Cristina Nita-Rotaru

CS6740: Network security

Additional material: Cryptography

1: Terminology and classic ciphers

Readings for this lecture

Required readings:

- Cryptography on Wikipedia

Interesting reading

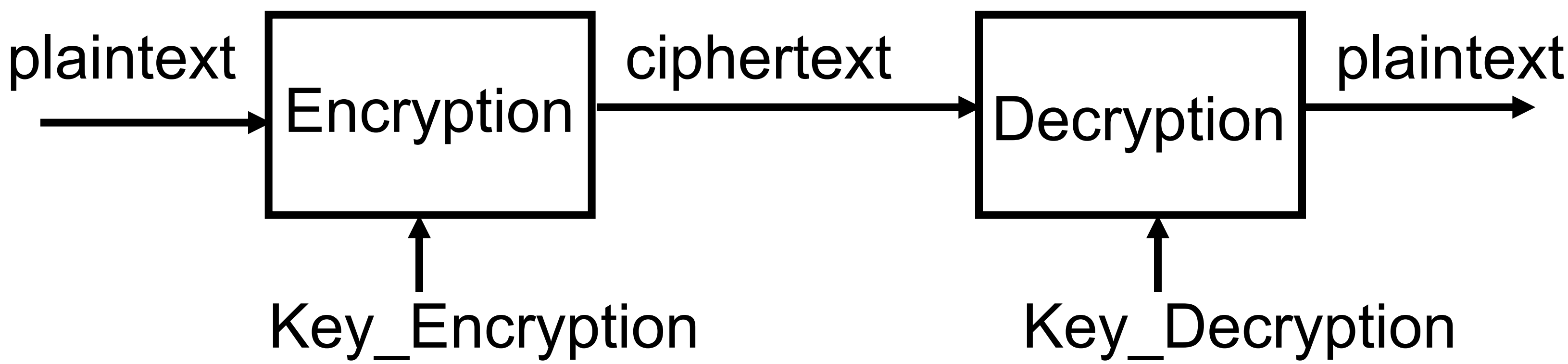
- The Code Book by Simon Singh

The science of secrets…

- **Cryptography:** the study of mathematical techniques related to aspects of providing information security services (create)
- **Cryptanalysis:** the study of mathematical techniques for attempting to defeat information security services (break)
- **Cryptology:** the study of cryptography and cryptanalysis (both)

Basic terminology in cryptography

- cryptography
- cryptanalysis
- cryptology
- plaintexts
- ciphertexts
- keys
- encryption
- decryption



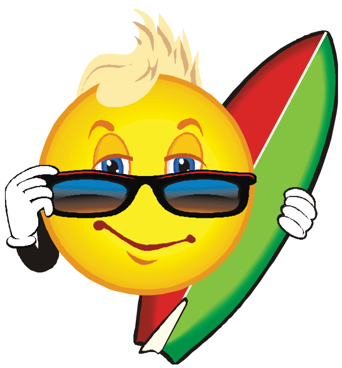
Cryptographic protocols

- Protocols that
  - Enable parties
  - Achieve objectives (goals)
  - Overcome adversaries (attacks)
- Need to understand
  - Who are the parties and the context in which they act
  - What are the goals of the protocols
  - What are the capabilities of adversaries

Cryptographic protocols: Parties

- The good guys

Alice



Bob

Introduction of Alice and Bob attributed to the original RSA paper.

- The bad guys

Carl

Eve

Check out wikipedia for a longer list of malicious crypto players.

Cryptographic protocols: Objectives/Goals

- Most basic problem:
  - Ensure security of communication over an insecure medium
- Basic security goals:
  - **Confidentiality** (secrecy, confidentiality)
    - Only the intended recipient can see the communication
  - **Authenticity** (integrity)
    - Communication is generated by the alleged sender

Goals of modern cryptography

- Pseudo-random number generation
- Non-repudiation: digital signatures
- Anonymity
- Zero-knowledge proof
- E-voting
- Secret sharing

Cryptographic protocols: Attackers

- Interaction with data and protocol
  - Eavesdropping or actively participating in the protocol
- Resources:
  - Computation, storage
  - Limited or unlimited
- Access to previously encrypted communication
  - Only encrypted information (ciphertext)
  - Pairs of message and encrypted version (plaintext, ciphertext)
- Interaction with the cipher algorithm
  - Choose or not for what message to have the encrypted version (chose ciphertext)

Interaction with data and protocol

- Passive: the attacker only monitors the communication. It threatens confidentiality.
  - **Example**: listen to the communication between Alice and Bob, and if it's encrypted try to decrypt it.
- Active: the attacker is actively involved in the protocol in deleting, adding or modifying data. It threatens all security services.
  - **Example**: Alice sends Bob a message: 'meet me today at 5', Carl intercepts the message and modifies it 'meet me tomorrow at 5', and then sends it to Bob.

Resources

- In practice attackers have limited computational power
- Some theoretical models consider that the attacker has unlimited computational resources

Attacker knowledge of previous encryptions

- Ciphertext-only attack
  - Attacker knows only the ciphertext
  - A cipher that is not resilient to this attack is not secure
- Known plaintext attack
  - Attacker knows one or several pairs of ciphertext and the corresponding plaintext
  - Goal is to be able to decrypt other ciphertexts for which the plaintext is unknown

Interactions with cipher algorithm

- <u>Chosen-plaintext attack</u>
  - Attacker can choose a number of messages and obtain the ciphertexts for them
  - *<u>Adaptive</u>*: the choice of plaintext depends on the ciphertext received from previous requests
- <u>Chosen-ciphertext attack</u>
  - Similar to the chosen-plaintext attack, but the cryptanalyst can choose a number of ciphertexts and obtain the plaintexts
  - *<u>Adaptive</u>*: the choice of ciphertext may depend on the plaintext received from previous requests

Approaches to secure communication

- Steganography
  - "covered writing"
  - hides the existence of a message
  - depends on secrecy of method
- Cryptography
  - "hidden writing"
  - hide the meaning of a message
  - depends on secrecy of a short key, not method

Shift cipher

- A substitution cipher
- The key space:
  - [0 .. 25]
- Encryption given a key K:
  - each letter in the plaintext P is replaced with the K' th letter following corresponding number (shift right)
- Decryption given K:
  - shift left

History: K = 3, Caesar's cipher

Shift Cipher: Cryptanalysis

- Can an attacker find K?
  - YES: by a bruteforce attack through exhaustive key search
  - key space is small (<= 26 possible keys)

- Lessons:
  - Cipher key space needs to be large enough
  - Exhaustive key search can be effective

Mono-alphabetical substitution cipher

- The key space: all permutations of $\Sigma = \{A, B, C, \ldots, Z\}$
- Encryption given a key (permutation) π:
  - each letter X in the plaintext P is replaced with $\pi(X)$
- Decryption given a key π:
  - each letter Y in the cipherext P is replaced with $\pi^{-1}(Y)$

Example:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
π =	B	A	D	C	Z	H	W	Y	G	O	Q	X	L	V	T	R	N	M	S	K	J	I	P	F	E	U

BECAUSE $\rightarrow$ AZDBJSZ

Cryptanalysis of mono-alphabetical substitution cipher

- Exhaustive search is infeasible
  - key space size is $26! \approx 4*10^{26}$
- Dominates the art of secret writing throughout the first millennium A.D.
- Thought to be unbreakable by many back then, until …. frequency analysis

