Lecture 10: Cryptography

Credit: https://xkcd.com/177/

Credit: Sagnik!
Basic Setup

Credit: https://flylib.com/books/en/1.581.1.188/1/
Recall the XOR operation:

\[
\begin{array}{c|c|c|c}
    x & y & x \oplus y & (x \oplus b) \oplus b \\
    \hline
    0 & 0 & 0 & 0 \\
    0 & 1 & 1 & 0 \\
    1 & 0 & 1 & 0 \\
    1 & 1 & 0 & 1 \\
\end{array}
\]

Notice that for any bits \( x, b \) we have \((x \oplus b) \oplus b = x\).
One-Time Pad

Alice (the sender) wants to send a $n$-bit message $m$ to Bob (the receiver).

Setup:
- Alice and Bob generate a random key $k$.

Encryption:

Decryption:

Notice that $D(E(m)) = (m \oplus k) \oplus k = m$, i.e. Bob always receives the message Alice sent.
One-Time Pad

Alice (the sender) wants to send a \( n \)-bit message \( m \) to Bob (the receiver).

Setup:
- Alice and Bob generate a random key \( k \).

Encryption:
- Alice encrypts \( c = E(m) := m \oplus k \).

Decryption:

Notice that \( D(E(m)) = (m \oplus k) \oplus k = m \), i.e. Bob always receives the message Alice sent.
One-Time Pad

Alice (the sender) wants to send a $n$-bit message $m$ to Bob (the receiver).

**Setup:**
- Alice and Bob generate a random key $k$.

**Encryption:**
- Alice encrypts $c = E(m) := m \oplus k$.

**Decryption:**
- Bob decrypts $D(c) := c \oplus k$.

Notice that $D(E(m)) = (m \oplus k) \oplus k = m$, i.e. Bob always receives the message Alice sent.
One-Time Pad: Disadvantages

One-Time Pad is the only existing mathematically unbreakable encryption. But if only one of the following is not met, it is no longer unbreakable:

- $k$ is at least as long as $m$;
One-Time Pad: Disadvantages

One-Time Pad is the only existing mathematically unbreakable encryption. But if only one of the following is not met, it is no longer unbreakable:

- $k$ is at least as long as $m$;
- $k$ truly random (not generated by a simple computer function);
One-Time Pad: Disadvantages

One-Time Pad is the only existing mathematically unbreakable encryption. But if only one of the following is not met, it is no longer unbreakable:

- $k$ is at least as long as $m$;
- $k$ truly random (not generated by a simple computer function);
- each key is used only once;
One-Time Pad: Disadvantages

One-Time Pad is the only existing mathematically unbreakable encryption. But if only one of the following is not met, it is no longer unbreakable:

- $k$ is at least as long as $m$;
- $k$ truly random (not generated by a simple computer function);
- each key is used only once;
- there should only be two copies of the key; one for Alice and one for Bob.
One-Time Pad: Disadvantages

One-Time Pad is the only existing mathematically unbreakable encryption. But if only one of the following is not met, it is no longer unbreakable:

- $k$ is at least as long as $m$;
- $k$ truly random (not generated by a simple computer function);
- each key is used only once;
- there should only be two copies of the key; one for Alice and one for Bob.

But what if I (Alice) want to send my credit card information to Amazon (Bob) to make a purchase?

- Not practical; I would need to somehow communicate with Amazon to agree on a key for every single purchase.
One-Time Pad: Disadvantages

One-Time Pad is the only existing mathematically unbreakable encryption. But if only one of the following is not met, it is no longer unbreakable:

- $k$ is at least as long as $m$;
- $k$ truly random (not generated by a simple computer function);
- each key is used only once;
- there should only be two copies of the key; one for Alice and one for Bob.

But what if I (Alice) want to send my credit card information to Amazon (Bob) to make a purchase?

- Not practical; I would need to somehow communicate with Amazon to agree on a key for every single purchase.
- And every single user would’ve had to do this.
One-Time Pad: Disadvantages

One-Time Pad is the only existing mathematically unbreakable encryption. But if only one of the following is not met, it is no longer unbreakable:

- $k$ is at least as long as $m$;
- $k$ truly random (not generated by a simple computer function);
- each key is used only once;
- there should only be two copies of the key; one for Alice and one for Bob.

