Public Key Crtyptography

Public Key Cryptography

Alice and Bob never have to meet!

Number Theory Algs needed for Public Key

The following can be done quickly.

Number Theory Algs needed for Public Key

The following can be done quickly.

1. Given (a, n, p) compute $a^n \pmod{p}$. Repeated Squaring. Takes $\lg n +$ (number of 1's in n).

Number Theory Algs needed for Public Key

The following can be done quickly.

1. Given (a, n, p) compute $a^n \pmod{p}$. Repeated Squaring. Takes $\lg n +$ (number of 1's in n).
2. Given n, find a safe prime of length n and a generator g.

Number Theory Algs needed for Public Key

The following can be done quickly.

1. Given (a, n, p) compute $a^n \pmod{p}$. Repeated Squaring. Takes $\lg n +$ (number of 1's in n).
2. Given n, find a safe prime of length n and a generator g.
3. Given a, b rel prime find inverse of a mod b: Euclidean alg.

Number Theory Assumptions

1. Discrete Log is hard.
2. Factoring is hard.

Note Actual hardness assumptions are not quite DL hard and Factoring hard but are close.

The Diffie-Helman Key Exchange

Alice and Bob will share a secret s. Security parameter L.

The Diffie-Helman Key Exchange

Alice and Bob will share a secret s. Security parameter L.

1. Alice finds a (p, g), p of length L, g gen for $\mathbb{Z}_p$.

The Diffie-Helman Key Exchange

Alice and Bob will share a secret s. Security parameter L.

1. Alice finds a (p, g), p of length L, g gen for $\mathbb{Z}_p$.
2. Alice sends (p, g) to Bob in the clear (Eve sees (p, g)).

The Diffie-Helman Key Exchange

Alice and Bob will share a secret s. Security parameter L.

1. Alice finds a (p, g), p of length L, g gen for $\mathbb{Z}_p$.
2. Alice sends (p, g) to Bob in the clear (Eve sees (p, g)).
3. Alice picks random $a \in \{1, \ldots, p-1\}$, computes g^a and sends it to Bob in the clear (Eve sees g^a).

The Diffie-Helman Key Exchange

Alice and Bob will share a secret s. Security parameter L.

1. Alice finds a (p, g), p of length L, g gen for $\mathbb{Z}_p$.
2. Alice sends (p, g) to Bob in the clear (Eve sees (p, g)).
3. Alice picks random $a \in \{1, \ldots, p-1\}$, computes g^a and sends it to Bob in the clear (Eve sees g^a).
4. Bob picks random $b \in \{1, \ldots, p-1\}$, computes g^b and sends it to Alice in the clear (Eve sees g^b).

The Diffie-Helman Key Exchange

Alice and Bob will share a secret s. Security parameter L.

1. Alice finds a (p, g), p of length L, g gen for $\mathbb{Z}_p$.
2. Alice sends (p, g) to Bob in the clear (Eve sees (p, g)).
3. Alice picks random $a \in \{1, \ldots, p-1\}$, computes g^a and sends it to Bob in the clear (Eve sees g^a).
4. Bob picks random $b \in \{1, \ldots, p-1\}$, computes g^b and sends it to Alice in the clear (Eve sees g^b).
5. Alice computes $(g^b)^a = g^{ab}$.

The Diffie-Helman Key Exchange

Alice and Bob will share a secret s. Security parameter L.

1. Alice finds a (p, g), p of length L, g gen for $\mathbb{Z}_p$.
2. Alice sends (p, g) to Bob in the clear (Eve sees (p, g)).
3. Alice picks random $a \in \{1, \ldots, p-1\}$, computes g^a and sends it to Bob in the clear (Eve sees g^a).
4. Bob picks random $b \in \{1, \ldots, p-1\}$, computes g^b and sends it to Alice in the clear (Eve sees g^b).
5. Alice computes $(g^b)^a = g^{ab}$.
6. Bob computes $(g^a)^b = g^{ab}$.