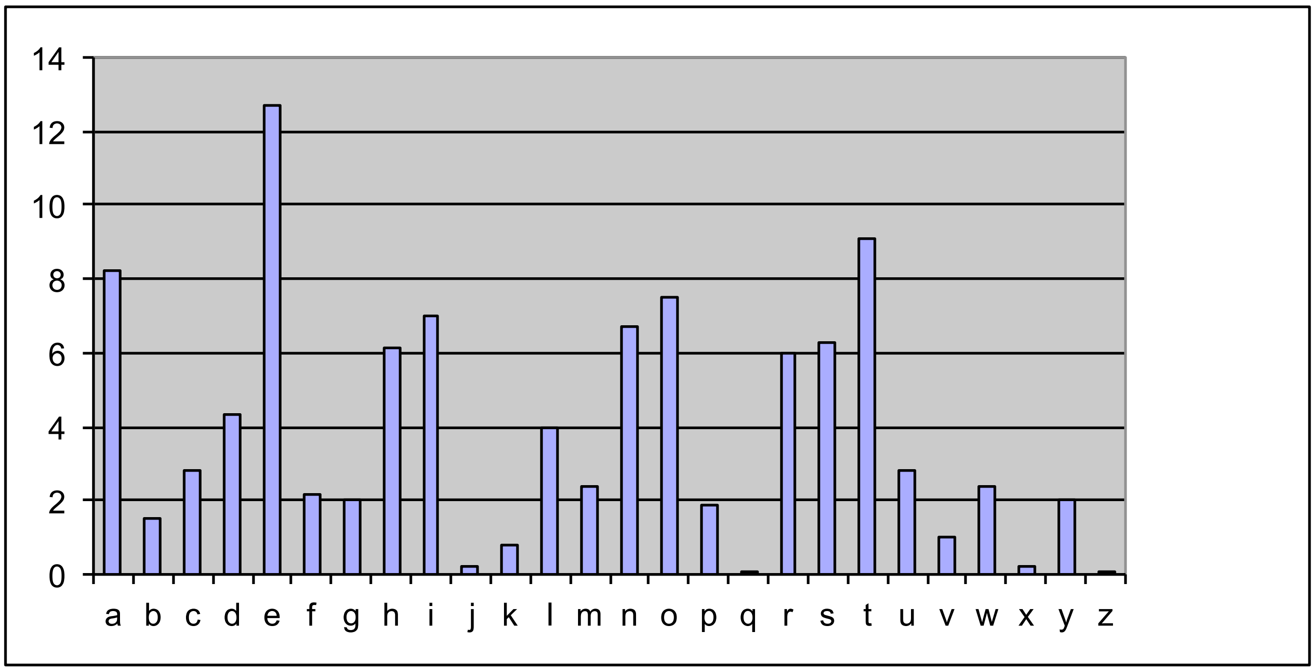
History of frequency analysis

- Discovered by the Arabs
  - Earliest known description of frequency analysis is in a book by the ninth-century scientist Al-Kindi
- Rediscovered or introduced from the Arabs in the Europe during the Renaissance
- Frequency analysis made substitution cipher insecure

Frequency analysis

- Each language has certain features: frequency of letters, or of groups of two or more letters
- Substitution ciphers preserve the language features
- Substitution ciphers are vulnerable to frequency analysis attacks

Frequency of letters in English



How to defeat frequency analysis?

- Use larger blocks as the basis of substitution. Rather than substituting one letter at a time, substitute 64 bits at a time, or 128 bits.
  - Leads to block ciphers such as DES & AES.

- Use different substitutions to get rid of frequency features.
  - Leads to polyalphabetical substitution ciphers
  - Stream ciphers

Towards polyalphabetic substitution ciphers

- Main weaknesses of monoalphabetic substitution ciphers
  - each letter in the ciphertext corresponds to only one letter in the plaintext letter
- Idea for a stronger cipher (1460’s by Alberti)
  - use more than one cipher alphabet, and switch between them when encrypting different letters
- Giovani Battista Bellaso published it in 1553
- Developed into a practical cipher by Blaise de Vigenère and published in 1586

Vigenère cipher

Definition:

Given m, a positive integer, $P = C = (Z_{26})^n$, and $K = (k_1, k_2, \ldots, k_m)$ a key, we define:

Encryption:

$e_k(p_1, p_2 \ldots p_m) = (p_1+k_1, p_2+k_2 \ldots p_m+k_m) \pmod{26}$

Decryption:

$d_k(c_1, c_2 \ldots c_m) = (c_1-k_1, c_2-k_2 \ldots c_m- k_m) \pmod{26}$

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Example:

Plaintext: C R Y P T O G R A P H Y

Key: L U C K L U C K L U C K

Ciphertext: N L A Z E I I B L J J I

Security of Vigenere cipher

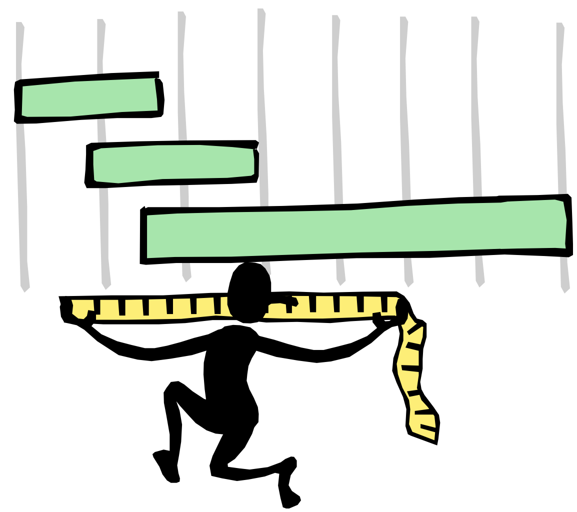
- Vigenere masks the frequency with which a character appears in a language: one letter in the ciphertext corresponds to multiple letters in the plaintext. Makes the use of frequency analysis more difficult
- Any message encrypted by a Vigenere cipher is a collection of as many shift ciphers as there are letters in the key

Vigenere cipher cryptanalysis

- Find the length of the key
- Divide the message into that many shift cipher encryptions
- Use frequency analysis to solve the resulting shift ciphers
  - how?

How to find the key length?

- For Vigenere, as the length of the keyword increases, the letter frequency shows less English-like characteristics and becomes more random
- Two methods to find the key length:
  - Kasisky test
  - Index of coincidence (Friedman)

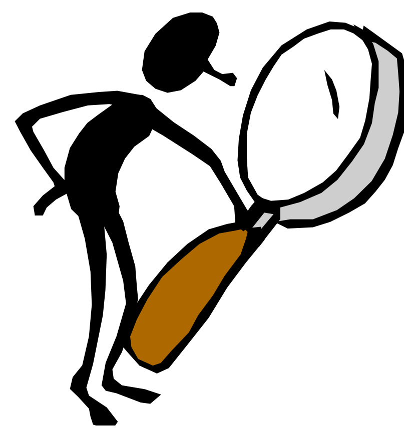


History of breaking Vigenere

- 1596 - Cipher was published by Vigenere
- 1854 - It is believed the Charles Babbage knew how to break it in 1854, but he did not published the results
- 1863 - Kasiski showed the Kasiski examination that showed how to break Vigenere
- 1920 - Friedman published ``The index of coincidence and its applications to cryptography"

Kasisky test for finding key length

- Observation: two identical segments of plaintext, will be encrypted to the same ciphertext, if the they occur in the text at the distance Δ, (Δ≡0 (mod m), m is the key length).
- Algorithm:
  - Search for pairs of identical segments of length at least 3
  - Record distances between the two segments: Δ1, Δ2, …
  - m divides gcd(Δ1, Δ2, …)



Example of the Kasisky test

Key	K I N G K I N G K I N G K I N G K I N G K I N G
PT	t h e s u n a n d t h e m a n i n t h e m o o n
CT	D P R Y E V N T N <u>B U K</u> W I A O X <u>B U K</u> W W B T

Repeating patterns (strings of length 3 or more) in ciphertext are likely due to repeating plaintext strings encrypted under repeating key strings; thus the location difference should be multiples of key lengths.

