

Week 5 - Friday

COMP 4290

Last time

- What did we talk about last time?
- Hash functions
- Birthday paradox

Questions?

Assignment 2

Project 2

Jennifer Perez Presents

Attacks Against Hash Functions

Birthday attack's revenge

- If a hash value is made up of 8 bytes
 - 8 bytes = 64 bits
 - $2^{64} = 18446744073709551616$
 - So, we need to check one hash against 2^{63} other hashes to have a 50% probability of matching
- But, by the birthday paradox

$$k \approx \sqrt{2 \ln(2) 2^{64}} \approx 1.18 \cdot 2^{32}$$

- We need a much smaller number to get a collision!

Theoretically awesome ...

- 2^{32} operations is within the reach of modern computers in seconds, minutes, or hours
- Collisions do not guarantee that the system can be hacked
- A collision with an existing password is necessary
- If the system has 2^{32} users, there's a good probability that two of them have the same password hash

Practical issues

- Real hashes are usually longer
- Salt makes things a bit more complex
- Most people use weak passwords
- It's easier to guess them
- Social engineering
 - Shoulder surfing
 - Finding the Post-It with their password
- Still, the mathematical possibilities are interesting ...

Application: Digital Signature Attacks

Signing hashes

- Sometimes a document needs to be digitally signed
 - Contracts
 - Commitment schemes
- Hashes are often signed so that there's less data to sign and transmit
 - Signatures usually use public key crypto like RSA
- But, if the length of the hash is not very long, we can employ the Birthday Paradox

Signing scheme example

- Digital signature schemes typically employ public key cryptography
- We need the one way property so that we can verify that it works without being able to break it
- **Full Domain Hash** uses RSA to do this:
 - Given message M , we find $H(M)$, then raise $H(M)$ to the secret decryption exponent to find signature S
 - $S = H(M)^d \bmod n$
 - To verify the signature, take the signature and raise it to the publicly known encryption exponent e and compare that to the hash of the message
- If $S^e \bmod n = H(M)$, we feel reasonably sure of two things:
 - S is a signature for a M (M has not been changed)
 - Only someone who knows the private key for n could have signed it
- Why does it use the hash of the message instead of the message?

Same rules, different game

- Erica wants to buy a house from Carmen (and cheat her)
- Let's say that they are going to electronically sign a 64-bit hash of the contract
- Erica creates 2^{32} variations on a contract that Carmen will agree to
- Erica creates 2^{32} variations on a contract which is highly advantageous to Erica
- Sound ridiculous?

2¹⁶ contracts without trying

I, Erica, {hereby, through this document}, {agree, consent} to {purchase, buy} the {property, estate} {located, which can be found} at 742 Evergreen Terrace.

A {fair, equitable} {sale, asking} price for this {property, estate} is the {sum, amount} of \$500,000. This {document, contract} states that this {sum, amount} is {to be paid, payable} on September 19, 2025 by Erica to Carmen in {exchange, return} for {ownership, possession} of the {aforementioned, aforesaid} {property, estate}.

Erica 1, Carmen 0

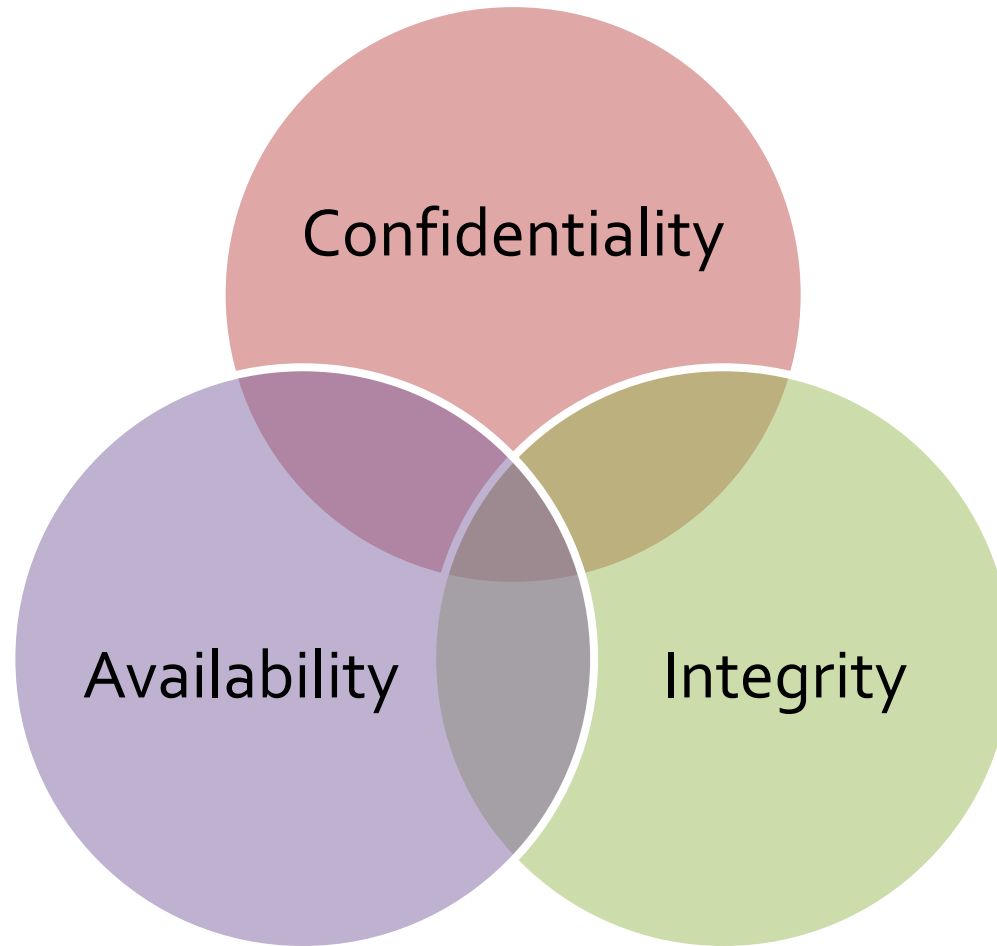
- Having generated the good and bad contracts, Erica can find a good and bad pair with matching hashes with high probability
- Erica sends the good one to Carmen to sign
- Erica keeps the bad one, brings it to court when Carmen says that Erica didn't pay the right amount