Solve these issues with public-key cryptography: use pairs of keys

- public keys: everyone knows!
One-Time Pad: Disadvantages

One-Time Pad is the only existing mathematically unbreakable encryption. But if only one of the following is not met, it is no longer unbreakable:

- $k$ is at least as long as $m$;
- $k$ truly random (not generated by a simple computer function);
- each key is used only once;
- there should only be two copies of the key; one for Alice and one for Bob.

Solve these issues with *public-key cryptography*: use pairs of keys

- **public keys**: everyone knows!
- **private keys**: only Bob knows.
RSA Protocol

Everyone can send messages to Bob. For now, let’s say Alice wants to send a message $m$ to Bob.

**Setup:**
- Bob chooses two large (2048-bit) distinct primes $p, q$.

**Encryption:**

**Decryption:**
RSA Protocol

Everyone can send messages to Bob. For now, let’s say Alice wants to send a message $m$ to Bob.

**Setup:**
- Bob chooses two large (2048-bit) distinct primes $p, q$.
- Bob chooses $e$ such that $\text{gcd}(e, (p - 1)(q - 1)) = 1$.

**Encryption:**

**Decryption:**
RSA Protocol

Everyone can send messages to Bob.
For now, let’s say Alice wants to send a message $m$ to Bob.

Setup:

- Bob chooses two large (2048-bit) distinct primes $p, q$.
- Bob chooses $e$ such that $\gcd(e, (p - 1)(q - 1)) = 1$.
- The public key is $(N, e)$, where $N = pq$.

Encryption:

Decryption:
RSA Protocol

Everyone can send messages to Bob.
For now, let’s say Alice wants to send a message $m$ to Bob.

**Setup:**

- Bob chooses two large (2048-bit) distinct primes $p, q$.
- Bob chooses $e$ such that $\gcd(e, (p - 1)(q - 1)) = 1$.
- The public key is $(N, e)$, where $N = pq$.
- Bob computes the private key $d := e^{-1} \mod (p - 1)(q - 1)$.

**Encryption:**

**Decryption:**
RSA Protocol

Everyone can send messages to Bob.
For now, let’s say Alice wants to send a message $m$ to Bob.

Setup:
- Bob chooses two large (2048-bit) distinct primes $p, q$.
- Bob chooses $e$ such that $\gcd(e, (p - 1)(q - 1)) = 1$.
- the public key is $(N, e)$, where $N = pq$.
- Bob computes the private key $d := e^{-1} \mod (p - 1)(q - 1)$.

Encryption:
- Alice encrypts $c = E(m) := m^e \mod N$

Decryption:
RSA Protocol

Everyone can send messages to Bob.
For now, let’s say Alice wants to send a message $m$ to Bob.

**Setup:**
- Bob chooses two large (2048-bit) distinct primes $p, q$.
- Bob chooses $e$ such that $\gcd(e, (p - 1)(q - 1)) = 1$.
- the public key is $(N, e)$, where $N = pq$.
- Bob computes the private key $d := e^{-1} \mod (p - 1)(q - 1)$.

**Encryption:**
- Alice encrypts $c = E(m) := m^e \mod N$

**Decryption:**
- Bob decrypts $D(c) := c^d \mod N$

```plaintext
>>> c^d % N
```
We need to analyze:

- Correctness: $D(E(m)) = m$?
We need to analyze:

- Correctness: $D(E(m)) = m$?
- Efficiency: Can Alice and Bob perform their steps efficiently?
# TODO

We need to analyze:

- Correctness: $D(E(m)) = m$?
- Efficiency: Can Alice and Bob perform their steps efficiently?
- Security: Can Eve break it?
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.** Let $f : \{0, 1, 2, \ldots, p-1\} \to \{0, 1, \ldots, p-1\}$ with $x \mapsto ax \mod p$.

Since $f(0) = 0$, $a \equiv 0 \pmod{p} = 0$, $\{1, 2, \ldots, p-1\} = \{f(1), \ldots, f(p-1)\}$.