Security principles

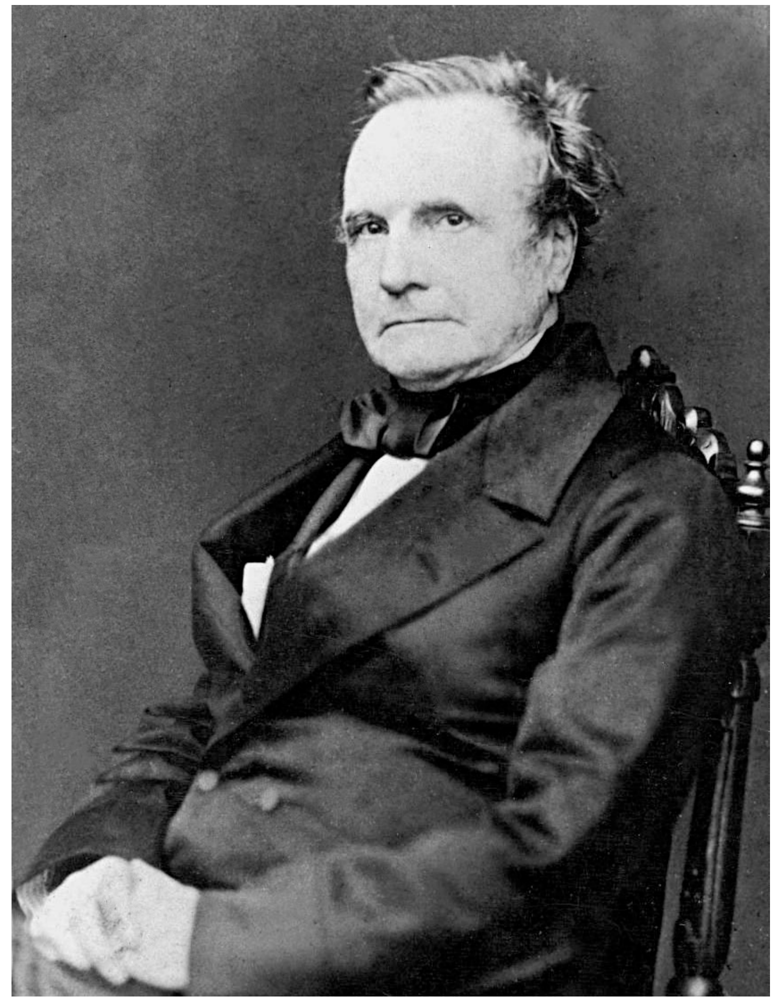
- **Kerckhoffs's Principle:**
  - A cryptosystem should be secure even if everything about the system, except the key, is public knowledge.
- **Shannon's maxim**:
  - "The enemy knows the system."
- Security by obscurity doesn’t work
- Should assume that the adversary knows the algorithm; the only secret the adversary is assumed to not know is the key
- What is the difference between the algorithm and the key?

Friedrich Wilhelm Kasiski (1805 – 1881)

- German infantry officer, cryptographer and archeologist.

Charles Babbage (1791 – 1871)

- English mathematician, philosopher, inventor and mechanical engineer who originated the concept of a programmable computer.
- Considered a "father of the computer", he invented the first mechanical computer that eventually led to more complex designs.

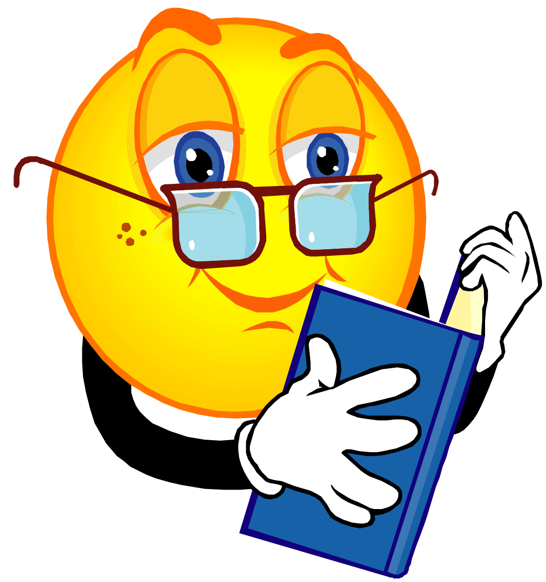


William Frederick Friedman (1891 – 1969)

- US Army cryptographer who ran the research division of the Army's Signals Intelligence Service (SIS) in the 1930s, and parts of its follow-on services into the 1950s.
- In 1940, people from his group, led by Frank Rowlett broke Japan's PURPLE cipher machine

Take home lessons

- Shift ciphers are easy to break using brute force attacks, they have small key space
- Substitution ciphers preserve language features and are vulnerable to frequency analysis attacks
- Vigenère cipher is vulnerable: once the key length is found, a cryptanalyst can apply frequency analysis



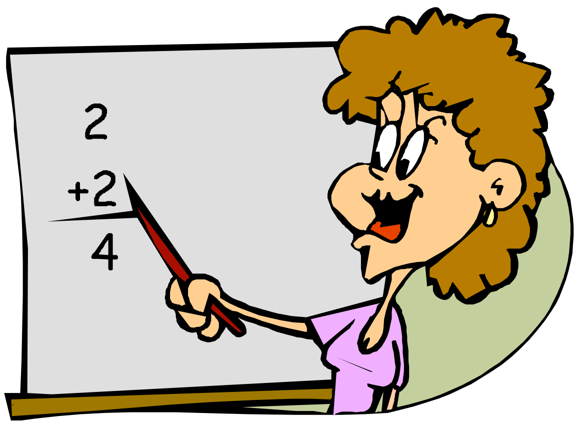
2: One-time Pad, information theoretic security, and stream ciphers

Readings for this lecture

- Required reading from wikipedia
  - One-Time Pad
  - Information theoretic security
  - Stream cipher
  - Pseudorandom number generator

- Stream ciphers on Dan Boneh's Cryptography I course on Coursera

Begin Math



Random Variable

A **discrete random variable, X,** consists of a finite set X, and a probability distribution defined on X. The probability that the random variable **X** takes on the value x is denoted **Pr**[**X** =x]; sometimes, we will abbreviate this to **Pr**[x] if the random variable **X** is fixed. It must be that

$$0 \leq \Pr[x] \quad \text{for all } x \in \mathcal{X}$$

$$\sum_{x \in \mathcal{X}} \Pr[x] = 1$$

Example of random variables

- Let random variable $\mathbf{D_1}$ denote the outcome of throwing one die (with numbers 0 to 5 on the 6 sides) randomly, then D={0,1,2,3,4,5} and $\mathbf{Pr}[\mathbf{D_1}=i] = 1/6$ for $0 \le i \le 5$
- Let random variable $\mathbf{D_2}$ denote the outcome of throwing a second such die randomly
- Let random variable $\mathbf{S_1}$ denote the sum of the two dice, then S ={0,1,2,…,10}, and

  $\mathbf{Pr}[\mathbf{S_1}=0] = \mathbf{Pr}[\mathbf{S_1}=10] = 1/36$

  $\mathbf{Pr}[\mathbf{S_1}=1] = \mathbf{Pr}[\mathbf{S_1}=9] = 2/36 = 1/18$

  …
- Let random variable $\mathbf{S_2}$ denote the sum of the two dice modulo 6, what is the distribution of $\mathbf{S_2}$?

Relationships between random variables

Assume **X** and **Y** are two random variables,
then we define:

- joint probability: **Pr**[x, y] is the probability that **X** takes value x and **Y** takes value y.
- conditional probability: **Pr**[x|y] is the probability that **X** takes value x given that **Y** takes value y.

  $$\mathbf{Pr}[x|y] = \mathbf{Pr}[x, y] \; / \; \mathbf{Pr}[y]$$

- independent random variables: **X** and **Y** are said to be independent if **Pr**[x,y] = **Pr**[x]P[y], for all x ∈ X and all y ∈ **Y**.

Examples

- Joint probability of $\mathbf{D_1}$ and $\mathbf{D_2}$ for $0 \le i, j \le 5$, $\mathbf{Pr}[\mathbf{D_1}=i, \mathbf{D_2}=j]$ = ?
- What is the conditional probability $\mathbf{Pr}[\mathbf{D_1}=i \mid \mathbf{D_2}=j]$ for $0 \le i, j \le 5$?
- Are $\mathbf{D_1}$ and $\mathbf{D_2}$ independent?

- Suppose $\mathbf{D_1}$ is plaintext and $\mathbf{D_2}$ is key, and $\mathbf{S_1}$ and $\mathbf{S_2}$ are ciphertexts of two different ciphers, which cipher would you use?

Practice exercises

- What is the joint probability of $\mathbf{D_1}$ and $\mathbf{S_1}$?
- What is the joint probability of $\mathbf{D_2}$ and $\mathbf{S_2}$?
- What is the conditional probability

  $\mathbf{Pr}[\mathbf{S_1}=s \mid \mathbf{D_1}=i]$ for $0 \le i \le 5$ and $0 \le s \le 10$?
- What is the conditional probability

  $\mathbf{Pr}[\mathbf{D_1}=i \mid \mathbf{S_2}=s]$ for $0 \le i \le 5$ and $0 \le s \le 5$?
- Are $\mathbf{D_1}$ and $\mathbf{S_1}$ independent?
- Are $\mathbf{D_1}$ and $\mathbf{S_2}$ independent?

Bayes' Theorem

If P[y] > 0 then

$$P[x \mid y] = \frac{P[x]P[y \mid x]}{P[y]}$$

$$P[y] = \sum_{x \in X} P[x, y] = \sum_{x \in X} P[x]p[y \mid x]$$

Corollary

X and Y are independent random variables iff P[x|y] = P[x], for all x ∈ X and all y ∈ Y.