The lesson?

- Use hash functions with a long digest
- A hash function with an m -bit digest can produce about 2^m different hashes
- But some attacks only need around $2^{m/2}$ different messages to find a collision
- Don't do real estate deals with Erica

Week 1 Review

The basics of computer security



Confidentiality

- You don't want other people to be able to read your stuff
 - Some of your stuff, anyway
- Cryptography, the art of encoding information so that it is only readable by those knowing a secret (key or password), is a principle tool used here
- Confidentiality is also called **secrecy** or **privacy**

Integrity

- You don't want people to mess up your stuff
- You want to know:
 - That your important data cannot be easily changed
 - That outside data you consider trustworthy cannot be easily changed either
- There are many different ways that data can be messed up, and every application has different priorities

Availability

- You want to be able to use your stuff
- Many attacks are based on **denial of service**, simply stopping a system from functioning correctly
- Availability can mean any of the following:
 - The service is present in usable form
 - There is enough capacity for authorized users
 - The service is making reasonable progress
 - The service completes in an acceptable period of time

Harm

Malicious, human-caused threats often involve one or more of the following kind of harm:

Interception

- Someone read something they weren't supposed to

Interruption

- Something became unavailable or unusable

Modification

- Someone changed something they weren't supposed to

Fabrication

- Someone created fake things

Method, opportunity, motive

- As with traditional crime, a computer attacker must have three things:

Method

- Skills and tools to perform the attack

Opportunity

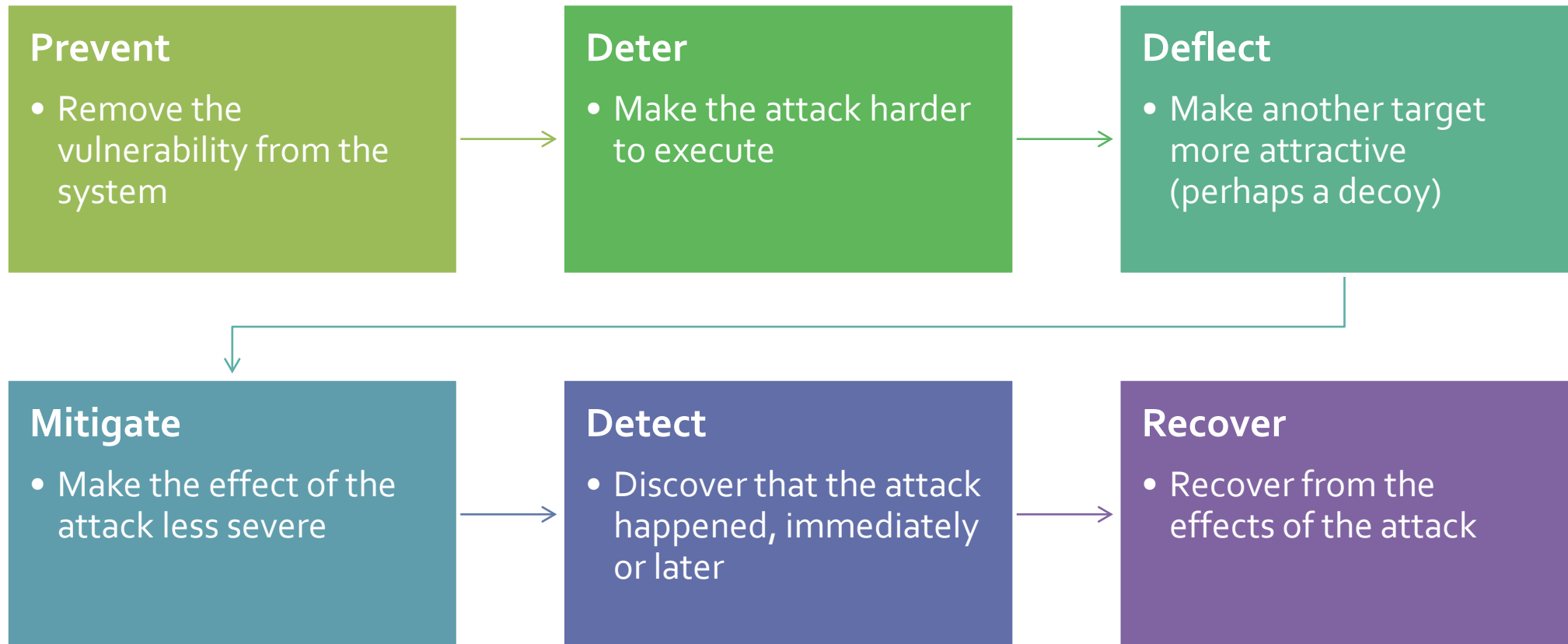
- Time and access to accomplish the attack

