CRYPTOGRAPHY

Chapters 2, 20, 21

WHAT IS CRYPTOGRAPHY?

Cryptography is the study of techniques for secure communication in the presence of third parties, which are referred to as adversaries

Modern cryptography is not necessary synonymous with *encryption*, which is the technique of securing a message against adversaries

A *cipher* (or cypher) is a pair of algorithms that can create the encryption and the reversing decryption

ENCRYPTION TECHNIQUES

Regardless of how advanced many of the encryption techniques seem, all of them ultimately boil down to the use of two different methods:

- Substitution
  - Swapping one letter for another
- Transposition
  - Rearranging the relative position of one letter for another

Encryption Techniques (2)

We can also categorize encryption techniques by several different means:

- The number of keys used
  - Symmetric
    - Sender and receiver use the same key
  - Asymmetric
    - Sender and receiver use different keys
- The way plaintext is processed
  - Block Ciphers
    - Processes input one block of elements at a time
  - Stream Ciphers
    - Continuously process elements as they arrive

CRYPTOGRAPHICAL ERAS

In the computer age, we typically divide up cryptographic techniques into two main eras:

- Classical cryptographical era
  - All methods of analog encryption, and mostly used in the pre-WWII historical eras
- Modern cryptographical era
  - Virtually all forms of digital encryption since WWII, and encompasses the majority of methods we will discuss in this course

HISTORY OF CRYPTOGRAPHY

Interestingly, cryptography has a long and rich history:

- The Caesar Cipher

One of the oldest forms of encryption, rumored to be used by Julius Caesar himself; he would encrypt messages to his legion commanders by employing a simple letter shift of three letters across the alphabet set. Internet forums also commonly use a form of Caesar cypher to hide “spoilers” in the form of ROT13 (meaning, 13 rotations instead of 3)*.

In[ ]:=
```
az = CharacterRange["a", "z"];
decode = ReplaceAll@Dispatch@Thread[az -> RotateRight[az,3]];
encode = ReplaceAll@AssociationThread[az, RotateLeft[az,3]]
StringJoin@encode@Characters["attack at noon"]
StringJoin@decode@Characters[%]
```

*There’s also an old joke on the internet that states the best form of encryption for all of the most sensitive documents is ROT26...

History of Cryptography (2)

- Substitution Cipher
  - While technically the Caesar cipher is a form of substitution cipher, cryptographers typically reserve the phrase for slightly more advanced forms of crypto. Substitution ciphers involve replacing letters with differing letters.

In[•]:=
```
XXXX
```

Out[•]= XXXX

History of Cryptography (3)

- Book Cipher

History of Cryptography (4)

- Case Study: The Enigma Machine

SYMMETRIC ENCRYPTION

- The universal technique for providing confidentiality for transmitted or stored data.
- Also referred to as *conventional encryption* or *single-key* encryption
- Requirements for secure use:
  - Need a strong encryption algorithm
  - Sender and receiver must have copies of the same key and that key
  - The key must remain private and secure from other parties

Symmetric Encryption (2)

Symmetric Encryption (3)

Symmetric encryption is not invulnerable by a long shot. There are two main methods of attacking any form of symmetric encryption:

- Cryptanalytic attacks
  - Rely on some knowledge of the underlying algorithm
  - Can also use the encrypted text against itself in an attack by looking for patterns
- Brute-force attacks
  - Attacker attempts to try all possible keys on the ciphertext until a pattern emerges
    - On average, half of all keys must be tried (referred to as the “half life” of the algorithm)

Symmetric Encryption (3)

Symmetric encryption can further be attacked by making use of the ciphertext itself. *Every* message has inherent patterns in it, no matter how complex the encryption, and we can derive some information about the message itself without ever having to crack it.

- Ciphertext only

  If a ball fell from space with a message in an alien language on it, we would have no idea how to translate the contents of that message. Rarely will we ever encounter such a thing on Earth. This is an example of only knowing the ciphertext.

- Known plaintext

  More often, we will have *some* idea about the message we are attempting to decrypt, even if it is only the language the plaintext message will be in. We also might know the rough type of encryption that was used to secure that message.

- Chosen plaintext

  Maybe we are fortunate to have a greater idea about what the message plaintext *should* say. Is it a secret message to a foreign agent? Is it a shopping list? We can use that knowledge to further analyze the message.

