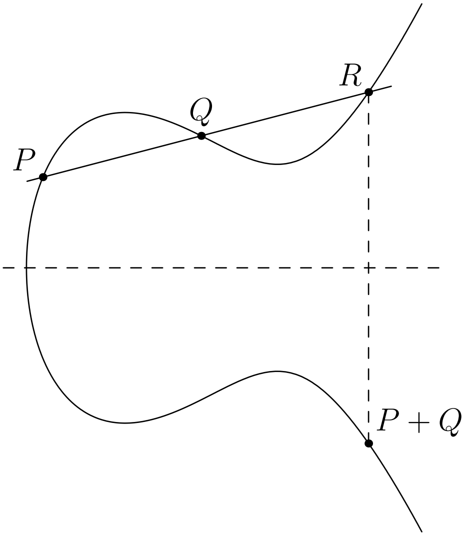
Elliptic Curves - Definition

- An elliptic curve over the field F, $\mathcal{E}(F)$, is defined as the set of points in $(x, y) \in F^2$ that satisfy
$$y^2 = x^3 + ax + b$$
for some $a, b \in F$.
In particular, we also want this thing to be non-singular, or some value called the discriminant (very similar to the discriminant of a quadratic equation) to be non-zero.
- For crypto purposes, we will use $F = \mathbb{F}_q$ where $q = p^k$ for some prime p, but often times we will see $q = p$. In fact, I haven't introduced what $\mathbb{F}_q$ for $k > 1$ even looks like so we won't be seeing it yet.

Elliptic Curves - Definition

- An elliptic curve over the field F, $\mathcal{E}(F)$, is defined as the set of points in $(x, y) \in F^2$ that satisfy
$$y^2 = x^3 + ax + b$$
for some $a, b \in F$.
In particular, we also want this thing to be non-singular, or some value called the discriminant (very similar to the discriminant of a quadratic equation) to be non-zero.
- For crypto purposes, we will use $F = \mathbb{F}_q$ where $q = p^k$ for some prime p, but often times we will see $q = p$. In fact, I haven't introduced what $\mathbb{F}_q$ for $k > 1$ even looks like so we won't be seeing it yet.
- We want to turn this thing into a group. To do that, we need a picture.
- If you want to sound smart, you can also call this type of object an abelian variety (it is a specific type of abelian variety), or a non-singular projective cubic.

Elliptic Curves in $\mathbb{R}$ - The Picture



Special Cases

- What about $P + P = 2P$?

Special Cases

- What about $P + P = 2P$?
  - Taking the tangent line to P seems intuitive enough.

Special Cases

- What about $P + P = 2P$?
  - Taking the tangent line to P seems intuitive enough.
- With respect to the previous image, $P + Q + R$ doesn't seem to exist!
  - We switch into $\mathbb{RP}^2$.
  - Define $0 = \infty$.

Elliptic Curves in $\mathbb{R}$ - The Formulas

- The line definition works every time because algebra.

Elliptic Curves in $\mathbb{R}$ - The Formulas

- The line definition works every time because algebra.
  - Let $P = (P_x, P_y), Q = (Q_x, Q_y)$.
  - We can compute $\lambda = \frac{P_y - Q_y}{P_x - Q_x}$ if $P \neq Q$ and $\lambda = \frac{3x^2 + a}{2y}$ otherwise, as well as

$$
\begin{aligned}
(P+Q)_x &= \lambda^2 - (P_x + Q_x) \\
-(P+Q)_y &= \lambda((P+Q)_x - P_x) + P_y.
\end{aligned}
$$

- Computing multiples $n \cdot P$ is easy with a method similar to exponentiation by squaring.

Elliptic Curves - The Formulas

- But how do we do this in $\mathbb{F}_p$?