Motive

- A reason to perform the attack

Controls

- There are six common ways of controlling attacks, many of which can be used together



Definition of authentication

- **Authentication** is the binding of an identity to a subject
 - Example: Bill Gates (external entity) is a registered user whose identity on this system is **gatesw** (identity of system subject)
- The external identity must provide information to authenticate based on
 1. What the entity knows (passwords)
 2. What the entity has (security badge)
 3. What the entity is (fingerprints or voice ID)

Passwords

- Passwords are one of the most common forms of authentication mechanisms based on what the entity knows
- The password represents **authentication information** that the user must know
- The system keeps **complementation information** that can be used to check the password
- Real systems generally do not store passwords in the clear but store hashes of them
- Unix chooses one of 4,096 different hash functions, hashes the password into an 11-character string, and then prepends 2 characters specifying which hash function was used

Attacking a password system

- A **dictionary attack** is an attack based on guessing the password from trial and error
 - A dictionary attack can work on the complementary information (hashes of passwords)
 - If this information is unavailable, a dictionary attack can directly attack the authentication functions (literally trying to log in repeatedly)
- Let **P** be the probability that an attacker guesses the password over a certain span of time
- Let **G** be the number of guesses that can be made per unit time
- Let **T** be the number of time units of guessing
- Let **N** be the number of possible passwords
- Then,

$$P \geq \frac{TG}{N}$$

Defending authentication functions

- **Backoff**

- Force the user to wait longer and longer between failed authentication techniques
- Exponential backoff means that the first time waits 1 second before allowing a user to log in, the second waits 2 seconds, the third waits 4 seconds, etc.

- **Disconnection**

- If the connection is remote and requires significant time to connect (dialing, VPN, etc.), the system can simply break connection after a number of failed attempts

- **Disabling**

- With n failed attempts, an account is locked until an administrator resets the account

- **Jailing**

- In jailing, the user is allowed to enter a fake system that looks like the real one
- In theory, jailing can be used to learn more about an attacker's goals
- Attractive data (called honeypots) can be made available, tempting the attacker to spend more time on the system (until he can be caught)

Biometrics

- Biometrics means identifying humans by their physical and biological characteristics
- This technology is often seen in spy and science fiction movies
 - It does exist, but it's far from perfect
- Like passwords, the actual biometric scans are usually not stored
 - Instead specific features are stored for later comparison
- Biometrics pose unique privacy concerns because the information collected can reveal health conditions

Problems with biometrics

- People assume that they are more secure than they are
- Attacks:
 - Fingerprints can be lifted off a champagne glass
 - Voices can be recorded
 - Iris recognition can be faked with special contact lenses
- Both false positives and false negatives are possible
- Disabilities can prevent people from using some kinds of biometrics
- It's possible to tamper with transmission from the biometric reader
- Biometric characteristics can change
- Identical twins sometimes pose a problem

Week 2 Review

Tokens

- Tokens are physical objects you possess
 - Keys
 - Badges
 - Cell phones
 - RFIDs
- **Passive tokens** take no action and do not change
 - Example: photo ID
- **Active tokens** change or interact with surroundings
 - Examples: RFID or magnetic card

Access control

- **Subjects** are human users or programs that are executing on their behalf
- **Objects** are things that actions can be performed on
 - Files
 - Database fields
 - Directories
 - Hardware devices
- **Access modes** are the different ways that access can be done: read, write, modify, delete, etc.
- **Access control** is the process of managing the access modes that subjects can have on objects

Access control goals

- Check every access
 - The user may no longer have rights to a resource
 - The user may have gained rights
- Enforce least privilege
 - **Least privilege** means you get the bare minimum to get your job done
- Verify acceptable usage
 - Access to an object is not enough: Some actions might be legal and others illegal

Access control matrix example

Subjects	Objects			
	file 1	file 2	process 1	process 2
process 1	<i>read, write, own</i>	<i>read</i>	<i>read, write, execute, own</i>	<i>write</i>
process 2	<i>append</i>	<i>read, own</i>	<i>read</i>	<i>read, write, execute, own</i>

Cryptography

- "Secret writing"
- The art of encoding a message so that its meaning is hidden
- **Cryptanalysis** is breaking those codes

Encryption and decryption

- **Encryption** is the process of taking a message and encoding it
- **Decryption** is the process of decoding the code back into a message
- A **plaintext** is a message before encryption
- A **ciphertext** is the message in encrypted form
- A **key** is an extra piece of information used in the encryption process

Notation

- A plaintext is M (sometimes P)
- A ciphertext is C
- The encryption function $E(x)$ takes M and converts it into C
 - $E(M) = C$
- The decryption function $D(x)$ takes C and converts it into M
 - $D(C) = M$
- We sometimes specify encryption and decryption functions $E_k(x)$ and $D_k(x)$ specific to a key k

