Lecture 8: Cryptography

Trust No One.

Cryptography: Basic Set Up

Cryptography: Basic Set Up

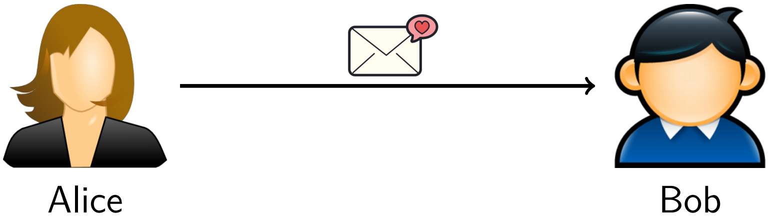
Alice

Cryptography: Basic Set Up

Alice

Bob

Cryptography: Basic Set Up



Cryptography: Basic Set Up

Cryptography: Basic Set Up

Cryptography: Basic Set Up

Goal: system st Bob gets the message, Eve doesn't

XOR

First scheme built on the XOR operation:

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

XOR

First scheme built on the XOR operation:

x	y	x ⊕ y
0	0	0
0	1	1
1	0	1
1	1	0

Claim: $(x \oplus b) \oplus b = x$ for any bits x, b

XOR

First scheme built on the XOR operation:

x	y	x ⊕ y
0	0	0
0	1	1
1	0	1
1	1	0

Claim: $(x \oplus b) \oplus b = x$ for any bits x, b
$b = 0$ doesn't flip, $b = 1$ flips twice

One-Time Pad

Alice wants to send an n-bit message m to Bob

One-Time Pad

Alice wants to send an n-bit message m to Bob

Setup:

- A and B generate random n-bit pad p

One-Time Pad

Alice wants to send an n-bit message m to Bob

Setup:

- A and B generate random n-bit pad p

Encryption:

- A creates ciphertext $c = E_p(m) := m \oplus p$

One-Time Pad

Alice wants to send an n-bit message m to Bob

Setup:

- A and B generate random n-bit pad p

Encryption:

- A creates ciphertext $c = E_p(m) := m \oplus p$

Decryption:

- B decrypts $m = D_p(c) := c \oplus p$

One-Time Pad

Alice wants to send an n-bit message m to Bob

Setup:

- A and B generate random n-bit pad p

Encryption:

- A creates ciphertext $c = E_p(m) := m \oplus p$

Decryption:

- B decrypts $m = D_p(c) := c \oplus p$

Does Bob receive the message correctly?
Can Eve read the message?

OTP Correctness

Claim: Bob always receives the message Alice sent.

OTP Correctness

Claim: Bob always receives the message Alice sent.

Formally: $\forall$ messages m & pads p, $D_p(E_p(m)) = m$

OTP Correctness

Claim: Bob always receives the message Alice sent.

Formally: $\forall$ messages m & pads p, $D_p(E_p(m)) = m$

Proof:

- $E_p(m) = m \oplus p$, so $D_p(E_p(m)) = (m \oplus p) \oplus p$

OTP Correctness

Claim: Bob always receives the message Alice sent.

Formally: $\forall$ messages m & pads p, $D_p(E_p(m)) = m$

Proof:

- $E_p(m) = m \oplus p$, so $D_p(E_p(m)) = (m \oplus p) \oplus p$
- Each bit of m XORed by same bit twice
- By previous claim, each bit of m stays the same

OTP Correctness

Claim: Bob always receives the message Alice sent.

Formally: $\forall$ messages m & pads p, $D_p(E_p(m)) = m$

Proof:

- $E_p(m) = m \oplus p$, so $D_p(E_p(m)) = (m \oplus p) \oplus p$
- Each bit of m XORed by same bit twice
- By previous claim, each bit of m stays the same
- Thus $D_p(E_p(m)) = m$

OTP Security

Claim: Any message possible just given ciphertext.

OTP Security

Claim: Any message possible just given ciphertext.

Formally: $\forall$ c & m, $\exists$ pad p st $E_p(m) = c$

OTP Security

Claim: Any message possible just given ciphertext.

Formally: $\forall$ c & m, $\exists$ pad p st $E_p(m) = c$
Proof:

- Take $p = c \oplus m$

OTP Security

Claim: Any message possible just given ciphertext.