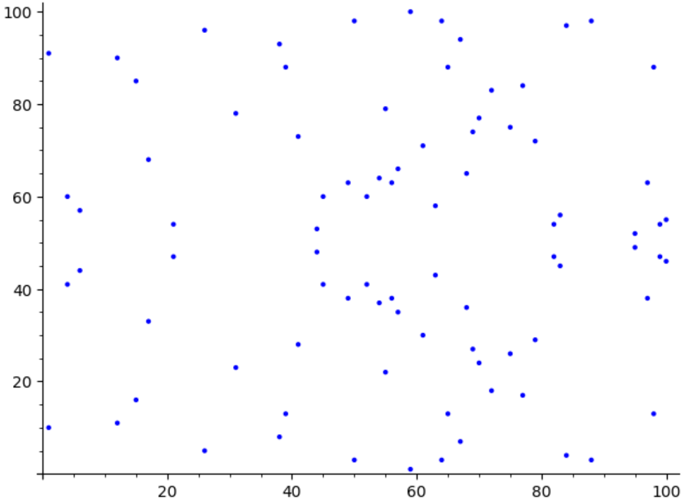
Elliptic Curves - The Formulas

- But how do we do this in $\mathbb{F}_p$?
- Notice how every non-zero element of $\mathbb{F}_p$ has an inverse? Yeah dividing is multiplying by the inverse (yay high school).
- You can't get a very good picture of this in $\mathbb{F}_p$ which is why we tend to draw it in $\mathbb{R}$ and then just copy over the intuition.
- The math of elliptic curves is hard and you surely don't want to do it yourself. Use a library for it.