Attacks

- Cryptography is supposed to prevent people from reading certain messages
- Thus, we measure a **cryptosystem** based on its resistance to an **adversary** or **attacker**
- Kinds of attacks:
 - **Ciphertext only:** Attacker only has access to an encrypted message, with a goal of decrypting it
 - **Known plaintext:** Attacker has access to a plaintext and its matching ciphertext, with a goal of discovering the key
 - **Chosen plaintext:** Attacker may ask to encrypt any plaintext, with a goal of discovering the key
 - Others, less common

Terminology remix

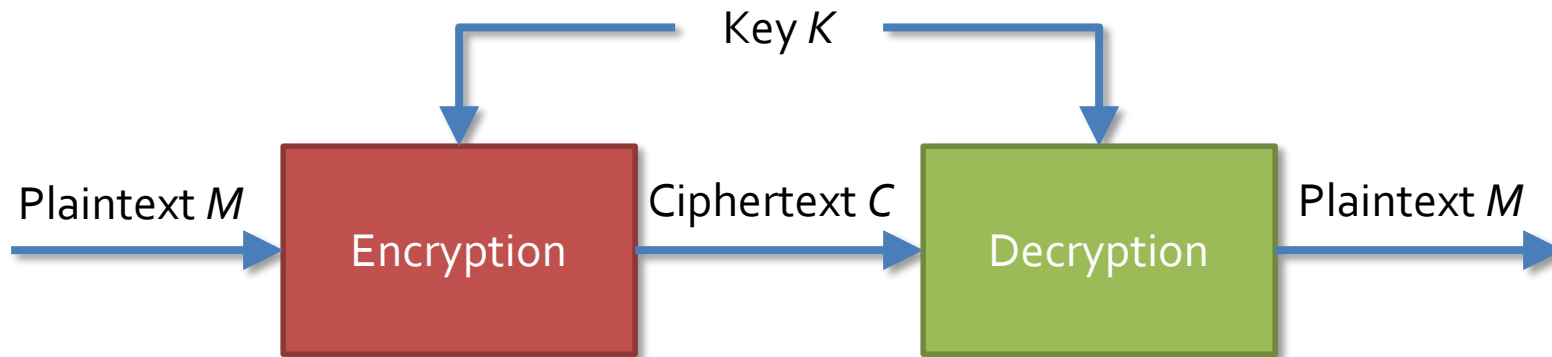
- Rather than use letters, a system popularized by Ron Rivest is to use **Alice** and **Bob** as the two parties communicating
 - **Carl** or another “C” name can be used if three people are involved
- **Trent** is a trusted third party
- **Eve** is used for an evil user who often eavesdrops
- **Mallory** is used for a malicious user who is usually trying to modify messages

Encryption algorithms

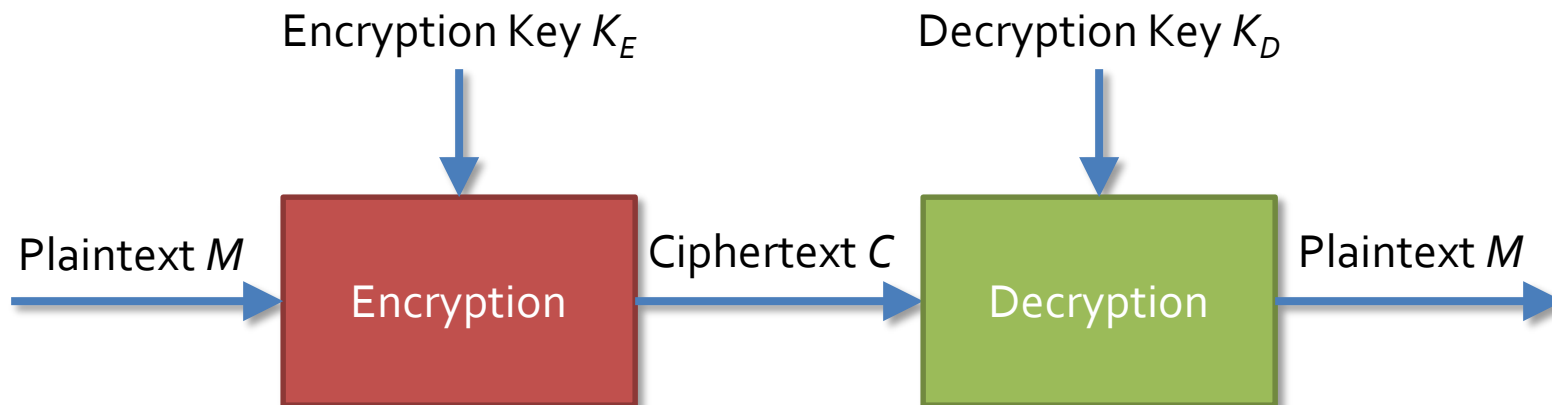
- The algorithms for encryption often rely on a secret piece of information, called a key
- We can notate the use of a specific key in either of the two following ways:
 - $C = E_K(M)$
 - $C = E(K, M)$
- In symmetric (or private key) encryption, the encryption key and the decryption key are the same
- In asymmetric (or public key) encryption, the encryption key and the decryption key are different

Symmetric vs. asymmetric

Symmetric Encryption



Asymmetric Encryption



Cryptanalysis

- There are two kinds of security for encryption schemes
 - **Unconditionally secure**
 - No matter how much time or energy an attacker has, it is impossible to determine the plaintext
 - **Computationally secure**
 - The cost of breaking the cipher exceeds the value of the encrypted information
 - The time required to break the cipher exceeds the useful lifetime of the information
- We focus on computationally secure, because there is only one practical system that is unconditionally secure
- "I want them to remain secret for as long as men are capable of evil" -Avi from Cryptonomicon