The Diffie-Helman Key Exchange

Alice and Bob will share a secret s. Security parameter L.

1. Alice finds a (p, g), p of length L, g gen for $\mathbb{Z}_p$.
2. Alice sends (p, g) to Bob in the clear (Eve sees (p, g)).
3. Alice picks random $a \in \{1, \ldots, p-1\}$, computes g^a and sends it to Bob in the clear (Eve sees g^a).
4. Bob picks random $b \in \{1, \ldots, p-1\}$, computes g^b and sends it to Alice in the clear (Eve sees g^b).
5. Alice computes $(g^b)^a = g^{ab}$.
6. Bob computes $(g^a)^b = g^{ab}$.
7. g^{ab} is the shared secret.

The Diffie-Helman Key Exchange

Alice and Bob will share a secret s. Security parameter L.

1. Alice finds a (p, g), p of length L, g gen for $\mathbb{Z}_p$.
2. Alice sends (p, g) to Bob in the clear (Eve sees (p, g)).
3. Alice picks random $a \in \{1, \ldots, p-1\}$, computes g^a and sends it to Bob in the clear (Eve sees g^a).
4. Bob picks random $b \in \{1, \ldots, p-1\}$, computes g^b and sends it to Alice in the clear (Eve sees g^b).
5. Alice computes $(g^b)^a = g^{ab}$.
6. Bob computes $(g^a)^b = g^{ab}$.
7. g^{ab} is the shared secret.

Def Let f be $f(p, g, g^a, g^b) = g^{ab}$.

Hardness assumption: f is hard to compute.

ElGamal Uses DH So Can Control Message

1. Alice and Bob do Diffie Helman.
2. Alice and Bob share secret $s = g^{ab}$.
3. Alice and Bob compute $(g^{ab})^{-1} \pmod{p}$.
4. To send m, Alice sends $c = mg^{ab}$
5. To decrypt, Bob computes $c(g^{ab})^{-1} \equiv mg^{ab}(g^{ab})^{-1} \equiv m$

We omit discussion of Hardness assumption (HW)

RSA

Let L be a security parameter

RSA

Let L be a security parameter

1. **Alice** picks two primes p, q of length L. $N = pq$.

RSA

Let L be a security parameter

1. **Alice** picks two primes p, q of length L. $N = pq$.
2. **Alice** computes $R = \phi(N) = \phi(pq) = (p-1)(q-1)$.

RSA

Let L be a security parameter

1. **Alice** picks two primes p, q of length L. $N = pq$.
2. **Alice** computes $R = \phi(N) = \phi(pq) = (p-1)(q-1)$.
3. **Alice** picks an $e \in \{\frac{R}{3}, \ldots, \frac{2R}{3}\}$ that is relatively prime to R.
   Alice finds d such that $ed \equiv 1 \pmod{R}$.

RSA

Let L be a security parameter

1. **Alice** picks two primes p, q of length L. $N = pq$.
2. **Alice** computes $R = \phi(N) = \phi(pq) = (p-1)(q-1)$.
3. **Alice** picks an $e \in \{\frac{R}{3}, \ldots, \frac{2R}{3}\}$ that is relatively prime to R.
   Alice finds d such that $ed \equiv 1 \pmod{R}$.
4. **Alice** broadcasts (N, e). (Bob and Eve both see it.)

RSA

Let L be a security parameter

1. **Alice** picks two primes p, q of length L. $N = pq$.
2. **Alice** computes $R = \phi(N) = \phi(pq) = (p-1)(q-1)$.
3. **Alice** picks an $e \in \{\frac{R}{3}, \ldots, \frac{2R}{3}\}$ that is relatively prime to R.
   Alice finds d such that $ed \equiv 1 \pmod{R}$.
4. **Alice** broadcasts (N, e). (Bob and Eve both see it.)
5. **Bob:** To send $m \in \{1, \ldots, N-1\}$, send $m^e \pmod{N}$.