For all $x = 1, \ldots, p-1$, $f(x) \equiv ax \pmod{p}$.

$$\prod_{x=1}^{p-1} x = \prod_{x=1}^{p-1} f(x) \equiv \prod_{x=1}^{p-1} (ax) \equiv a \prod_{x=1}^{p-1} x \pmod{p}$$

Since $p$ is a prime, $\gcd(p, x) = 1 \Rightarrow x^{-1} \pmod{p}$ exists.

$$(\prod_{x=1}^{p-1} x^{-1})(\prod_{x=1}^{p-1} x) \equiv a^{p-1}(\prod_{x=1}^{p-1} x)(\prod_{x=1}^{p-1} x^4) \pmod{p} \Rightarrow 1 \equiv a^{p-1} \pmod{p}$$
Goal: $D(E(m)) = m$.

\[
\left( m^e \mod N \right)^d \mod N \neq m
\]

Notice that $0 \leq D(E(m)) \leq N-1$, so only need to show $D(E(m)) \equiv m \pmod{N}$.

\[
E(m) = m^e \mod N \equiv m^e \pmod{N}
\]

\[
D(c) = c^d \mod N \equiv c^d \pmod{N}
\]

\[
D\left( E(m) \right) \equiv E(m)^d \equiv (m^e)^d = m^{ed} \pmod{N}
\]

Goal: $med \equiv m \pmod{N}$

$m, n$ are coprime.

\[
X \equiv 3 \pmod{n}
\]

\[
X \equiv 3 \pmod{m}
\]

Find me a solution!!

\[
X = 3
\]

Find me all solutions!!

\[
3 + (mn)k, \; k \in \mathbb{Z}
\]
RSA correctness

**Theorem:** Let $D$, $E$ be the RSA decryption and RSA encryption functions respectively. Then $D(E(m)) = m$, i.e. RSA protocol always decrypts correctly.

**Proof.** Let $x = m^{ed}$. Then $x \equiv m \pmod{N}$.

Since $ed \equiv 1 \pmod{(p-1)(q-1)}$, so $\exists k \in \mathbb{Z}$, $ed - 1 = k(p-1)(q-1)$.

Then $x = \left( m^{(p-1)(q-1)} \right)^k \cdot m^{p-1} = m \cdot m^{k(p-1)(q-1)} \equiv m \pmod{N}$.

Thus, $x \equiv m \pmod{p}$ and $x \equiv m \pmod{q}$.

Notice that $x = m$ is a solution.
RSA Efficiency

Setup

- Bob chooses two large distinct primes \( p \) and \( q \).
  
  how???

\[ e \text{ s.t. } \gcd(e, (p-1)(q-1)) = 1 \]

Encryption:

Decryption:
RSA Efficiency

Setup

- Bob chooses two large distinct primes $p$ and $q$. how???
- Bob chooses $e$ such that $\gcd(e, (p - 1)(q - 1)) = 1$. how???(choose a prime, like 3)

$$e^{-1} \mod (p-1)(q-1).$$

Encryption:

Decryption:
RSA Efficiency

Setup

- Bob chooses two large distinct primes $p$ and $q$.
  how???
- Bob chooses $e$ such that $\gcd(e, (p - 1)(q - 1)) = 1$.
  how?? (choose a prime, like 3)
- Bob computes $d := e^{-1} \mod (p - 1)(q - 1)$.
  how?? (extended Euclidean algorithm is fast!)

Encryption:
\[ E(m) = m^e \mod N. \]

Decryption:
RSA Efficiency

Setup

- Bob chooses two large distinct primes $p$ and $q$.  
  how???
- Bob chooses $e$ such that $\gcd(e, (p - 1)(q - 1)) = 1$.  
  how??? (choose a prime, like 3)
- Bob computes $d := e^{-1} \mod (p - 1)(q - 1)$.  
  how??? (extended Euclidean algorithm is fast!)

Encryption:

- Alice encrypts $c = E(m) := m^e \mod N$.  
  how??? (repeated squaring is fast!)