Review of Modular Arithmetic

- Modulo operator takes the remainder
- Two numbers are said to be congruent modulo n if they have the same remainder when divided by n
- For example,
 $39 \equiv 3 \pmod{12}$
- Addition, subtraction, and multiplication:
 - $[(a \bmod n) + (b \bmod n)] \bmod n = (a + b) \bmod n$
 - $[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n$
 - $[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$

Divided and Conquered

- We can't actually divide
- Instead, we have to find the multiplicative inverse
- The multiplicative inverse of x exists if and only if x is relatively prime to n
- $13 \cdot 5 \equiv 65 \equiv 1 \pmod{16}$
- So, 13 and 5 are multiplicative inverses mod 16
- But, 0, 2, 4, 6, 8, 10, and 12 do not have multiplicative inverses mod 16

Week 3 Review

Definition

- A shift cipher encrypts a message by shifting all of the letters down in the alphabet
- Using the Latin alphabet, there are 26 (well, 25) possible shift ciphers
- We can model a shift cipher by numbering the letters A, B, C, ... Z as 0, 1, 2, ... 25
- Then, we let the key k be the shift
- For a given letter x :
$$E_k(x) = (x + k) \bmod 26$$

Cryptanalysis of a Shift Cipher

- Cryptanalysis of a shift cipher is incredibly easy
- You just have to try 26 possibilities to be sure you have the right one
- A shift cipher is a simplified version of a **substitution cipher**

Definition

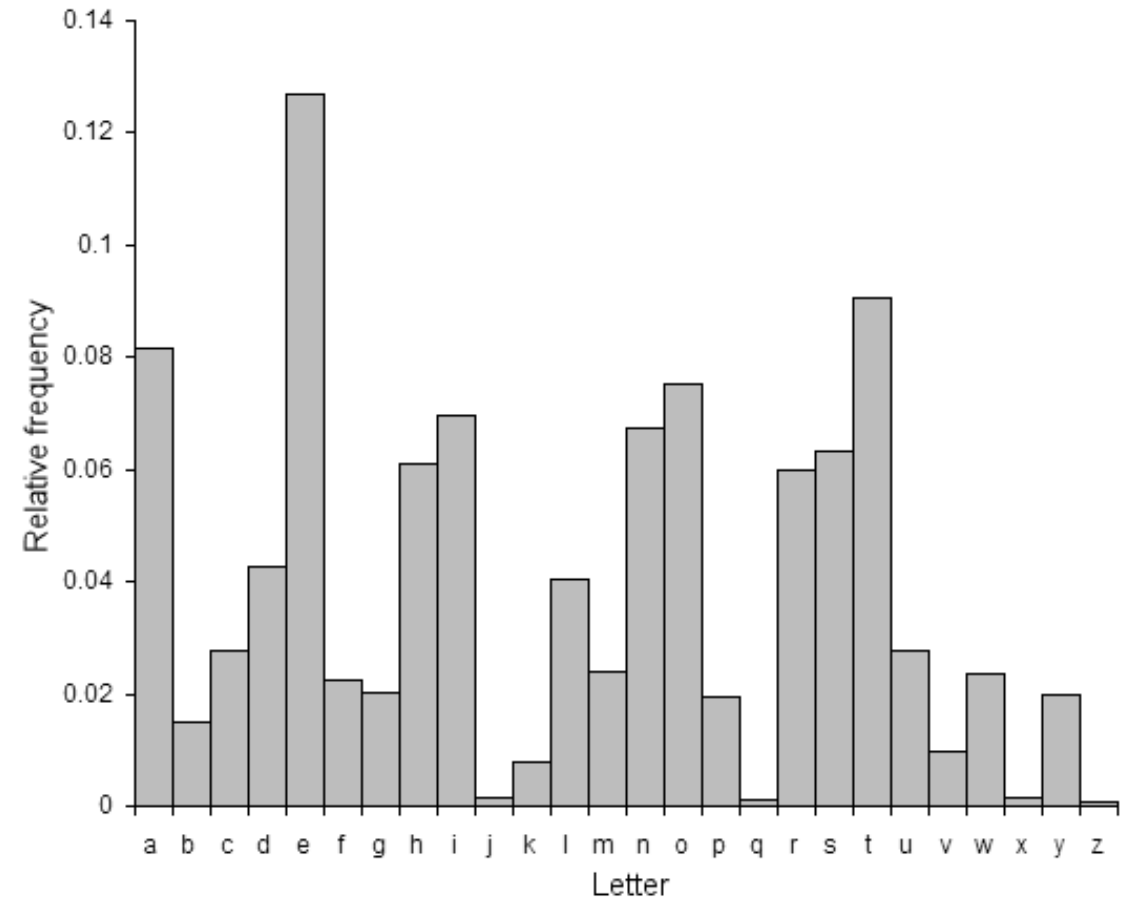
- In a transposition cipher, the letters are reordered but their values are not changed
- Any transposition cipher is a permutation function of some kind

Substitution ciphers

- **Substitution ciphers** cover a wide range of possible ciphers, including the shift cipher
- In a substitution cipher, each element of the plaintext is substituted for some corresponding element of the ciphertext
- **Monoalphabetic** substitution ciphers always use the same substitutions for a letter (or given sequence of letters)
- **Polyalphabetic** substitution ciphers use different substitutions throughout the encryption process