RSA

Let L be a security parameter

1. **Alice** picks two primes p, q of length L. $N = pq$.
2. **Alice** computes $R = \phi(N) = \phi(pq) = (p-1)(q-1)$.
3. **Alice** picks an $e \in \{\frac{R}{3}, \ldots, \frac{2R}{3}\}$ that is relatively prime to R.
   Alice finds d such that $ed \equiv 1 \pmod{R}$.
4. **Alice** broadcasts (N, e). (Bob and Eve both see it.)
5. **Bob:** To send $m \in \{1, \ldots, N-1\}$, send $m^e \pmod{N}$.
6. If **Alice** gets $m^e \pmod{N}$ she computes

$$(m^e)^d \equiv m^{ed} \equiv m^{ed \pmod{R}} \equiv m^{1 \pmod{R}} \equiv m$$

Hardness Assumption for RSA

Recall If Alice and Bob do RSA and Eve observes:

Hardness Assumption for RSA

Recall If Alice and Bob do RSA and Eve observes:

1. Eve sees (N, e, m^e). The message is m.

Hardness Assumption for RSA

Recall If Alice and Bob do RSA and Eve observes:

1. Eve sees (N, e, m^e). The message is m.
2. Eve knows that there exists primes p, q such that $N = pq$, but she does not know what p, q are.

Hardness Assumption for RSA

Recall If Alice and Bob do RSA and Eve observes:

1. Eve sees (N, e, m^e). The message is m.
2. Eve knows that there exists primes p, q such that $N = pq$, but she does not know what p, q are.
3. Eve knows that e is relatively prime to $(p-1)(q-1)$.

Hardness Assumption for RSA

Recall If Alice and Bob do RSA and Eve observes:

1. Eve sees (N, e, m^e). The message is m.
2. Eve knows that there exists primes p, q such that $N = pq$, but she does not know what p, q are.
3. Eve knows that e is relatively prime to $(p-1)(q-1)$.

Definition: Let f be $f(N, e, m^e) = m$, where $N = pq$ and e has an inverse mod $(p-1)(q-1)$.

Hardness assumption (HA): f is hard to compute.

Plain RSA Bytes!

The RSA given above is referred to as **Plain RSA** .
Insecure! m is always coded as $m^e \pmod{N}$.

Plain RSA Bytes!

The RSA given above is referred to as **Plain RSA** .
Insecure! m is always coded as $m^e \pmod{N}$.

Make secure by padding: $m \in \{0,1\}^{L_1}$, $r \in \{0,1\}^{L_2}$.

Plain RSA Bytes!

The RSA given above is referred to as **Plain RSA** .
Insecure! m is always coded as $m^e \pmod{N}$.

Make secure by padding: $m \in \{0,1\}^{L_1}$, $r \in \{0,1\}^{L_2}$.

To send $m \in \{0,1\}^{L_1}$, pick rand $r \in \{0,1\}^{L_2}$, send $(rm)^e$.
(NOTE- rm means r CONCAT with m here and elsewhere.)
DEC: Alice finds rm and takes rightmost L_1 bits.
Caveat: RSA still has issues when used in real world. They have been fixed. Maybe.

Attacks on RSA

1. Factoring Algs we did: Jevons, Pollard ρ, Pollard $p-1$,
2. Factoring Algs we didn't do: Quad Sieve, Number Field Sieve.
3. Low-e attack, Same-N attacks.
4. There are also hardware and sociology attacks. We did not cover them, and could not have.

Attacks on RSA

1. Factoring Algs we did: Jevons, Pollard ρ, Pollard $p-1$,
2. Factoring Algs we didn't do: Quad Sieve, Number Field Sieve.
3. Low-e attack, Same-N attacks.
4. There are also hardware and sociology attacks. We did not cover them, and could not have.

These attacks tell Alice and Bob how to **up their game** .