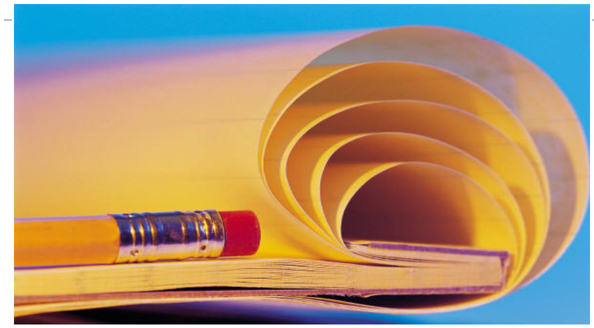
End Math

}

One-Time Pad

- Fix the vulnerability of the Vigenere cipher by using very long keys
- Key is a random string that is at least as long as the plaintext
- Encryption is similar to shift cipher
- Invented by Vernam in the 1920s

One-Time Pad



Let $Z_m = \{0,1,\ldots,m-1\}$ be the alphabet.

Plaintext space = Ciphtertext space = Key space = $(Z_m)^n$

The key is chosen uniformly randomly

Plaintext $X = (x_1\ x_2\ \ldots\ x_n)$

Key $K = (k_1\ k_2\ \ldots\ k_n)$

Ciphertext $Y = (y_1\ y_2\ \ldots\ y_n)$

$e_k(X) = (x_1+k_1\ \ x_2+k_2\ \ldots\ x_n+k_n) \bmod m$

$d_k(Y) = (y_1-k_1\ \ \ y_2-k_2\ \ldots\ \ y_n-k_n) \bmod m$

Binary version of One-Time Pad

Plaintext space = Ciphtertext space =

Keyspace = $\{0,1\}^n$

Key is chosen randomly

For example:

- Plaintext is 11011011
- Key is 01101001
- Then ciphertext is 10110010

Bit operators

- Bit AND

  $0 \wedge 0 = 0$ $0 \wedge 1 = 0$ $1 \wedge 0 = 0$ $1 \wedge 1 = 1$

- Bit OR

  $0 \vee 0 = 0$ $0 \vee 1 = 1$ $1 \vee 0 = 1$ $1 \vee 1 = 1$

- Addition mod 2 (also known as Bit XOR)

  $0 \oplus 0 = 0$ $0 \oplus 1 = 1$ $1 \oplus 0 = 1$ $1 \oplus 1 = 0$

- Can we use operators other than Bit XOR for binary version of One-Time Pad?

How good is One-Time Pad?

- Intuitively, it is secure …
  - The key is random, so the ciphertext is completely random
- How to formalize the confidentiality requirement?
  - Want to say "certain thing" is not learnable by the adversary (who sees the ciphertext). But what is the "certain thing"?
- Which (if any) of the following is the correct answer?
  - The key.
  - The plaintext.
  - Any bit of the plaintext.
  - Any information about the plaintext.
    - E.g., the first bit is 1, the parity is 0, or that the plaintext is not "aaaa", and so on

Shannon (Information-Theoretic) Security = Perfect Secrecy

Basic idea: Ciphertext should reveal no "information" about Plaintext

Definition.

An encryption over a message space ***M*** is perfectly secure if

∀ probability distribution over ***M***

∀ message m ∈ ***M***

∀ ciphertext c ∈ ***C*** for which Pr[C=c] > 0

We have

Pr [PT=m | CT=c] = Pr [PT = m]

Explanation of the definition

- Pr [**PT** = m] is what the adversary believes the probability that the plaintext is m, before seeing the ciphertext
- Pr [**PT** = m | **CT**=c] is what the adversary believes after seeing that the ciphertext is c
- Pr [**PT**=m | **CT**=c] = Pr [**PT** = m] means that after knowing that the ciphertext is C_0, the adversary's belief does not change

Equivalent definition of Perfect Secrecy

Definition. An encryption scheme over a message space ***M*** is perfectly secure if ∀ probability distribution over ***M*,** the random variables **PT** and **CT** are independent. That is,

∀ message $m \in M$

∀ ciphertext $c \in C$

$$\Pr[\mathbf{PT}=m \wedge \mathbf{CT}=c] = \Pr[\mathbf{PT} = m] \Pr[\mathbf{CT} = c]$$

Note that this is equivalent to: When $\Pr[\mathbf{CT} = c] \neq 0$, we have

$$\Pr[\mathbf{PT} = m] = \Pr[\mathbf{PT}=m \wedge \mathbf{CT}=c] / \Pr[\mathbf{CT} = c] = \Pr[\mathbf{PT}=m \mid \mathbf{CT}=c]$$

This is also equivalent to: When $\Pr[\mathbf{PT} = m] \neq 0$, we have

$$\Pr[\mathbf{CT} = c] = \Pr[\mathbf{PT}=m \wedge \mathbf{CT}=c] / \Pr[\mathbf{PT} = m] = \Pr[\mathbf{CT}=c \mid \mathbf{PT}=m]$$

Example for information theoretical security

- Consider an example of encrypting the result of a 6-side dice (1 to 6).
  - Method 1: randomly generate K=[0..5], ciphertext is result + K.
    - What is plaintext distribution? After seeing that the ciphertext is 6, what could be the plaintext. After seeing that the ciphertext is 11, what could be the plaintext?
  - Method 2: randomly generate K=[0..5], ciphertext is (result + K) mod 6.
    - Same questions.
    - Can one do a brute-force attack?

Perfect secrecy

- Fact: When keys are uniformly chosen in a cipher, the cipher has perfect secrecy iff. the number of keys encrypting M to C is the same for any (M,C)
  - This implies that
    $\forall c \forall m_1 \forall m_2 \; Pr[\mathbf{CT}=c \mid \mathbf{PT}=m_1] = Pr[\mathbf{CT}=c \mid \mathbf{PT}=m_2]$

- One-time pad has perfect secrecy when limited to messages over the same length (Proof?)

Key randomness in One-Time Pad

- One-Time Pad uses a very long key, what if the key is not chosen randomly, instead, texts from, e.g., a book are used as keys.
  - this is not One-Time Pad anymore
  - this does not have perfect secrecy
  - this can be broken
  - How?
- The key in One-Time Pad should never be reused.
  - If it is reused, it is Two-Time Pad, and is insecure!
  - Why?

Usage of One-Time Pad

- To use one-time pad, one must have keys as long as the messages.
- To send messages totaling certain size, sender and receiver must agree on a shared secret key of that size.
  - typically by sending the key over a secure channel
- This is difficult to do in practice.
- Can't one use the channel for send the key to send the messages instead?
- Why is OTP still useful, even though difficult to use?

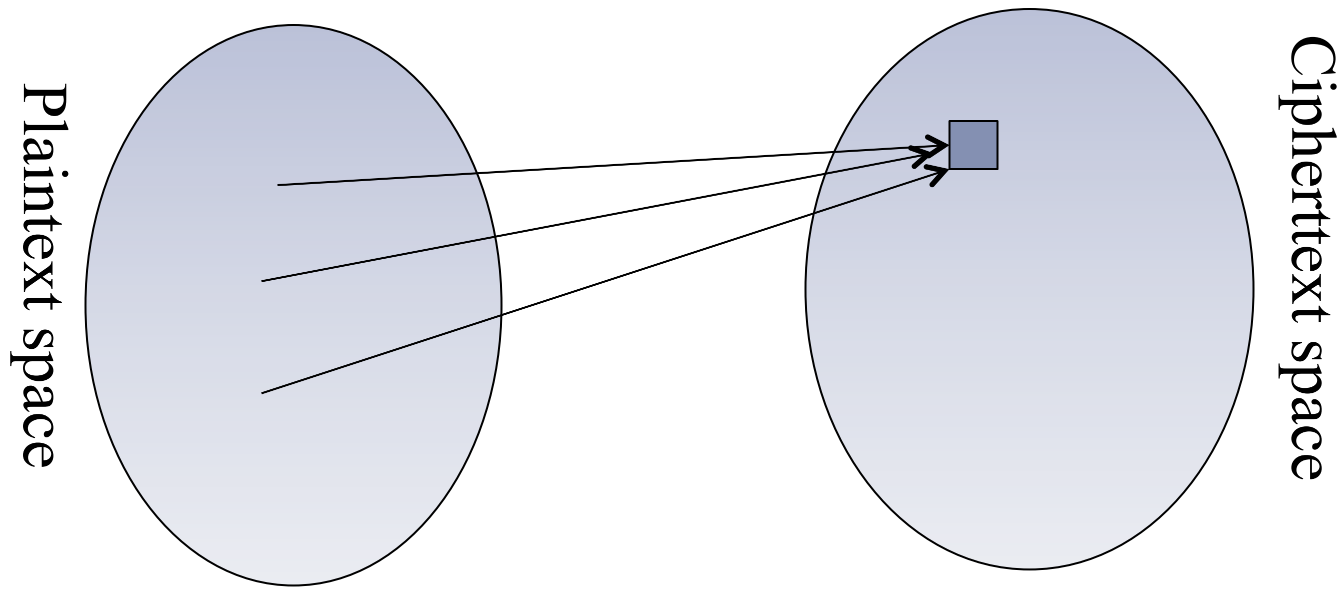
Usage of One-Time Pad

- The channel for distributing keys may exist at a different time from when one has messages to send.
- The channel for distributing keys may have the property that keys can be leaked, but such leakage will be detected
  - Such as in Quantum cryptography