Frequency Attack

- English language defeats us
- Some letters are used more frequently than others:
ETAOINSHRDLU
- Longer texts will behave more consistently
- Make a histogram, break the cipher



Vigenère cipher

- The Vigenère cipher is a form of polyalphabetic substitution cipher
- In this cipher, we take a key word and repeat it, over and over, until it is as long as the message
- Then, we add the repetitions of keywords to our message mod 26

Cryptanalysis of Vigenère

- The index of coincidence measures the differences in the frequencies in the ciphertext
- It is the probability that two randomly chosen letters from the ciphertext are the same

- $$IC = \frac{1}{N(N-1)} \sum_{i=0}^{25} F_i(F_i - 1)$$

Period	1	2	3	4	5	10	Large
Expected IC	0.066	0.052	0.047	0.045	0.044	0.041	0.038

Normalized index of coincidence

- Some systems look at a "normalized" index of coincidence, which is found by multiplying the formula given on the previous page by the number of letters in the language
 - 26 for English
 - When reading the literature, both normalized and unnormalized versions can be called index of coincidence
- Here are index of coincidence values for a few common languages

Language	Index
English	1.73
French	2.02
German	2.05
Italian	1.94
Portuguese	1.94
Russian	1.76
Spanish	1.94

After the length is known...

- The rest is easy
- Try various shifts for each letter of the key so that high frequency letters (E, T, A) occur with high frequency and low frequency letters (Q, X, Z) occur with low frequency
- Guess and check

Perfect secrecy

- A One-Time Pad has the property of **perfect secrecy** or **Shannon secrecy**
- Perfect secrecy means that $P(M) = P(M|C)$
 - Remember that it is possible to find a key that would decrypt a ciphertext to **any** plaintext
- Thus, learning the ciphertext tells you **nothing** about the plaintext

One-Time Pad weaknesses

- You can only use it one time
 - Otherwise, recovering the key is trivial
 - Completely vulnerable to known plaintext attack
- The key is as long as the message
- If you have a way of sending a key that long securely, why not send the message the same way?
- Generating keys with appropriate levels of randomness presents a problem

Stream and block ciphers

- A common way of dividing ciphers is into **stream ciphers** and **block ciphers**
- Block ciphers divide messages into fixed length parts (or blocks) and encipher each part with the same key
- Stream ciphers encipher each message character by character
 - Some other authors define a stream cipher to be like a block cipher except that the key changes with each block based on the message

Confusion and Diffusion

- Confusion is the property of a cryptosystem that changing a single character in the plaintext should not have a predictable effect
- Diffusion is the property of a cryptosystem that each character in the plaintext should impact many characters in the ciphertext
- Examples:
 - Caesar cipher has poor confusion and no diffusion
 - One time pad has good confusion but no diffusion
 - Auto-key ciphers may have poor confusion but good diffusion
 - AES and DES have good confusion and diffusion within a block

DES

- **Data Encryption Standard**
- DES is a typical block cipher
- It was chosen as the government's standard for encryption in 1976 (but has since been deprecated)
- DES works on blocks 64 bits in size
- DES uses a 56 bit key
- NSA helped design it... amidst some controversy

DES strengths

- DES is fast
- Easy to implement in software or hardware
- Encryption is the same as decryption
- Triple DES is still standard for many financial applications
- Resistant to differential and linear cryptanalysis (2^{47} and 2^{43} known pairs required, respectively)

DES weaknesses

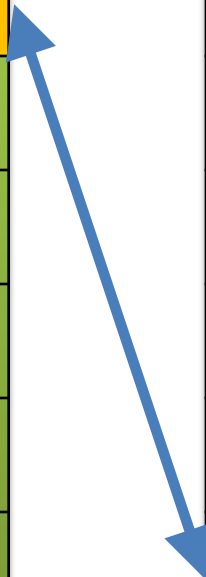
- Short key size
 - Brute force attack by EFF in 1998 in 56 hours then in 1999 in just over 22 hours
 - Brute force attack by University of Bochum and Kiel in 9 days in 2006 (but, using a machine costing only \$10,000)
- If you could check 1,000,000,000 keys per second (which is unlikely with a commodity PC), it would take an average of 417 days to recover a key

Double DES

- "DES is wrong if you listen to NIST, Double DES ain't no better, man, that got dissed"
--MC Plus+
- Double DES encrypts a plaintext with DES twice, using two different keys
- Double DES is susceptible to a **meet-in-the-middle attack**
- This attack uses a space-time tradeoff
- Although two keys should mean $56 + 56 = 112$ bits of security or 2^{112} time for a brute force attack, the meet-in-the-middle attack can run in roughly 2^{57} or 2^{58} time, using 2^{56} space

Double DES attack

Encrypt P_1		Decrypt C_1	
K_1	492989976	864059530	K_1
K_2	688857766	717075649	K_2
K_3	282627672	993328605	K_3
K_4	499659602	991061777	K_4
K_5	532263602	154785500	K_5
K_6	498278096	210537840	K_6
K_7	752271744	688857766	K_7
K_8	846172716	528110960	K_8