- Chosen ciphertext

  We might also have a bit of insight as to how the message was encrypted. Was the message encrypted using DES? AES? What kinds of computing machines were used? We can also use that knowledge for our cracking benefit.

- Chosen text

  Finally, we may have some understanding about what things should be contained within the message. If it's a message from a secret agent, it's highly unlikely that it is his Amazon wish list, and more likely that the message contains something about international secrets. Even better if we know the general gist of the message.

WHAT MAKES COMPUTATIONALLY-SECURE ENCRYPTION?

We can never have truly secure encryption. Any encryption that we deploy will always have faults and insecurities. So, how do we know if the encryption we are using is secure *enough*?

- Encryption is computationally secure if:
  - Cost of breaking cipher exceeds value of information
  - Time required to break cipher exceeds the useful lifetime of the information
- Usually very difficult to estimate the amount of effort required to break
- Helps to estimate time and cost of a brute-force attack

SYMMETRIC ENCRYPTION COMPARED

	DES	3 DES	AES
Plaintext block size	64	64	128
Ciphertext block size	64	64	128
Key size	56	112 or 168	128 – 512 and

DES = Data Encryption Standard

AES = Advanced Encryption Standard

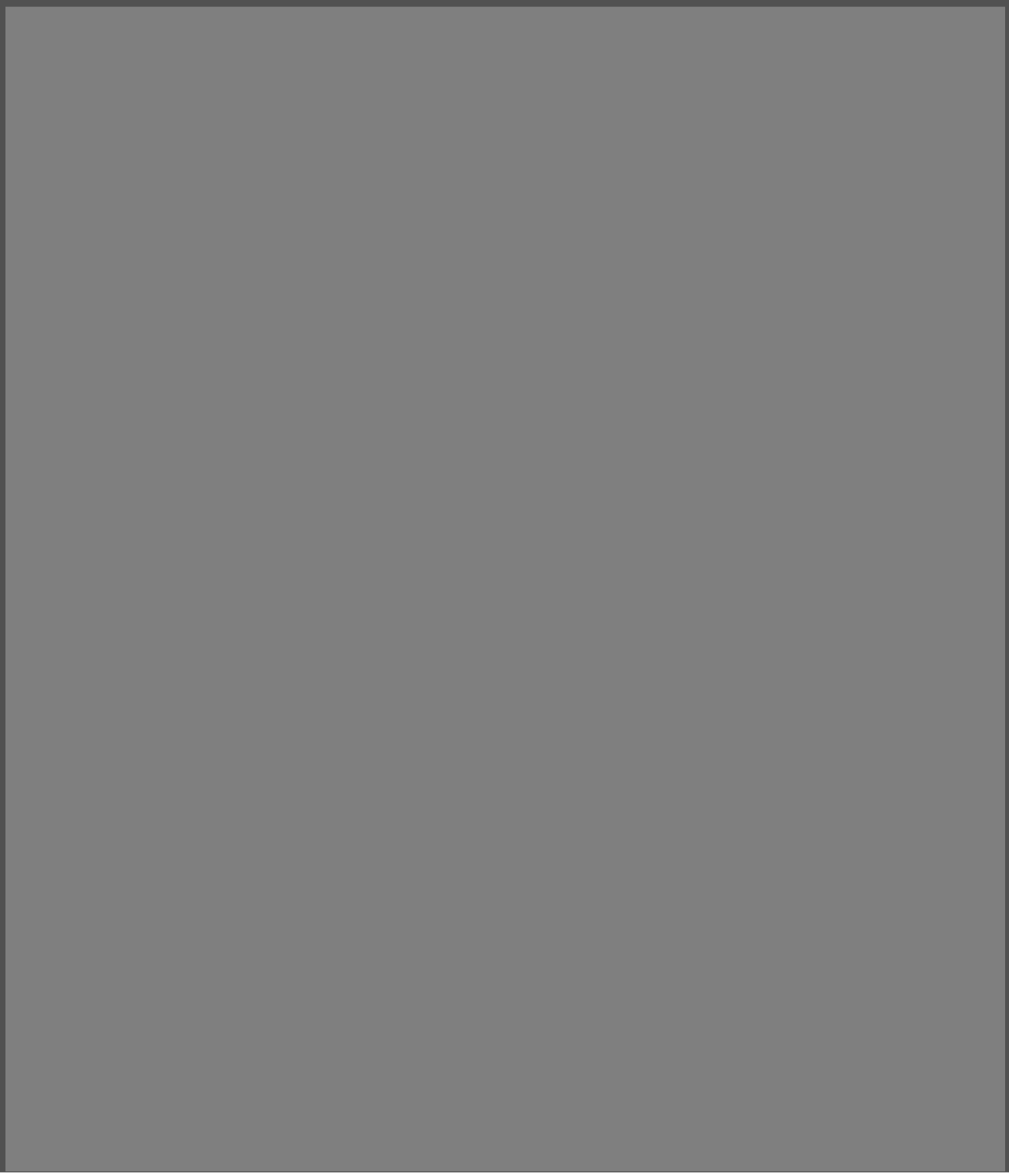
* All sizes are in bits

Symmetric Encryption Compared (2)

Key Size (bits)	Cipher	Number of Keys	T (10^9) dec / sec	T (1
56	DES	$2^{56} \approx 7.2 \times 10^{16}$	2^{55} ns = 1.125 yrs	
128	AES	$2^{128} \approx 3.4 \times 10^{38}$	2^{127} ns = 5.3×10^{21} yrs	5
168	3 DES	$2^{168} \approx 3.7 \times 10^{50}$	2^{167} ns = 5.8×10^{33} yrs	5
192	AES	$2^{192} \approx 6.3 \times 10^{57}$	2^{191} ns = 9.8×10^{40} yrs	9
256	AES	$2^{256} \approx 1.2 \times 10^{77}$	2^{255} ns = 1.8×10^{60} yrs	1

THOUGHT EXPERIMENT: LIFE EXPECTANCY OF THE SUN

In[ ]:= **life expectancy of the sun**



Consider: Our weakest form of AES encryption has a half life of 5.3×10^{17} years with a modern supercomputer. If the sun will only last roughly 1×10^{10} years, why do we need encryption that lasts longer than that? Discuss.

TYPES OF SYMMETRIC ENCRYPTION

DATA ENCRYPTION STANDARD (DES)

The original encryption standard. Today, it is severely outdated.