Formally: $\forall\ c\ \&\ m,\ \exists$ pad p st $E_p(m) = c$

Proof:

- Take $p = c \oplus m$
- Then $E_p(m) = p \oplus m = (c \oplus m) \oplus m = c$

OTP Security

Claim: Any message possible just given ciphertext.

Formally: $\forall$ c & m, $\exists$ pad p st $E_p(m) = c$
Proof:

- Take $p = c \oplus m$
- Then $E_p(m) = p \oplus m = (c \oplus m) \oplus m = c$

Intuition: set $p_i = 1$ iff ith bit needs to flip

OTP Security

Claim: Any message possible just given ciphertext.

Formally: $\forall$ c & m, $\exists$ pad p st $E_p(m) = c$
Proof:

- Take $p = c \oplus m$
- Then $E_p(m) = p \oplus m = (c \oplus m) \oplus m = c$

Intuition: set $p_i = 1$ iff ith bit needs to flip

w/o pad, c says nothing about m!

Problems With OTP

How do Alice and Bob agree on their pad?

Problems With OTP

How do Alice and Bob agree on their pad?
Can't just send it over the channel!

Problems With OTP

How do Alice and Bob agree on their pad?
Can't just send it over the channel!

Secure only for a single message — can't reuse pad!

Problems With OTP

How do Alice and Bob agree on their pad?
Can't just send it over the channel!

Secure only for a single message — can't reuse pad!

Solve these issues with *public key cryptography*

Problems With OTP

How do Alice and Bob agree on their pad?
Can't just send it over the channel!

Secure only for a single message — can't reuse pad!

Solve these issues with *public key cryptography*

Idea: don't assume shared secret key
Have separate private (only Bob) and public keys

“Textbook” RSA Protocol

Alice wants to send an n-bit message m to Bob

"Textbook" RSA Protocol

Alice wants to send an n-bit message m to Bob

Setup:

- B chooses primes p, q st $pq > 2^n$
- B chooses e st $\gcd(e, (p-1)(q-1)) = 1$
- B publicizes $N = pq$ and e
- B keeps p, q, $d = e^{-1} \pmod{(p-1)(q-1)}$

"Textbook" RSA Protocol

Alice wants to send an n-bit message m to Bob

Setup:

- B chooses primes p, q st $pq > 2^n$
- B chooses e st $\gcd(e, (p-1)(q-1)) = 1$
- B publicizes $N = pq$ and e
- B keeps p, q, $d = e^{-1} \pmod{(p-1)(q-1)}$

Encryption:

- A encrypts $c = E_{N,e}(m) := m^e \pmod{N}$

"Textbook" RSA Protocol

Alice wants to send an n-bit message m to Bob

Setup:

- B chooses primes p, q st $pq > 2^n$
- B chooses e st $\gcd(e, (p-1)(q-1)) = 1$
- B publicizes $N = pq$ and e
- B keeps p, q, $d = e^{-1} \pmod{(p-1)(q-1)}$

Encryption:

- A encrypts $c = E_{N,e}(m) := m^e \pmod{N}$

Decryption:

- B decrypts $m = D_{N,d}(c) := c^d \pmod{N}$

Fermat's Little Theorem

Theorem: Let p be a prime and $a \not\equiv 0 \pmod{p}$. Then $a^{p-1} \equiv 1 \pmod{p}$.

Fermat's Little Theorem

Theorem: Let p be a prime and $a \not\equiv 0 \pmod{p}$. Then $a^{p-1} \equiv 1 \pmod{p}$.

Proof:

- Consider set $S_p = \{1, 2, 3, ..., p-1\}$

Fermat's Little Theorem

Theorem: Let p be a prime and $a \not\equiv 0 \pmod{p}$.
Then $a^{p-1} \equiv 1 \pmod{p}$.

Proof:

- Consider set $S_p = \{1, 2, 3, ..., p-1\}$
- Claim: $f(x) = ax \pmod{p}$ is bijection $S_p \to S_p$

Fermat's Little Theorem

Theorem: Let p be a prime and $a \not\equiv 0 \pmod{p}$. Then $a^{p-1} \equiv 1 \pmod{p}$.