- Two pairs of plaintexts and ciphertexts are needed
- Encrypt P_1 with all possible keys and save them
- Decrypt C_1 with all possible keys
 - If the result matches anything in the list, use the key to encrypt P_2
 - If that matches C_2 , you win!
- On the left, I show all the decryptions, but only the encryptions need to be stored

Triple DES

- Although susceptible to a brute force attack, DES has no other major weaknesses
 - Double DES can be defeated by an extension of the brute force attack
 - What about triple DES?
- Let $E_K(X)$ and $D_K(X)$ be encryption and decryption using DES with key K
- Triple DES uses keys K_1 , K_2 , and K_3
 - $C = E_{K_1}(D_{K_2}(E_{K_3}(M)))$
 - Setting $K_1 = K_2 = K_3$ allows for compatibility with single DES systems
- Triple DES is still a standard for financial transactions with no known practical attacks

Week 4 Review

AES

- **A**dvanced **E**ncryption **S**tandard
- Block cipher designed to replace DES
- Block size of 128-bits
- Key sizes of 128, 192, and 256 bits
- Like DES, has a number of rounds (10, 12, or 14 depending on key size)
- Originally called Rijndael, after its Belgian inventors
- Competed with 14 other algorithms over a 5 year period before being selected by NIST

AES pros and cons

- Strengths

- Strong key size
- Fast in hardware and software
- Rich algebraic structure
- Well-studied, open standard

- Weaknesses

- Almost none
- A few theoretical attacks exist on reduced round numbers of AES
- No practical attacks other than side channel attacks

Public key cryptography

- Sometimes, we need something different
- We want a **public key** that anyone can use to encrypt a message to Alice
- Alice has a **private key** that can decrypt such a message
- The **public key** can only encrypt messages; it cannot be used to decrypt messages

RSA Algorithm

- Named for **R**ivest, **S**hamir, and **A**dleman
- Take a plaintext ***M*** converted to an integer
- Create an ciphertext ***C*** as follows:
$$C = M^e \bmod n$$
- Decrypt ***C*** back into ***M*** as follows:
$$M = C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$$

The pieces

Term	Details	Source
M	Message to be encrypted	Sender
C	Encrypted message	Computed by sender
n	Modulus, $n = pq$	Known by everyone
p	Prime number	Known by receiver
q	Prime number	Known by receiver
e	Encryption exponent	Known by everyone
d	Decryption exponent	Computed by receiver
$\phi(n)$	Totient of n	Known by receiver

How it Works

- To encrypt:
 $C = M^e \bmod n$
- e could be 3 and is often 65537, but is always publically known
- To decrypt:
 $M = C^d \bmod n = M^{ed} \bmod n$
- We get d by finding the multiplicative inverse of $e \bmod \phi(n)$
- So, $ed \equiv 1 \pmod{\phi(n)}$

Why it Works

- We know that $ed \equiv 1 \pmod{\phi(n)}$
- This means that $ed = k\phi(n) + 1$ for some nonnegative integer k
- $M^{ed} = M^{k\phi(n) + 1} \equiv M \cdot (M^{\phi(n)})^k \pmod{n}$
- By Euler's Theorem
 $M^{\phi(n)} \equiv 1 \pmod{n}$
- So, $M \cdot (M^{\phi(n)})^k \equiv M \pmod{n}$

Notation for sending

- We will refer to several schemes for sending data
- Let X and Y be parties and Z be a message
- $\{Z\}_k$ means message Z encrypted with key k
- Thus, our standard notation will be:
 - $X \rightarrow Y: \{Z\}_k$
 - Which means, X sends message Z , encrypted with key k , to Y
- X and Y will be participants like Alice and Bob and k will be a clearly labeled key
- $A || B$ means concatenate message A with B

Kinds of keys

- Typical to key exchanges is the idea of interchange keys and session keys
- An **interchange key** is a key associated with a particular user over a (long) period of time
- A **session key** is a key used for a particular set of communication events
- Why have both kinds of keys?

Key exchange criteria

- To be secure, a key exchange whose goal is to allow secret communication from Alice to Bob must meet this criteria:
 1. Alice and Bob cannot transmit their key unencrypted
 2. Alice and Bob may decide to trust a third party (Cathy or Trent)
 3. Cryptosystems and protocols must be public, only the keys are secret

Classical exchange: Attempt 0

- If Bob and Alice have no prior arrangements, classical cryptosystems require a trusted third party Trent
- Trent and Alice share a secret key k_{Alice} and Trent and Bob share a secret key k_{Bob}
- Here is the protocol:
 1. Alice \rightarrow Trent: {request session key to Bob} k_{Alice}
 2. Trent \rightarrow Alice: { $k_{session}$ } k_{Alice} || { $k_{session}$ } k_{Bob}
 3. Alice \rightarrow Bob: { $k_{session}$ } k_{Bob}

What's the problem?