- Adopted in 1977 by National Bureau of Standards (Now NIST--the same people who bring you the atomic clock time every day!)
- DES is a (very) minor variation of the cipher created by Horst Feistel in the early 1970s.

Data Encryption Standard (2)

TRIPLE DES (3DES)

Since the late 1980s, the faults and insecurity of DES have been well-known. Unfortunately, at that time, a suitable replacement for DES was years away. A stop-gap measure was created to marginally secure DES so that it could continue to be used without significant cost to the implementers. That stop-gap was 3DES.

- Properties of 3DES:
  - Exactly the same underlying algorithm as DES
  - Performs the algorithm three separate times
  - Is nearly three times as a slow as the original DES algorithm with only a slight improvement of encryption strength.

Triple DES (2)

ADVANCED ENCRYPTION STANDARD (AES)

- Say something about the EFF

Advanced Encryption Standard (2)

Advanced Encryption Standard (3)

Advanced Encryption Standard (4)

Advanced Encryption Standard (5)

PRACTICAL SECURITY ISSUES

- Typically symmetric encryption is applied to a unit of data larger than a single 64-bit or 128-bit block
- Electronic codebook (ECB) mode is the simplest approach to multiple-block encryption
  - Each block of plaintext is encrypted using the same key
  - Cryptanalysts may be able to exploit regularities in the plaintext
- Modes of operation
  - Alternative techniques developed to increase the security of symmetric block encryption for large sequences
  - Overcomes the weaknesses of ECB

BLOCK AND STREAM CIPHERS

- Block Ciphers
  - Processes the input one block of elements at a time
  - Produces an output block for each input block
  - Can reuse keys
  - More common
- Stream Ciphers
  - Processes the input elements continuously
  - Produces output one element at a time
  - Primary advantage is that they are almost always faster and use far less code
  - Encrypts plaintext one byte at a time
  - Pseudorandom stream is one that is unpredictable without knowledge of the input key

Block and Stream Ciphers (2)

Cipher	Key Length	Speed (Mbps) *
DES	56	21
3 DES	168	10
AES	128	61
RC4	Variable	113

*Speed compared on an Intel Pentium 4

Block and Stream Ciphers (3)

BLOCK CIPHER MODES OF OPERATION

SYMMETRIC CIPHERS' BIGGEST ISSUE

We still haven't discussed the biggest issue regarding symmetric ciphers: how do we exchange the key?