Elliptic Curves - On Your Own

- You will need either SageMath or some fraction of packages that come with Sage.
- ```
  E = EllipticCurve(GF(101), [1, -3]) # EllipticCurve(F, [A, B])
  print E.is_ordinary()
  print E.gens()[0].xy()
  E.plot()
  ```
  ```

Elliptic Curves - On Your Own



Elliptic Curves - Pitfalls

- Remember the DUAL_EC_DRBG scandal? Here's how the backdoor worked.
  - Start with your curve $\mathcal{E}(\mathbb{F}_p)$ and two public points P, Q. Let $\pi_x : \mathcal{E}(\mathbb{F}_p) \to \mathbb{F}_p$ be the projection onto the x-axis. Let $T : \mathbb{F}_p \to \mathbb{F}_p$ truncate an integer to 240 bits (we are working with a 256 bit prime). Let $f_1(s) = sP$, $f_2(s) = sQ$. Let s_0 be the initial seed. We will produce seeds $s_1, s_2, \ldots$ and random numbers $r_1, r_2, \ldots$ like so.

$$
\begin{tikzcd}
s_0 \arrow[r, "\pi_x \circ f_1"] & s_1 \arrow[r, "\pi_x \circ f_1"] \arrow[d, "T \circ \pi_x \circ f_2"] & s_2 \arrow[r, "\pi_x \circ f_1"] \arrow[d, "T \circ \pi_x \circ f_2"] & \cdots \arrow[d, "T \circ \pi_x \circ f_2"] \\
 & r_1 & r_2 & \cdots
\end{tikzcd}
$$

Elliptic Curves - Pitfalls

- Remember the DUAL_EC_DRBG scandal? Here's how the backdoor worked.
    - What if you know a l such that $P = lQ$?
    - Suppose you retrieve $R = f_2(s_1) = s_1 Q$. What can you do?

Elliptic Curves - Pitfalls

- Remember the DUAL_EC_DRBG scandal? Here's how the backdoor worked.
  - What if you know a l such that $P = lQ$?
  - Suppose you retrieve $R = f_2(s_1) = s_1 Q$. What can you do?
  - We can now compute $\pi_x(lR) = \pi_x((ls_1)Q) = \pi_x((s_1 l)Q) = \pi_x(s_1 P) = s_2$ and the PRNG is broken.

Elliptic Curves - Pitfalls

- Remember the DUAL_EC_DRBG scandal? Here's how the backdoor worked.
  - What if you know a l such that $P = lQ$?
  - Suppose you retrieve $R = f_2(s_1) = s_1 Q$. What can you do?
  - We can now compute $\pi_x(lR) = \pi_x((ls_1)Q) = \pi_x((s_1 l)Q) = \pi_x(s_1 P) = s_2$ and the PRNG is broken.
  - We can do this in 2^{16} brute force attempts (notice how the sign of y makes no difference to the resulting x).

Elliptic Curves - Pitfalls

- Remember the DUAL_EC_DRBG scandal? Here's how the backdoor worked.
  - What if you know a l such that $P = lQ$?
  - Suppose you retrieve $R = f_2(s_1) = s_1Q$. What can you do?
  - We can now compute $\pi_x(lR) = \pi_x((ls_1)Q) = \pi_x((s_1l)Q) = \pi_x(s_1P) = s_2$ and the PRNG is broken.
  - We can do this in 2^{16} brute force attempts (notice how the sign of y makes no difference to the resulting x).
- Lesson? Don't trust the NSA to not spy on you.

Elliptic Curves - Extra

- Elliptic Curves offer similar security levels to RSA at significantly smaller key sizes and parameters.
- Around a year ago there was a seemingly inane math question posted to several Facebook groups that boiled down to the following.
  Find integers x, y, z satisfying

$$\frac{x}{y+z} + \frac{y}{z+x} + \frac{z}{x+y} = 4.$$

  I haven't taught you why this is equivalent to an elliptic curve but if you're interested you can play around with it and try to find the connection. (Hint: the smallest x, y, z are roughly 80 digits long)

Elliptic Curve Diffie-Hellman

- Earlier we noted that we used a group for regular Diffie-Hellman. We remark that we can do the same with elliptic curves, because they form a group!
  - Alice and Bob agree on a field $\mathbb{F}_p$, a curve $\mathcal{E}(\mathbb{F}_p)$, and a point P.
  - Alice picks a random number a and Bob picks a random number b.
  - Alice sends Bob aP and Bob sends Alice bP.
  - Both parties compute $K = (ab)P = (ba)P$.

Elliptic Curve Diffie-Hellman - Pitfalls

- A lot of the pitfalls of regular DH carry over. If the number of points on the curve is too small, BSGS can solve it relatively quickly. If the number of points is smooth, we can use Pohlig-Hellman.

Elliptic Curve Diffie-Hellman - Pitfalls

- A lot of the pitfalls of regular DH carry over. If the number of points on the curve is too small, BSGS can solve it relatively quickly. If the number of points is smooth, we can use Pohlig-Hellman.
- We also have some new issues.
- If the curve in question has a prime number of points, there is an efficient algorithm to solve the ECDLP.

Elliptic Curve Diffie-Hellman - Pitfalls

- A lot of the pitfalls of regular DH carry over. If the number of points on the curve is too small, BSGS can solve it relatively quickly. If the number of points is smooth, we can use Pohlig-Hellman.
- We also have some new issues.
- If the curve in question has a prime number of points, there is an efficient algorithm to solve the ECDLP.
  - (Smart) We lift $\mathcal{E}(\mathbb{F}_p)$ to $\mathcal{E}(\mathbb{Q}_p)$ over the p-adic numbers via the natural embedding and then perform a Hensel Lift via Hensel's Lemma to lift the two curve points in question to $\mathcal{E}(\mathbb{Q}_p)$. Some math in $\mathbb{Q}_p$ gives us the desired exponent.

Elliptic Curve Diffie-Hellman - Pitfalls

- When you generate a random curve, you really have no idea how many points it will have.

Elliptic Curve Diffie-Hellman - Pitfalls

- When you generate a random curve, you really have no idea how many points it will have.
  - More specifically, we have by a theorem of Hasse, the bound

$$|\#\mathcal{E}(\mathbb{F}_p) - (p+1)| \leq 2\sqrt{p}.$$

  - To count the number of points, we use an algorithm given by Schoof and later improved on by Elkies and Atkins. This requires yet even more math to understand, so use a library for this.

Elliptic Curve Diffie-Hellman - Pitfalls

- When you generate a random curve, you really have no idea how many points it will have.
  - More specifically, we have by a theorem of Hasse, the bound

$$|\#\mathcal{E}(\mathbb{F}_p) - (p+1)| \leq 2\sqrt{p}.$$

  - To count the number of points, we use an algorithm given by Schoof and later improved on by Elkies and Atkins. This requires yet even more math to understand, so use a library for this.
  - Let $t = p + 1 - \#\mathcal{E}(\mathbb{F}_p)$. We call this the trace of Frobenius of the curve $\mathcal{E}(\mathbb{F}_p)$. Given a desired trace of Frobenius t, we are able to produce curves with $p + 1 \pm t$ points via the theory of Complex Multiplication and class field theory, which is also more math, so use a library.

Conclusion

- tl;dr cryptography is hard. Please don't try to write your own library. You will screw up some way or another.

Resources

- picoCTF
- plaidCTF
- uiuctf
- CSAW CTF
- My Github (ctf solutions/source)
- Boneh-Durfee and Coppersmith
- SageMath
- Rational Points on Elliptic Curves (Silverman & Tate)
- Abstract Algebra (Dummit & Foote)

Q & A

- Any questions? Any enthusiasm for wanting to get murdered learning the math behind my Underhanded Crypto Challenge submission?