Proof:

- Consider set $S_p = \{1, 2, 3, ..., p-1\}$
- Claim: $f(x) = ax \pmod{p}$ is bijection $S_p \to S_p$
- $\{1, 2, ..., p-1\} = \{a, 2a, ..., (p-1)a\} \pmod{p}$

Fermat's Little Theorem

Theorem: Let p be a prime and $a \not\equiv 0 \pmod{p}$.
Then $a^{p-1} \equiv 1 \pmod{p}$.

Proof:

- Consider set $S_p = \{1, 2, 3, ..., p-1\}$
- Claim: $f(x) = ax \pmod{p}$ is bijection $S_p \to S_p$
- $\{1, 2, ..., p-1\} = \{a, 2a, ..., (p-1)a\} \pmod{p}$
- Means $\prod_i i \equiv \prod_i ia \equiv a^{p-1} \prod_i i \pmod{p}$

Fermat's Little Theorem

Theorem: Let p be a prime and $a \not\equiv 0 \pmod{p}$. Then $a^{p-1} \equiv 1 \pmod{p}$.

Proof:

- Consider set $S_p = \{1, 2, 3, ..., p-1\}$
- Claim: $f(x) = ax \pmod{p}$ is bijection $S_p \to S_p$
- $\{1, 2, ..., p-1\} = \{a, 2a, ..., (p-1)a\} \pmod{p}$
- Means $\prod_i i \equiv \prod_i ia \equiv a^{p-1} \prod_i i \pmod{p}$
- Multiply by $\prod_i i^{-1}$, get $1 \equiv a^{p-1} \pmod{p}$

Proof Of Claim

To finish FLT proof, need to prove:
Claim: $f(x) = ax \pmod{p}$ is bijection $S_p \to S_p$
Proof:

- Need that for $x \in S_p$, $f(x) \in S_p$

Proof Of Claim

To finish FLT proof, need to prove:
Claim: $f(x) = ax \pmod{p}$ is bijection $S_p \to S_p$
Proof:

- Need that for $x \in S_p$, $f(x) \in S_p$
  - If $x \in S_p$, $p \not| x$
  - $p \not| a$ either, so $p \not| ax$
  - Hence $ax \pmod{p} \in S_p$

Proof Of Claim

To finish FLT proof, need to prove:
Claim: $f(x) = ax \pmod{p}$ is bijection $S_p \to S_p$
Proof:

- Need that for $x \in S_p$, $f(x) \in S_p$
  - If $x \in S_p$, $p \not| x$
  - $p \not| a$ either, so $p \not| ax$
  - Hence $ax \pmod{p} \in S_p$
- Inverse is $f^{-1}(y) = a^{-1}y \pmod{p}$

Proof Of Claim

To finish FLT proof, need to prove:
Claim: $f(x) = ax \pmod{p}$ is bijection $S_p \to S_p$
Proof:

- Need that for $x \in S_p$, $f(x) \in S_p$
  - If $x \in S_p$, $p \not| x$
  - $p \not| a$ either, so $p \not| ax$
  - Hence $ax \pmod{p} \in S_p$
- Inverse is $f^{-1}(y) = a^{-1}y \pmod{p}$
  - $f^{-1}(f(x)) \equiv a^{-1}ax \equiv x \pmod{p}$
  - $f(f^{-1}(x)) \equiv aa^{-1}x \equiv x \pmod{p}$

RSA Correctness

Theorem: RSA protocol always decrypts correctly.

RSA Correctness

Theorem: RSA protocol always decrypts correctly.

Formally: $\forall$ p, q, e, and m, $D_{N,d}(E_{N,e}(m)) = m$

RSA Correctness

Theorem: RSA protocol always decrypts correctly.

Formally: $\forall$ p, q, e, and m, $D_{N,d}(E_{N,e}(m)) = m$

Proof:

- Note: $D(E(m)) = m^{ed} \bmod N$
- So just need to prove $m^{ed} \equiv m \pmod{N}$

RSA Correctness

Theorem: RSA protocol always decrypts correctly.

Formally: $\forall$ p, q, e, and m, $D_{N,d}(E_{N,e}(m)) = m$

Proof:

- Note: $D(E(m)) = m^{ed} \bmod N$
- So just need to prove $m^{ed} \equiv m \pmod{N}$
- $ed = 1 + k(p-1)(q-1)$
- So $m^{ed} = (m^{(p-1)})^{k(q-1)} m \equiv m \pmod{p}$

RSA Correctness

Theorem: RSA protocol always decrypts correctly.