MESSAGE AUTHENTICATION

- Protects against active attacks
- Verifies that messages are authentic and not altered
- Can be used with conventional encryption

Message Authentication (2)

CRYPTOGRAPHIC HASH FUNCTIONS

Cryptographic Hash Function (2)

HASH FUNCTION REQUIREMENTS

- Can be applied to block data of any size
- Produces a fixed-length output
- H(x) is relatively easy to compute for any given x
- One way or pre-image resistant
  - Computationally infeasible to find x such that H(x) = h
- Computationally infeasible to find $y \neq x$ such that H(y)=H(x)
- Collision resistant or strong collision resistance
  - Computationally infeasible to find any pair (x,y) such that H(x)=H(y)

SIMPLE ONE-WAY HASH FUNCTION

SIMPLE HASH ALGORITHM (SHA)

- Originally developed by NIST (the time people!) and published in 1993
- Quickly revised in 1995 as SHA-1 to introduce stronger hashing values (160-bit)
- Even further revised in 2002:
  - Adds 3 additional versions of SHA
  - SHA-256, SHA-384, SHA-512
  - With 256/384/512-bit hash values
  - Same basic structure as SHA-1 but greater security
- Older versions phased out by 2010 (but are still seen in use even today)

Simple Hash Algorithm (2)

Simple Hash Algorithm (3)

Simple Hash Algorithm (4)

HASH-BASED MESSAGE AUTHENTICATION CODE (HMAC)

- Interest in developing a MAC derived from a cryptographic hash code
  - Cryptographic hash functions generally execute faster
  - Library code is widely available
  - SHA-1 was not deigned for use as a MAC because it does not rely on a secret key
- Has been chosen as the mandatory-to-implement MAC for IP security
  - Used in other Internet protocols such as Transport Layer Security (TLS) and Secure Electronic Transaction (SET)

HMAC (2)

HMAC (3)

- Security depends on the cryptographic strength of the underlying hash function
- For a given level of effort on messages generated by a legitimate user and seen by the attacker, the probability of successful attack on HMAC is equivalent to one of the following attacks on the embedded hash function:
  - Either attacker computes output even with random secret IV
    - Brute force key $O(2^n)$ or use birthday attack
  - Or attacker finds collisions in hash function even when IV is random and secret
    - ie. find M and M' such that H(M)=H(M')
    - Birthday attack $O(\frac{2^n}{2})$
    - MD5 secure in HMAC since only observe

PUBLIC-KEY ENCRYPTION

SOMETIMES CALLED PUBLIC/PRIVATE-KEY ENCRYPTION, BUT *NEVER* PRIVATE-KEY ENCRYPTION

- Introduced by Whitfield Diffie and Martin Hellman in 1976
- Based on purely mathematical functions
- Distinguishing Property: Asymmetry
  - Uses two separate keys: public and private key pair
  - Public key is made public for anyone to see

Public-Key Encryption (2)

PUBLIC-KEY ENCRYPTION COMPARED

Algorithm	Digital Signature	Symm Key	Key Encrypt
RSA	Yes	Yes	Yes
Diffie – Hellman	No	Yes	No
DSS	Yes	No	No
Eliptic Curve	Yes	Yes	Yes

REQUIREMENTS FOR PUBLIC-KEY CRYPTOSYSTEMS

- Computationally easy to make keys
- Computationally easy for sender knowing public key to encrypt messages
- Computationally easy for receiver knowing private key to decrypt ciphertext
- Computationally infeasible for opponent to determine private key from public key
- Computationally infeasible for opponent to otherwise recover original message
- Useful if either key can be used for each role

RSA PUBLIC-KEY ENCRYPTION

- By Rivest, Shamir & Adleman of MIT in 1977
- Best known and widely used public-key algorithm
- Uses exponentiation of integers modulo a prime
  - Encrypt: C = M^e mod n
  - Decrypt: M = C^d mod n = $(M^e)^d$ mod n = M
- Both sender and receiver know values of n and e
- Only receiver knows value of d
- Public-key encryption algorithm with public key PU = {e, n} and private key PR = {d, n}

RSA Public-Key Encryption (2)