The "bad news" theorem for Perfect Secrecy

- Question: OTP requires key as long as messages, is this an inherent requirement for achieving perfect secrecy?
- Answer. Yes. Perfect secrecy implies that key-length ≥ msg-length

Proof:



- Implication: Perfect secrecy difficult to achieve in practice

Stream ciphers

- In One-Time Pad, a key is a random string of length at least the same as the message
- Stream ciphers:
  - Idea: replace "rand" by "pseudo rand"
  - Use Pseudo Random Number Generator
  - PRNG: $\{0,1\}^s \rightarrow \{0,1\}^n$
    - expand a short (e.g., 128-bit) random seed into a long (e.g., 10^6 bit) string that "looks random"
  - Secret key is the seed
  - $E_{key}[M] = M \oplus PRNG(key)$

The RC4 stream cipher

- A proprietary cipher owned by RSA, designed by Ron Rivest in 1987.
- Became public in 1994.
- Simple and effective design.
- Variable key size (typical 40 to 256 bits),
- Output unbounded number of bytes.
- Widely used (web SSL/TLS, wireless WEP).
- Extensively studied, not a completely secure PRNG, first part of output biased, when used as stream cipher, should use RC4-Drop[n]
  - Which drops first n bytes before using the output
  - Conservatively, set n=3072

Pseudo-random number generator

- Useful for cryptography, simulation, randomized algorithm, etc.
  - Stream ciphers, generating session keys
- The same seed always gives the same output stream
  - Why is this necessary for stream ciphers?
- Simulation requires uniform distributed sequences
  - E.g., having a number of statistical properties
- **Cryptographically secure pseudo-random number generator** requires unpredictable sequences
  - satisfies the "next-bit test“: given consecutive sequence of bits output (but not seed), next bit must be hard to predict
- Some PRNG’s are weak: knowing output sequence of sufficient length, can recover key.
  - Do not use these for cryptographic purposes

Properties of stream ciphers

- Typical stream ciphers are very fast
- Widely used, often incorrectly
  - Content Scrambling System (uses Linear Feedback Shift Registers incorrectly),
  - Wired Equivalent Privacy (uses RC4 incorrectly)
  - SSL (uses RC4, SSLv3 has no known major flaw)

Security properties of stream ciphers

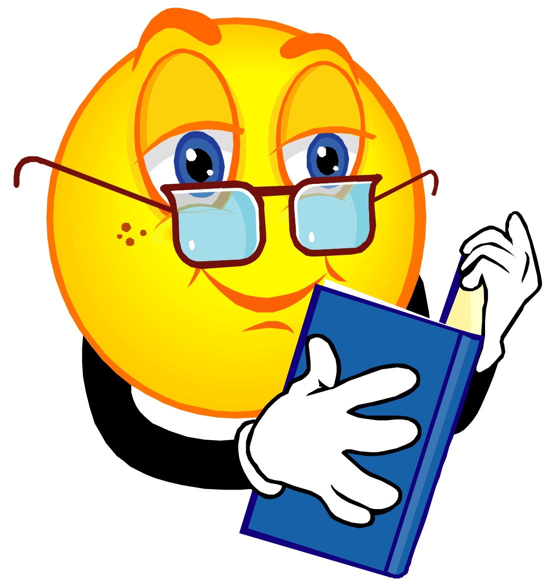
- Under known plaintext, chosen plaintext, or chosen ciphertext, the adversary knows the key stream (i.e., PRNG(key))
  - Security depends on PRNG
  - PRNG must be "unpredictable"
- Do stream ciphers have perfect secrecy?
- How to break a stream cipher in a brute-force way?
- If the same key stream is used twice, then easy to break.
  - This is a fundamental weakness of stream ciphers; it exists even if the PRNG used in the ciphers is strong

Using stream ciphers in practice

- If the same key stream is used twice, then easy to break.
  - This is a fundamental weakness of stream ciphers; it exists even if the PRNG used in the ciphers is strong
- In practice, one key is used to encrypt many messages
  - Example: Wireless communication
  - Solution: Use Initial vectors (IV).
  - $E_{key}[M] = [IV, M \oplus PRNG(key \mid\mid IV)]$
    - IV is sent in clear to receiver;
    - IV needs integrity protection, but not confidentiality protection
    - IV ensures that key streams do not repeat, but does not increase cost of brute-force attacks
    - Without key, knowing IV still cannot decrypt
  - Need to ensure that IV never repeats! How?

Take home lessons

- OTP has perfect forward secrecy if key is used once, is random and as long as the message
- One has to be very careful with how he uses stream ciphers: if the keystream repeats then it's easy to decrypt all messages encrypted with the keystream



3: Semantic security, block ciphers and encryption modes

Readings for this lecture

- Required reading from wikipedia
  - Block Cipher
  - Ciphertext Indistinguishability
  - Block cipher modes of operation

Notation for symmetric-key encryption

- A symmetric-key encryption scheme is comprised of three algorithms
  - **Gen** the key generation algorithm
    - The algorithm must be probabilistic/randomized
    - Output: a key k
  - **Enc** the encryption algorithm
    - Input: key k, plaintext m
    - Output: ciphertext $c := \mathbf{Enc}_k(m)$
  - **Dec** the decryption algorithm
    - Input: key k, ciphertext c
    - Output: plaintext $m := \mathbf{Dec}_k(m)$

Requirement: $\forall k \; \forall m \; [\; \mathbf{Dec}_k(\mathbf{Enc}_k(m)) = \mathrm{m} \;]$

Randomized vs. deterministic encryption

- Encryption can be randomized,
  - i.e., same message, same key, run encryption algorithm twice, obtains two different ciphertexts
  - E.g, $\mathbf{Enc}_k[m] = (r, PRNG[k||r] \oplus m)$, i.e., the ciphertext includes two parts, a randomly generated r, and a second part
  - Ciphertext space can be arbitrarily large
- Decryption is deterministic in the sense that
  - For the same ciphertext and same key, running decryption algorithm twice always result in the same plaintext
- Each key induces a one-to-many mapping from plaintext space to ciphertext space
  - Corollary: ciphertext space must be equal to or larger than plaintext space

Towards computational security

- Perfect secrecy is too difficult to achieve.
- Computational security uses two relaxations:
  - Security is preserved only against **efficient** (computationally bounded) adversaries
    - Adversary can only run in feasible amount of time
  - Adversaries can potentially succeed with some **very small probability** (that we can ignore the case it actually happens)
- Two approaches to formalize computational security: concrete and asymptotic

The concrete approach

- Quantifies the security by explicitly bounding the maximum success probability of adversary running with certain time:
  - "A scheme is (t,ε)-secure if **every** adversary running for time at most t succeeds in breaking the scheme with probability at most ε"
  - Example: a strong encryption scheme with n-bit keys may be expected to be $(t, t/2^n)$-secure.
    - $N=128$, $t=2^{60}$, then $\varepsilon= 2^{-68}$. (# of seconds since big bang is 2^{58})
- Makes more sense with symmetric encryption schemes because they use fixed key lengths.