Decryption:

$$D(c) = c^d \mod N$$
RSA Efficiency

Setup

- Bob chooses two large distinct primes $p$ and $q$. 
  how???
- Bob chooses $e$ such that $\gcd(e, (p - 1)(q - 1)) = 1$. 
  how??? (choose a prime, like 3)
- Bob computes $d := e^{-1} \mod (p - 1)(q - 1)$. 
  how??? (extended Euclidean algorithm is fast!)

Encryption:

- Alice encrypts $c = E(m) := m^e \mod N$. 
  how??? (repeated squaring is fast!)

Decryption:

- Bob decrypts $D(c) := c^d \mod N$. 
  how??? (repeated squaring is fast!)
We need two large (2048-bit) primes.

- By the Prime Number Theorem, number of primes \( \leq N \) is at least \( \frac{N}{\ln(N)} \).
We need two large (2048-bit) primes.

- By the Prime Number Theorem, number of primes \( \leq N \) is at least \( \frac{N}{\ln(N)} \).

- We need to generate and check \( \approx \ln N \) primes. This is linear in the number of bits of \( N \).
RSA Efficiency: Sampling Primes

We need two large (2048-bit) primes.

- By the Prime Number Theorem, number of primes $\leq N$ is at least $\frac{N}{\ln(N)}$.
- We need to generate and check $\approx \ln N$ primes. This is linear in the number of bits of $N$.
- ...but how to check primes?
RSA Efficiency: Sampling Primes

We need two large (2048-bit) primes.

- By the Prime Number Theorem, number of primes \( \leq N \) is at least \( \frac{N}{\ln(N)} \).
- We need to generate and check \( \approx \ln N \) primes. This is linear in the number of bits of \( N \).
- ...but how to check primes?
- there is an efficient algorithm that tests if \( N \) is prime (polynomial time in the number of bits of \( N \)).
Cryptograph relies on assumptions.

**RSA Assumption**: Given $N$, $e$, and $m^e \mod N$, there is no efficient algorithm for finding $m$.

We believe Eve cannot break RSA.

- Eve can break RSA by factoring $N = pq$ to get $(p - 1)(q - 1)$ to compute $d$. 
RSA Security

Cryptograph relies on assumptions.

**RSA Assumption:** Given $N, e$, and $m^e \mod N$, there is no efficient algorithm for finding $m$.

We believe Eve cannot break RSA.

- Eve can break RSA by factoring $N = pq$ to get $(p - 1)(q - 1)$ to compute $d$.
- But prime factorization is hard!
RSA Security

Cryptograph relies on assumptions.

**RSA Assumption:** Given $N$, $e$, and $m^e \mod N$, there is no efficient algorithm for finding $m$.

We believe Eve cannot break RSA.

- Eve can break RSA by factoring $N = pq$ to get $(p - 1)(q - 1)$ to compute $d$.
- But prime factorization is hard!
- For large $N$, no efficient, non-quantum algorithm is known.
Replay Attack

Does Eve really need to know $d$ to attack?

- Suppose my credit card number is $m$. 
Replay Attack

Does Eve really need to know $d$ to attack?

- Suppose my credit card number is $m$.
- I send Amazon $E(m)$ to make a purchase.
Replay Attack

Does Eve really need to know $d$ to attack?

- Suppose my credit card number is $m$.
- I send Amazon $E(m)$ to make a purchase.
- Eve can’t recover $m$ from $E(m)$. 
Replay Attack

Does Eve really need to know $d$ to attack?

- Suppose my credit card number is $m$.
- I send Amazon $E(m)$ to make a purchase.
- Eve can’t recover $m$ from $E(m)$.
- But Eve was listening to our communication and now she knows $E(m)$.
Replay Attack

Does Eve really need to know $d$ to attack?

- Suppose my credit card number is $m$.
- I send Amazon $E(m)$ to make a purchase.
- Eve can’t recover $m$ from $E(m)$.
- But Eve was listening to our communication and now she knows $E(m)$.
- Eve sends $E(m)$ to Amazon.
Replay Attack

Does Eve really need to know $d$ to attack?