RSA Public-Key Encryption (3)

```
originaltext = "hello world"

(* Alice computes the following *)
p = 857 (* Alice first secret prime *);
q = 433  (* Alice second secret prime *);
n = p*q (* Alice public key - sends to Bob *);
e = 65537 (* encrypting exponent-RSA standard *);
ep = (p-1)*(q-1) (* EulerPhi *);

(* Encryption Process - Bob encrypts to Alice *)
M = LetterNumber[originaltext] (* convert string to numbers *);
cipher = PowerMod[M,e,n] (* M^e mod (p*q) *);
PowerMod[M,e,n]  //"Ciphertext: "

(* Decryption Process - Alice decrypts *)
d = Min[ExtendedGCD[e,ep]] (* Alice private key *);
de = Mod[d,ep];
decrypttext = PowerMod[cipher,de,n];
FromLetterNumber[decrypttext] // "Decrypted text: "
```

DIFFIE-HELLMAN KEY EXCHANGE

- First published public-key algorithm
- By Diffie and Hellman in 1976 along with the exposition of public key concepts
- Used in a number of commercial products
- Practical method to exchange a secret key securely that can then be used for subsequent encryption of messages
- Security relies on difficulty of computing discrete logarithms

Diffie-Hellman Key Exchange (2)

Diffie-Hellman Key Exchange (3)

Diffie-Hellman Key Exchange (4)

```
q = 353 (* shared prime number *);
pr = 3 (* shared primitive root *);

(* Alice and Bob create their keys *)
p1 = 97 (* Alice's secret prime *);
p2 = 233 (* Bob's secret prime *);
ya = Mod[Power[pr,p1],q] (* Alice's public key *);
yb = Mod[Power[pr,p2],q] (* Bob's public key *);

(* Alice and Bob exchange public keys *)
ka = Mod[Power[yb,p1],q] (* Alice's private key *);
kb = Mod[Power[ya,p2],q] (* Bob's private key *);

(* Display our results *)
Print["Alice's public key: ", ya]
Print["Bob's public key: ", yb]
Print["Alice's private key: ", ka]
Print["Bob's private key: ", kb
```

MAN-IN-THE-MIDDLE ATTACK

There is one type of attack that we have not discussed yet, but virtually all types of encryption are susceptible to this type of attack. That attack is the man-in-the-middle attack.

- How the attack is carried out:
  - Darth (Vader) generates private keys $\mathcal{X}_{D1}$ and $\mathcal{X}_{D2}$, and their public keys $\mathcal{Y}_{D1}$ and $\mathcal{Y}_{D2}$
  - Alice transmits $\mathcal{Y}_A$ to Bob
  - Darth intercepts $\mathcal{Y}_A$ and transmits $\mathcal{Y}_{D1}$ to Bob. Darth also calculates $\mathcal{K}_2$
  - Bob receives $\mathcal{Y}_{D1}$ and calculates $\mathcal{K}_1$
  - Bob transmits $\mathcal{X}_A$ to Alice
  - Darth intercepts $\mathcal{X}_A$ and transmits $\mathcal{Y}_{D2}$ to Alice. Darth calculates $\mathcal{K}_1$
  - Alice receives $\mathcal{Y}_{D2}$ and calculates $\mathcal{K}_2$
- All subsequent communications are now compromised

DIGITAL SIGNATURES

- Used for authenticating both source and data integrity
- Created by encrypting hash code with private key
- Does not provide confidentiality
  - Even in the case of complete encryption
  - Message is safe from alteration but not eavesdropping

DIGITAL ENVELOPES

- Protects a message without needing to first arrange for sender and receiver to have the same secret key
- Equates to the same thing as a sealed envelope containing an unsigned letter

RANDOM NUMBERS

Having strongly random numbers is extremely important for having good encryption, but this is not something that computers can do well. Instead, we are forced to have *pseudorandom* numbers

- Criteria:
  - Uniform Distribution
    - Frequency of individual number's occurrence should be approximately the same
  - Independence
    - No one value from the sequence can be inferred from another

Random Numbers (2)

- Pseudorandom numbers should be *unpredictable*
  - Each number is statistically independent of other numbers in the sequence
  - Opponent should not be able to predict future elements of the sequence on the basis of earlier elements
- Predictability is the enemy of good encryption

Random Numbers (3)

Why can computers only create pseudorandom numbers?