The asymptotic approach

- A cryptosystem has a security parameter
  - E.g., number of bits in the RSA algorithm (1024,2048,…)
- Typically, the key length depends on the security parameter
  - The bigger the security parameter, the longer the key, the more time it takes to use the cryptosystem, and the more difficult it is to break the scheme
- The crypto system must be efficient, i.e., runs in time polynomial in the security parameter
- "A scheme is secure if every Probabilistic Polynomial Time (PPT) algorithm succeeds in breaking the scheme with only negligible probability"
  - "negligible" roughly means goes to 0 exponentially fast as the security parameter increases

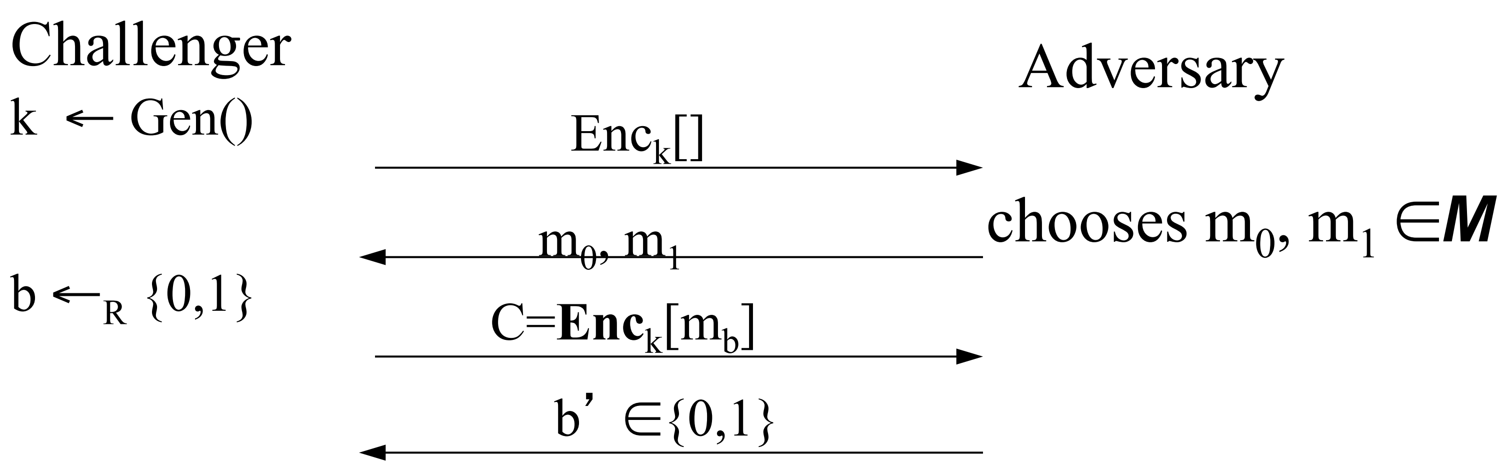
Defining security

- Desire "semantic security", i.e., having access to the ciphertext does not help adversary to compute any function of the plaintext.
  - Difficult to use

- Equivalent notion: Adversary cannot distinguish between the ciphertexts of two plaintexts

Towards IND-CPA security

- Ciphertext Indistinguishability under a Chosen-Plaintext Attack: Define the following IND-CPA experiment :
  - Involving an Adversary and a Challenger
  - Instantiated with an Adversary algorithm *A*, and an encryption scheme Π = (Gen, Enc, Dec)



Adversary wins if b=b'

The IND-CPA experiment explained

- A k is generated by Gen()
- Adversary is given oracle access to $Enc_k(\cdot)$,
  - Oracle access: one gets its question answered without knowing any additional information
- Adversary outputs a pair of equal-length messages m_0 and m_1
- A random bit b is chosen, and adversary is given $Enc_k(m_b)$
  - Called the challenge ciphertext
- Adversary does any computation it wants, while still having oracle access to $Enc_k(\cdot)$, and outputs b'
- Adversary wins if b=b'

CPA-secure (aka IND-CPA security)

- A encryption scheme Π = (Gen, Enc, Dec) has indistinguishable encryption under a chosen-plaintext attack (i.e., is IND-CPA secure) iff. for all PPT adversary *A*, there exists a negligible function negl such that
  - Pr[***A*** *wins* in IND-CPA experiment] $\leq$ ½ + negl(n)

- No deterministic encryption scheme is CPA-secure. Why?

Another (equivalent) explanation of IND-CPA security

- Ciphertext indistinguishability under chosen plaintext attack (IND-CPA)
  - Challenger chooses a random key K
  - Adversary chooses a number of messages and obtains their ciphertexts under key K
  - Adversary chooses two equal-length messages m_0 and m_1, sends them to a Challenger
  - Challenger generates $C=E_K[m_b]$, where b is a uniformly randomly chosen bit, and sends C to the adversary
  - Adversary outputs b' and wins if b=b'
  - Adversary advantage is | Pr[Adv wins] – ½ |
  - Adversary should not have a non-negligible advantage
    - E.g, Less than, e.g., $1/2^{80}$ when the adversary is limited to certain amount of computation;
    - decreases exponentially with the security parameter (typically length of the key)

Intuition of IND-CPA security

- Perfect secrecy means that any plaintext is encrypted to a given ciphertext with the same probability, i.e., given any pair of M_0 and M_1, the probabilities that they are encrypted into a ciphertext C are the same
  - Hence no adversary can tell whether C is ciphertext of M_0 or M_1.
- IND-CPA means
  - With bounded computational resources, the adversary cannot tell which of M_0 and M_1 is encrypted in C
- Stream ciphers can be used to achieve IND-CPA security when the underlying PRNG is cryptographically strong
  - (i.e., generating sequences that cannot be distinguished from random, even when related seeds are used)

Computational security vs. information theoretic security

- If a cipher has only computational security, then it can be broken by a brute force attack, e.g., enumerating all possible keys
  - Weak algorithms can be broken with much less time
- How to prove computational security?
  - Assume that some problems are hard (requires a lot of computational resources to solve), then show that breaking security means solving the problem
- Computational security is foundation of modern cryptography.

Why block ciphers?

- One thread of defeating frequency analysis
  - Use different keys in different locations
  - Example: one-time pad, stream ciphers
- Another way to defeat frequency analysis
  - Make the unit of transformation larger, rather than encrypting letter by letter, encrypting block by block
  - Example: block cipher

Block ciphers

- An n-bit plaintext is encrypted to an n-bit ciphertext
  - P : $\{0,1\}^n$
  - C : $\{0,1\}^n$
  - K : $\{0,1\}^s$
  - **E**: $K \times P \rightarrow C$: E_k: a permutation on $\{0,1\}^n$
  - **D**: $K \times C \rightarrow P$: D_k is E_k^{-1}
  - Block size: n
  - Key size: s

Data Encryption Standard (DES)

- Designed by IBM, with modifications proposed by the National Security Agency
- US national standard from 1977 to 2001
- De facto standard
- Block size is 64 bits;
- Key size is 56 bits
- Has 16 rounds
- Designed mostly for hardware implementations
  - Software implementation is somewhat slow
- Considered insecure now
  - vulnerable to brute-force attacks

Attacking block ciphers

- Types of attacks to consider
  - known plaintext: given several pairs of plaintexts and ciphertexts, recover the key (or decrypt another block encrypted under the same key)
  - how would chosen plaintext and chosen ciphertext be defined?
- Standard attacks
  - exhaustive key search
  - dictionary attack
  - differential cryptanalysis, linear cryptanalysis
- Side channel attacks.

DES's main vulnerability is short key size.

Chosen-plaintext dictionary attacks against block ciphers

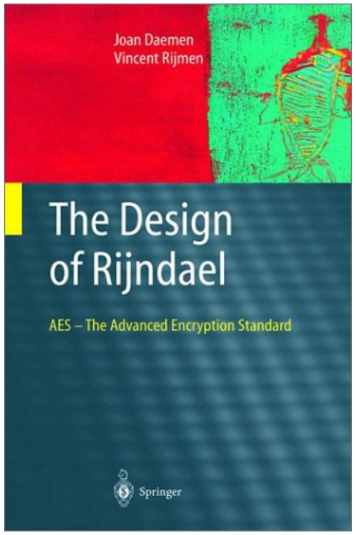
- Construct a table with the following entries
  - (K, $E_K[0]$) for all possible key K
  - Sort based on the second field (ciphertext)
  - How much time does this take?
- To attack a new key K (under chosen message attacks)
  - Choose 0, obtain the ciphertext C, looks up in the table, and finds the corresponding key
  - How much time does this step take?
- Trade off space for time

Advanced Encryption Standard

- In 1997, NIST made a formal call for algorithms stipulating that the AES would specify an unclassified, publicly disclosed encryption algorithm, available royalty-free, worldwide.
- Goal: replace DES for both government and private-sector encryption.
- The algorithm must implement symmetric key cryptography as a block cipher and (at a minimum) support block sizes of 128-bits and key sizes of 128-, 192-, and 256-bits.
- In 1998, NIST selected 15 AES candidate algorithms.
- On October 2, 2000, NIST selected **Rijndael** (invented by Joan Daemen and Vincent Rijmen) to as the AES.