- Suppose my credit card number is $m$.
- I send Amazon $E(m)$ to make a purchase.
- Eve can’t recover $m$ from $E(m)$.
- But Eve was listening to our communication and now she knows $E(m)$.
- Eve sends $E(m)$ to Amazon.
- Now Eve can use my credit card.
Defense Against Replay Attacks

Even secure protocol can be vulnerable, need careful implementation.

To defend against replay attacks,
  ➤ before encrypt $m$, randomly generate a string $s$. 

Send $E(\text{concatenate}(m, s))$.
If Amazon gets same message twice, reject.
Defense Against Replay Attacks

Even secure protocol can be vulnerable, need careful implementation.

To defend against replay attacks,
  ▶ before encrypt $m$, randomly generate a string $s$.
  ▶ Send $E(\text{concatenate}(m, s))$. 
Defense Against Replay Attacks

Even secure protocol can be vulnerable, need careful implementation.

To defend against replay attacks,

▸ before encrypt $m$, randomly generate a string $s$.
▸ Send $E(\text{concatenate}(m, s))$.
▸ If Amazon gets same message twice, reject.
Flipping RSA: Digital Signature

RSA can be used as in *proof of identity*.

- How does Alice know the receiver is Bob?
RSA can be used as in *proof of identity*.

- How does Alice know the receiver is Bob?
- Bob could prove his identity by showing Alice $d$, but he doesn’t want to do that.
Flipping RSA: Digital Signature

RSA can be used as in *proof of identity*.

- How does Alice know the receiver is Bob?
- Bob could prove his identity by showing Alice $d$, but he doesn’t want to do that.
- Alice chooses a message $m$ and asks Bob to send her $m^d \mod N$. 
Flipping RSA: Digital Signature

RSA can be used as in *proof of identity*.

- How does Alice know the receiver is Bob?
- Bob could prove his identity by showing Alice $d$, but he doesn’t want to do that.
- Alice chooses a message $m$ and asks Bob to send her $m^d \mod N$.
- Alice can verify $(m^d)^e \equiv m \pmod{N}$.

\[
E(D(m)) = D(E(m)) = m
\]
Digital Signature Attack

Should Bob sign arbitrary messages?

- Alice encrypts a top-secret message \( m \) and sends it to Bob.
Digital Signature Attack

Should Bob sign arbitrary messages?

- Alice encrypts a top-secret message $m$ and sends it to Bob.
- Eve intercepts the cipher $E(m)$.
Digital Signature Attack

Should Bob sign arbitrary messages?

- Alice encrypts a top-secret message $m$ and sends it to Bob.
- Eve intercepts the cipher $E(m)$.
- Eve chooses a number $r$ and asks Bob to sign $r^eE(m)$. 
Digital Signature Attack

Should Bob sign arbitrary messages?

- Alice encrypts a top-secret message $m$ and sends it to Bob.
- Eve intercepts the cipher $E(m)$.
- Eve chooses a number $r$ and asks Bob to sign $r^eE(m)$.
- Bob agrees and sends Eve $(r^eE(m))^d \mod N$. 

Eve knows $r$; so Eve computes $r^e \mod N$ to recover $m$. 


Digital Signature Attack

Should Bob sign arbitrary messages?

- Alice encrypts a top-secret message $m$ and sends it to Bob.
- Eve intercepts the cipher $E(m)$.
- Eve chooses a number $r$ and asks Bob to sign $r^e E(m)$.
- Bob agrees and sends Eve $(r^e E(m))^d \mod N$.
- Now Eve knows $(r^e E(m))^d \equiv r^{ed} m^{ed} \equiv rm \pmod{N}$.
Digital Signature Attack

Should Bob sign arbitrary messages?

- Alice encrypts a top-secret message $m$ and sends it to Bob.
- Eve intercepts the cipher $E(m)$.
- Eve chooses a number $r$ and asks Bob to sign $r^eE(m)$.
- Bob agrees and sends Eve $(r^eE(m))^d \mod N$.
- Now Eve knows $(r^eE(m))^d \equiv r^{ed}m^{ed} \equiv rm \pmod N$.
- Eve knows $r$; so Eve computes $r^{-1} \pmod N$ to recover $m$. 
THE END!

Thank you for coming!