Formally: $\forall$ p, q, e, and m, $D_{N,d}(E_{N,e}(m)) = m$

Proof:

- Note: $D(E(m)) = m^{ed} \bmod N$
- So just need to prove $m^{ed} \equiv m \pmod{N}$
- $ed = 1 + k(p-1)(q-1)$
- So $m^{ed} = (m^{(p-1)})^{k(q-1)} m \equiv m \pmod{p}$
- Similarly, have $m^{ed} \equiv m \pmod{q}$

RSA Correctness

Theorem: RSA protocol always decrypts correctly.

Formally: $\forall$ p, q, e, and m, $D_{N,d}(E_{N,e}(m)) = m$

Proof:

- Note: $D(E(m)) = m^{ed} \bmod N$
- So just need to prove $m^{ed} \equiv m \pmod{N}$
- $ed = 1 + k(p-1)(q-1)$
- So $m^{ed} = (m^{(p-1)})^{k(q-1)} m \equiv m \pmod{p}$
- Similarly, have $m^{ed} \equiv m \pmod{q}$
- $m^{ed} \equiv m \pmod{pq}$ is solution to those two
- CRT: m is *only* solution!

RSA Efficiency

Need protocol to run quickly
For security, p and q often 512 bits or more.

RSA Efficiency

Need protocol to run quickly
For security, p and q often 512 bits or more.

Setup: need to sample p and q (next slide)

RSA Efficiency

Need protocol to run quickly
For security, p and q often 512 bits or more.

Setup: need to sample p and q (next slide)
Setup: need to invert e to get d

- EGCD runs in log time!

RSA Efficiency

Need protocol to run quickly
For security, p and q often 512 bits or more.

Setup: need to sample p and q (next slide)
Setup: need to invert e to get d

- EGCD runs in log time!

Encryption: need to find $m^e \pmod{N}$

- Repeated squaring runs in log time!

RSA Efficiency

Need protocol to run quickly
For security, p and q often 512 bits or more.

Setup: need to sample p and q (next slide)
Setup: need to invert e to get d

- EGCD runs in log time!

Encryption: need to find $m^e \pmod{N}$

- Repeated squaring runs in log time!

Decryption: need to find $c^d \pmod{N}$

- Again use repeated squaring!

Sampling Primes

How to find primes p and q?
Can't use the same ones for every key!

Sampling Primes

How to find primes p and q?
Can't use the same ones for every key!

Theorem: Num primes $\leq n$ at least $\frac{n}{\ln(n)}$

Sampling Primes

How to find primes p and q?
Can't use the same ones for every key!

Theorem: Num primes $\leq n$ at least $\frac{n}{\ln(n)}$

Means we can guess randomly until we find one!

Sampling Primes

How to find primes p and q?
Can't use the same ones for every key!

Theorem: Num primes $\leq n$ at least $\frac{n}{\ln(n)}$

Means we can guess randomly until we find one!
Note: can quickly test primality

Time For A Break

4 minute breather!

Time For A Break

4 minute breather!

Today's Discussion Question:
What is the best kind of sandwich?

RSA Security

Correctness and efficiency great; need security too

RSA Security

Correctness and efficiency great; need security too

Open problem in Computer Science!

RSA Security

Correctness and efficiency great; need security too

Open problem in Computer Science!
Generally accepted as secure, but no proof (yet)

RSA Security

Correctness and efficiency great; need security too

Open problem in Computer Science!
Generally accepted as secure, but no proof (yet)

Can easily break if factor N into p and q
But naïve factoring too slow if p and q big

RSA Security

Correctness and efficiency great; need security too

Open problem in Computer Science!
Generally accepted as secure, but no proof (yet)

Can easily break if factor N into p and q
But naïve factoring too slow if p and q big

Note: can factor quickly on quantum computers
Not an immediate issue, but may be in the future!

Breaking Textbook RSA

Even if RSA secure, need careful implementation

Breaking Textbook RSA

Even if RSA secure, need careful implementation

Ex: suppose my credit card number is m
I send Amazon $E(m)$ to make a purchase

Breaking Textbook RSA

Even if RSA secure, need careful implementation

Ex: suppose my credit card number is m
I send Amazon $E(m)$ to make a purchase

Alice can't recover m from $E(m)$...