AES features

- Designed to be efficient in both hardware and software across a variety of platforms.
- Block size: 128 bits
- Variable key size: **128, 192, or 256 bits.**
- No known weaknesses



Need for Encryption Modes

- A block cipher encrypts only one block
- Needs a way to extend it to encrypt an arbitrarily long message
- Want to ensure that if the block cipher is secure, then the encryption is secure
- Aims at providing Semantic Security (**IND-CPA)** assuming that the underlying block ciphers are strong

Block Cipher Encryption Modes: ECB

- Message is broken into independent blocks;
- Electronic Code Book (ECB): each block encrypted separately.
- **Encryption: $c_i = E_k(x_i)$**
- **Decrytion: $x_i = D_k(c_i)$**

Properties of ECB

- Deterministic:
  - the same data block gets encrypted the same way,
    - reveals patterns of data when a data block repeats
  - when the same key is used, the same message is encrypted the same way
- Usage: not recommended to encrypt more than one block of data

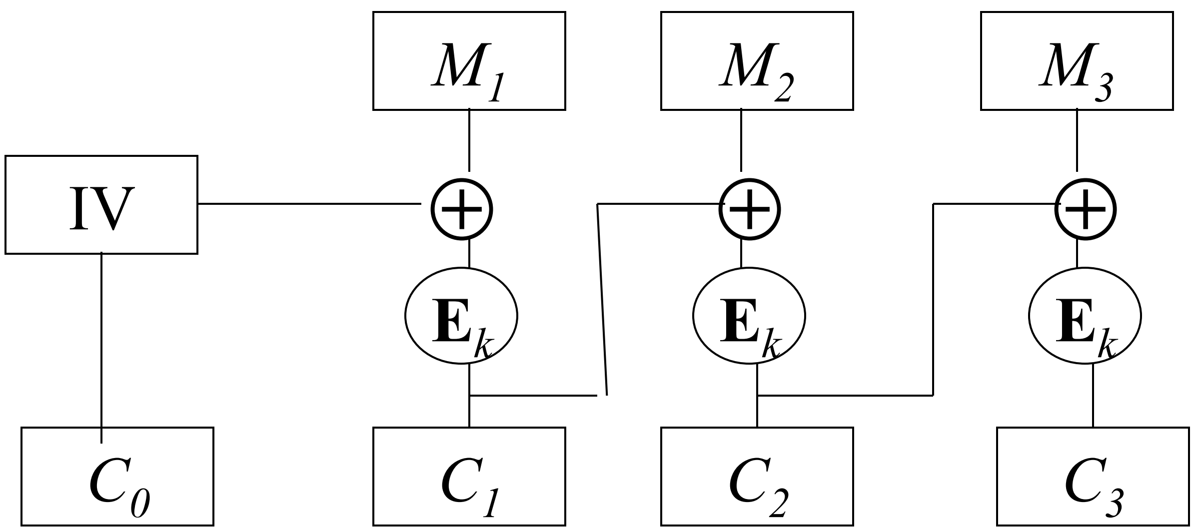
- How to break the semantic security (IND-CPA) of a block cipher with ECB?

DES Encryption Modes: CBC

- Cipher Block Chaining (CBC):
  - Uses a random Initial Vector (IV)
  - Next input depends upon previous output

    Encryption: $C_i = E_k(M_i \oplus C_{i-1})$, **with** $C_0 = IV$

    Decryption: $M_i = C_{i-1} \oplus D_k(C_i)$, **with** $C_0 = IV$

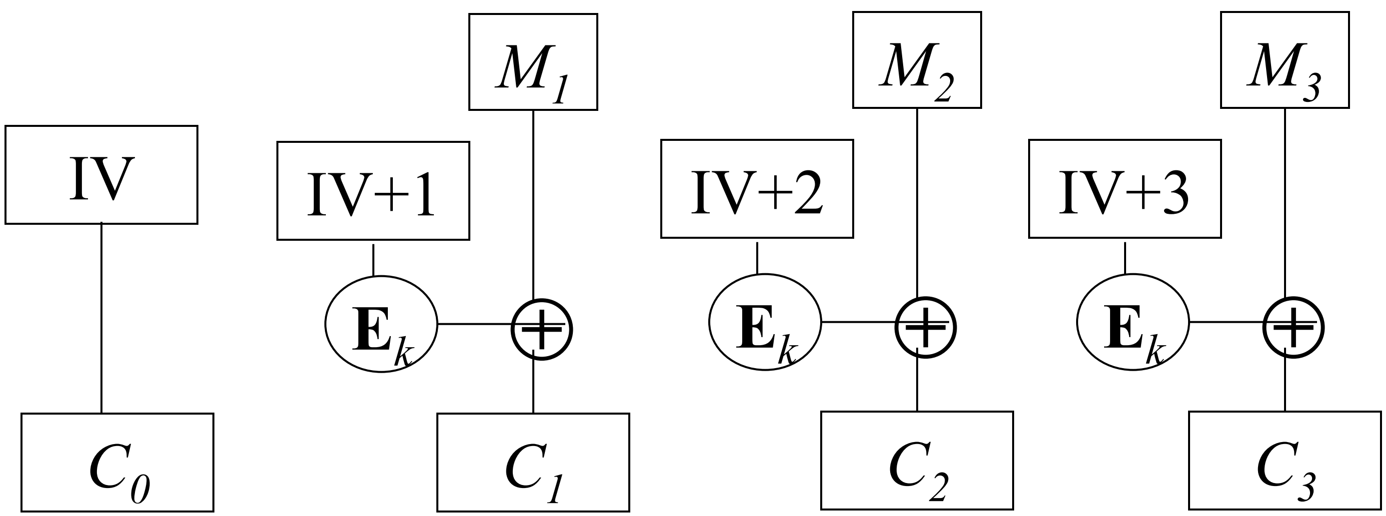


Properties of CBC

- Randomized encryption: repeated text gets mapped to different encrypted data.
  - can be proven to provide IND-CPA assuming that the block cipher is secure (i.e., it is a Pseudo Random Permutation (PRP)) and that IV's are randomly chosen and the IV space is large enough (at least 64 bits)
- Each ciphertext block depends on all preceding plaintext blocks.
- Usage: chooses **random** IV and protects the **integrity** of IV
  - The IV is not secret (it is part of ciphertext)
  - The adversary cannot control the IV

Encryption modes: CTR

- Counter Mode (CTR): Defines a stream cipher using a block cipher
  - Uses a random IV, known as the counter
  - Encryption: C_0=IV, $C_i = M_i \oplus E_k[IV+i]$
  - Decryption: IV=C_0, $M_i = C_i \oplus E_k[IV+i]$

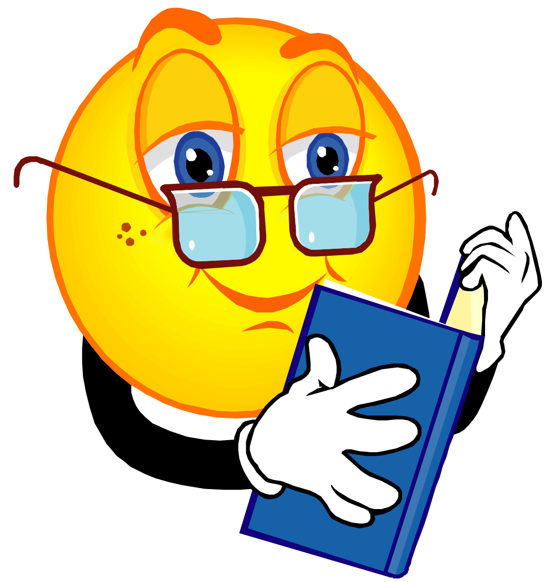


Properties of CTR

- Gives a stream cipher from a block cipher
- Randomized encryption:
  - when starting counter is chosen randomly
- Random Access: encryption and decryption of a block can be done in random order, very useful for hard-disk encryption.
  - E.g., when one block changes, re-encryption only needs to encrypt that block. In CBC, all later blocks also need to change