Factoring Algorithms: Pollard ρ

Pollard ρ Algorithm

Define $f_c(x) \leftarrow x * x + c$. Looks random.

$x \leftarrow RAND(0, N-1)$, $c \leftarrow RAND(0, N-1)$, $y \leftarrow f_c(x)$
while TRUE
    $x \leftarrow f_c(x)$
    $y \leftarrow f_c(f_c(y))$
    $d \leftarrow GCD(x - y, N)$
    if $d \neq 1$ and $d \neq N$ then break
output(d)

Pollard ρ Algorithm: Thought Exp

Let p be the least prime that div N. We do not know p.

Pollard ρ Algorithm: Thought Exp

Let p be the least prime that div N. We do not know p.

The sequence x, $f_c(x)$, $f(f_c(x))$, ... is random-looking.

Pollard ρ Algorithm: Thought Exp

Let p be the least prime that div N. We do not know p.

The sequence x, $f_c(x)$, $f(f_c(x))$, $\ldots$ is random-looking.

Put each element of the seq into its $\equiv$ class mod p.

Pollard ρ Algorithm: Thought Exp

Let p be the least prime that div N. We do not know p.

The sequence x, $f_c(x)$, $f(f_c(x))$, $\ldots$ is random-looking.

Put each element of the seq into its $\equiv$ class mod p.

View the $\equiv$-classes as buckets at the sequence as balls.

Pollard ρ Algorithm: Thought Exp

Let p be the least prime that div N. We do not know p.

The sequence x, $f_c(x)$, $f(f_c(x))$, $\ldots$ is random-looking.

Put each element of the seq into its $\equiv$ class mod p.

View the $\equiv$-classes as buckets at the sequence as balls.

By Bday Paradox $\exists$ 2 elements of the seq in same bucket within the first $2\sqrt{p} \le 2N^{1/4}$ with high prob.

Pollard ρ Algorithm: Thought Exp

Let p be the least prime that div N. We do not know p.

The sequence x, $f_c(x)$, $f(f_c(x))$, $\ldots$ is random-looking.

Put each element of the seq into its $\equiv$ class mod p.

View the $\equiv$-classes as buckets at the sequence as balls.

By Bday Paradox $\exists$ 2 elements of the seq in same bucket within the first $2\sqrt{p} \le 2N^{1/4}$ with high prob.

By Thm there is an i such that the ith element in same bucket as $2i$th element, some $i \le 3N^{1/4}$, with high prob.

Pollard ρ Algorithm: Thought Exp

Let p be the least prime that div N. We do not know p.

The sequence x, $f_c(x)$, $f(f_c(x))$, ... is random-looking.

Put each element of the seq into its $\equiv$ class mod p.

View the $\equiv$-classes as buckets at the sequence as balls.

By Bday Paradox $\exists$ 2 elements of the seq in same bucket within the first $2\sqrt{p} \le 2N^{1/4}$ with high prob.

By Thm there is an i such that the ith element in same bucket as $2i$th element, some $i \le 3N^{1/4}$, with high prob.

Hence $(\exists x, y)[x \equiv y \pmod{p}]$ so $GCD(x - y, N) \neq 1$.

Pollard ρ Algorithm: Thought Exp

Let p be the least prime that div N. We do not know p.

The sequence x, $f_c(x)$, $f(f_c(x))$, ... is random-looking.

Put each element of the seq into its $\equiv$ class mod p.

View the $\equiv$-classes as buckets at the sequence as balls.

By Bday Paradox $\exists$ 2 elements of the seq in same bucket within the first $2\sqrt{p} \le 2N^{1/4}$ with high prob.

By Thm there is an i such that the ith element in same bucket as $2i$th element, some $i \le 3N^{1/4}$, with high prob.

Hence $(\exists x, y)[x \equiv y \pmod{p}]$ so $GCD(x - y, N) \ne 1$.

Caveat Need the sequence to be truly random to prove it works. Don't have that, but it works in practice.