Breaking Textbook RSA

Even if RSA secure, need careful implementation

Ex: suppose my credit card number is m
I send Amazon $E(m)$ to make a purchase

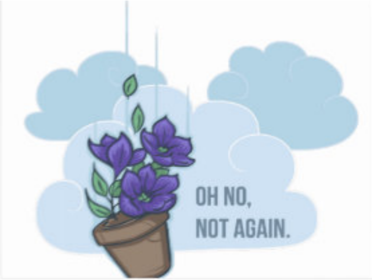
Alice can't recover m from $E(m)$...
...but what if she sends $E(m)$ to Amazon?

Breaking Textbook RSA

Even if RSA secure, need careful implementation

Ex: suppose my credit card number is m
I send Amazon $E(m)$ to make a purchase

Alice can't recover m from $E(m)$...
...but what if she sends $E(m)$ to Amazon?



Defense Against Replay Attacks

Last slide was a *replay attack*

Defense Against Replay Attacks

Last slide was a *replay attack*

Fix: pad message with a bunch of randomness
If Amazon gets same message twice, reject

Defense Against Replay Attacks

Last slide was a *replay attack*

Fix: pad message with a bunch of randomness
If Amazon gets same message twice, reject

Moral: even secure protocol can be vulnerable!

Digital Signature Scheme

Alternate use of RSA: proof of identity

Digital Signature Scheme

Alternate use of RSA: proof of identity

"Amazon" wants to send me a message.
How do I know it's actually Amazon?

Digital Signature Scheme

Alternate use of RSA: proof of identity

“Amazon” wants to send me a message.
How do I know it’s actually Amazon?

Idea: Amazon sends $s = m^d \pmod{N}$ along with m

Digital Signature Scheme

Alternate use of RSA: proof of identity

"Amazon" wants to send me a message.
How do I know it's actually Amazon?

Idea: Amazon sends $s = m^d \pmod{N}$ along with m
I can verify $s^e \equiv m \pmod{N}$

Digital Signature Scheme

Alternate use of RSA: proof of identity

“Amazon” wants to send me a message.
How do I know it’s actually Amazon?

Idea: Amazon sends $s = m^d \pmod{N}$ along with m
I can verify $s^e \equiv m \pmod{N}$

Only Amazon can sign consistently!
Ability to sign $\equiv$ ability to decrypt

Digital Signature Attack

Eve: I choose message to sign to prevent cheating!

Digital Signature Attack

Eve: I choose message to sign to prevent cheating!
Amazon: ok...

Digital Signature Attack

Eve: I choose message to sign to prevent cheating!
Amazon: ok...
Eve: Sign $r^e E(m)$ pls

Digital Signature Attack

Eve: I choose message to sign to prevent cheating!
Amazon: ok...
Eve: Sign $r^e E(m)$ pls
Amazon: $(r^e E(m))^d \pmod{N}$

Digital Signature Attack

Eve: I choose message to sign to prevent cheating!
Amazon: ok...
Eve: Sign $r^e E(m)$ pls
Amazon: $(r^e E(m))^d \pmod{N}$

What can Eve now do?

Digital Signature Attack

Eve: I choose message to sign to prevent cheating!
Amazon: ok...
Eve: Sign $r^e E(m)$ pls
Amazon: $(r^e E(m))^d \pmod{N}$

What can Eve now do?
$(r^e E(m))^d \equiv r^{ed} m^{ed} \equiv rm \pmod{N}$

Digital Signature Attack

Eve: I choose message to sign to prevent cheating!
Amazon: ok...
Eve: Sign $r^e E(m)$ pls
Amazon: $(r^e E(m))^d \pmod{N}$

What can Eve now do?
$(r^e E(m))^d \equiv r^{ed} m^{ed} \equiv rm \pmod{N}$

Uh oh — Eve knows r, so can invert to get m!

Digital Signature Attack

Eve: I choose message to sign to prevent cheating!
Amazon: ok...
Eve: Sign $r^e E(m)$ pls
Amazon: $(r^e E(m))^d \pmod N$

What can Eve now do?
$(r^e E(m))^d \equiv r^{ed} m^{ed} \equiv rm \pmod N$

Uh oh — Eve knows r, so can invert to get m!

Moral: don't sign arbitrary messages

Fin

Next time: polynomials!