- Unfortunately, this protocol is vulnerable to a replay attack
- (Evil user) Eve records $\{ k_{session} \} k_{Bob}$ sent in step 3 and also some message enciphered with $k_{session}$ (such as "Deposit \$500 in Dan's bank account")
- Eve can send the session key to Bob and then send the replayed message
- Maybe Eve is in cahoots with Dan to get him paid twice
- Eve may or may not know the contents of the message she is sending
- The real problem is no **authentication**

Public key exchange

- Suddenly, the sun comes out!
- Public key exchanges should be really easy
- The basic outline is:
 1. Alice \rightarrow Bob: $\{ k_{session} \} e_{Bob}$
- e_{Bob} is Bob's public key
- Only Bob can read it, everything's perfect!
- Except ...
- There is still no authentication

Easily fixable

- Alice only needs to encrypt the session key with her private key
- That way, Bob will be able to decrypt it with her public key when it arrives
- New protocol:
 1. Alice \rightarrow Bob: $\{\{ k_{\text{session}} \} d_{\text{Alice}} \} e_{\text{Bob}}$
- Man in the middle attacks are still possible if Alice gets the wrong public key for Bob

Week 5 Review

Catch-22

- Your computer needs to be able read the password file to check passwords
- But, even an administrator shouldn't be able to read everyone's passwords
- Hash functions to the rescue!

Definition

- A **cryptographic** (or one-way) hash function (called a cryptographic checksum in the book) takes a variable sized message M and produces a fixed-size hash code $H(M)$
- **Not the same as hash functions from data structures**
- The hash code produced is also called a **digest**
- It can be used to provide authentication of both the integrity and the sender of a message
- It allows us to store some information about a message that an attacker cannot use to recover the message

Crucial properties

Preimage Resistance

- Given a digest, should be hard to find a message that would produce it
- One-way property

Second Preimage Resistance

- Given a message m , it should be hard to find a different message that has the same digest

Collision Resistance

- Should be hard to find any two messages that hash to the same digest (collision)

Additional properties

Avalanching

- A small change in input should correspond to a large change in output

Applicability

- Hash function should work on a block of data of any size

Uniformity

- Output should be a fixed length

Speed

- It should be fast to compute a digest in software and hardware
- No longer than retrieval from secondary storage

Salt

- If you are the administrator of a large system, you might notice that two people have the same password hash
- With people's password habits, the odds are very high that their passwords are the same
- To add to the semantic security of such schemes extra data called **salt** is added to the end of a password
- The salt is usually based on the time the account was created or the account name

MD5

- Message Digest Algorithm 5
- Very popular hashing algorithm
- Designed by Ron Rivest (of RSA fame)
- **Digest size:** 128 bits
- **Security**
 - Completely broken
 - Reasonable size attacks (2^{32}) exist to create two messages with the same hash value
- MD5 hashes are still commonly used to check to see if a download finished without error

SHA family

- **Secure Hash Algorithm**
- Created by NIST
- SHA-0 was published in 1993, but it was replaced in 1995 by SHA-1
- The difference between the two is only a single bitwise rotation, but the NSA said it was important
- Digest size: 160 bits
- Security
 - Broken if you have the resources
 - Theoretical attacks running in 2^{51} - 2^{57} time exist
 - Google generated two PDF files with the same hash in just over 2^{63} hashes in 2017
- SHA-2 is a successor family of hash functions
 - 224, 256, 384, 512 bit digests
 - Much better security
 - Designed by the NSA

Keccak (SHA-3)

- Keccak uses a completely different form of hashing than SHA-0, SHA-1, and SHA-2
- Although there are only theoretical attacks on SHA-1 and no real attacks on SHA-2, the attacks on SHA-0 made people nervous about hash functions following the same design
- Keccak also allows for variable size digests, for added security
 - 224, 256, 384, and 512 are standard for SHA-3, but it is possible to go arbitrarily high in Keccak

General case

- If we care about a group of k items which can have a value between 1 and n , the probability that two are the same is:

$$P(n, k) = 1 - \frac{n!}{(n - k)! n^k}$$

- Because this form is a little unwieldy, we have an approximation that is easier to punch into a calculator:

$$P(n, k) > 1 - e^{\frac{-k(k-1)}{2n}}$$

Count it up

- If we want to find the number of items needed before there is greater than a $\frac{1}{2}$ probability of collision we get:

$$\begin{aligned}\frac{1}{2} &= 1 - e^{\frac{-k(k-1)}{2n}} \\ -\frac{1}{2} &= -e^{\frac{-k(k-1)}{2n}} \\ 2 &= e^{\frac{k(k-1)}{2n}} \\ \ln(2) &= \frac{k(k-1)}{2n}\end{aligned}$$

- For large k , $k(k-1) \approx k^2$, giving:

$$k \approx \sqrt{2 \ln(2) n} \approx 1.18\sqrt{n}$$

Signing scheme example

- Digital signature schemes typically employ public key cryptography
- We need the one way property so that we can verify that it works without being able to break it
- **Full Domain Hash** uses RSA to do this:
 - Given message M , we find $H(M)$, then raise $H(M)$ to the secret decryption exponent to find signature S
 - $S = H(M)^d \bmod n$
 - To verify the signature, take the signature and raise it to the publicly known encryption exponent e and compare that to the hash of the message
- If $S^e \bmod n = H(M)$, we feel reasonably sure of two things:
 - S is a signature for a M (M has not been changed)
 - Only someone who knows the private key for n could have signed it
- Why does it use the hash of the message instead of the message?

The lesson?

- Use hash functions with a long digest
- A hash function with an m -bit digest can produce about 2^m different hashes
- But, some attacks only need around $2^{m/2}$ different messages to find a collision

Upcoming

Next time...

- Exam 1 on Monday

Reminders

- **Office hours from 1:45-4 p.m. canceled today**
- Review chapters 1, 2, and 12 for Exam 1
- Finish Assignment 2
 - **Due tonight by midnight!**
- Start on Project 2