Take home lessons

- AES is the current standard for block ciphers
- Key size most important factor to deter brute force attacks
- Number of rounds also important to deter attacks
- When encrypting data larger than the block the encryption mode choice is crucial for the security of the encrypted data

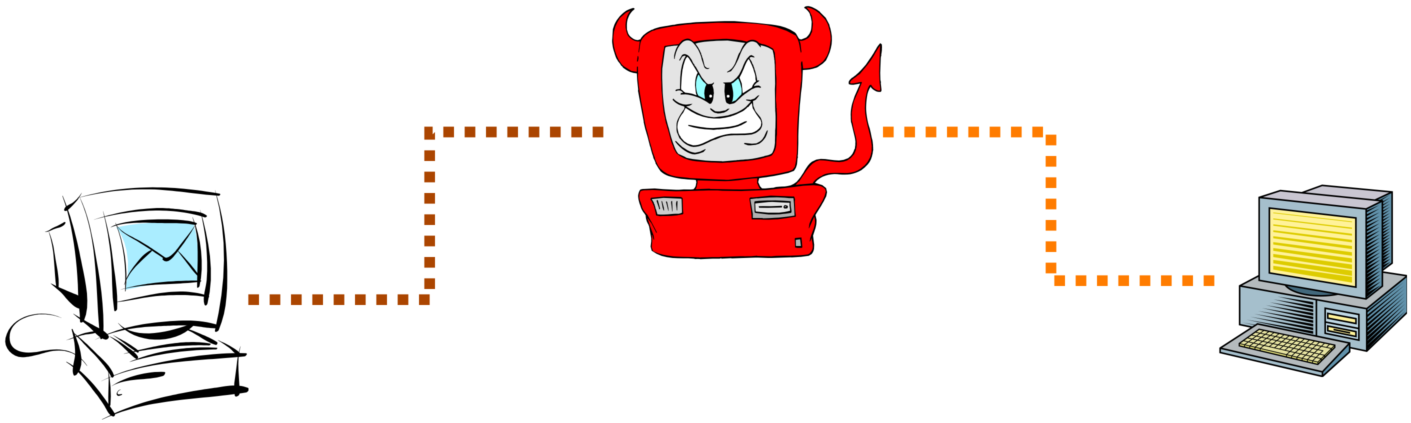


4: Cryptographic hash functions and message authentication codes

Readings for This Lecture

- Wikipedia
  - Cryptographic Hash Functions
  - Message Authentication Code

Data Integrity and Source Authentication



- Encryption does not protect data from modification by another party.
  - Why?
- Need a way to ensure that data arrives at destination in its original form as sent by the sender and it is coming from an authenticated source.

Hash Functions

- A hash function maps a message of an arbitrary length to a m-bit output
  - output known as the fingerprint or the message digest
- What is an example of hash functions?
  - Give a hash function that maps Strings to integers in [0,2^{32}-1]
- Cryptographic hash functions are hash functions with additional security requirements

Using Hash Functions for Message Integrity

- Method 1: Uses a Hash Function h, assuming an authentic (adversary cannot modify) channel for short messages
  - Transmit a message M over the normal (insecure) channel
  - Transmit the message digest h(M) over the secure channel
  - When receiver receives both M' and h, how does the receiver check to make sure the message has not been modified?

- This is insecure. How to attack it?
- A hash function is a many-to-one function, so collisions can happen.

Security Requirements for Cryptographic Hash Functions

Given a function $h: X \to Y$, then we say that h is:

- preimage resistant (one-way):

  if given $y \in Y$ it is computationally infeasible to find a value $x \in X$ s.t. $h(x) = y$

- 2-nd preimage resistant (weak collision resistant):

  if given $x \in X$ it is computationally infeasible to find a value $x' \in X$, s.t. $x' \neq x$ and $h(x') = h(x)$

- collision resistant (strong collision resistant):

  if it is computationally infeasible to find two distinct values $x', x \in X$, s.t. $h(x') = h(x)$

Usages of Cryptographic Hash Functions

- Software integrity
  - E.g., tripwire
- Timestamping
  - How to prove that you have discovered a secret on an earlier date without disclosing it?
- Covered later
  - Message authentication
  - One-time passwords
  - Digital signature

Bruteforce Attacks on Hash Functions

- Attacking one-wayness
  - Goal: given $h:X \to Y$, $y \in Y$, find x such that $h(x)=y$
  - Algorithm:
    - pick a random value x in X, check if $h(x)=y$, if $h(x)=y$, returns x; otherwise iterate
    - after failing q iterations, return fail
  - The average-case success probability is

$$\varepsilon = 1 - \left(1 - \frac{1}{|Y|}\right)^q \approx \frac{q}{|Y|}$$

  - Let $|Y|=2^m$, to get ε to be close to 0.5, $q \approx 2^{m-1}$

Bruteforce Attacks on Hash Functions

- Attacking collision resistance
  - Goal: given h, find x, x' such that h(x)=h(x')
  - Algorithm: pick a random set X_0 of q values in X
    for each $x \in X_0$, computes $y_x = h(x)$ if $y_x = y_{x'}$ for some $x' \neq x$ then return (x,x') else fail
  - The average success probability is

$$1-\left(1-\frac{1}{|Y|}\right)^{\frac{q(q-1)}{2}} \approx 1-e^{-\frac{q(q-1)}{2|Y|}}$$

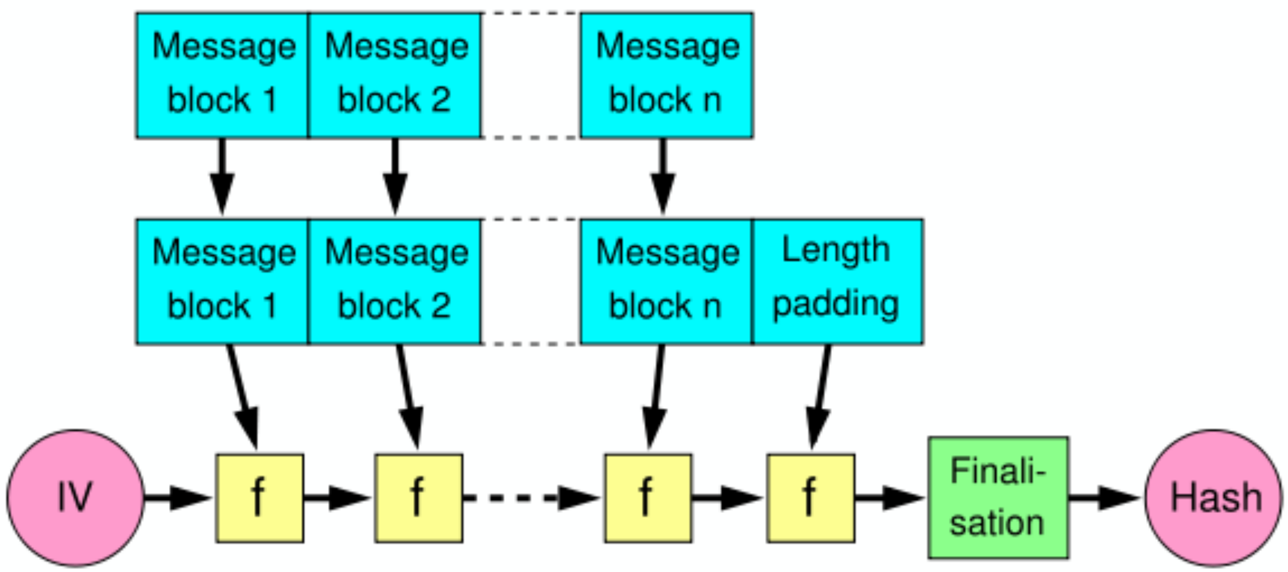
  - Let $|Y| = 2^m$, to get ε to be close to 0.5, $q \approx 2^{m/2}$
  - This is known as the birthday attack.

Well Known Hash Functions

- MD5
  - output 128 bits
  - collision resistance completely broken by researchers in China in 2004
- SHA1
  - output 160 bits
  - no collision found yet, but method exist to find collisions in less than 2^80
  - considered insecure for collision resistance
  - one-wayness still holds
- SHA2 (SHA-224, SHA-256, SHA-384, SHA-512)
  - outputs 224, 256, 384, and 512 bits, respectively
  - No real security concerns yet

Merkle-Damgard Construction for Hash Functions

- Message is divided into fixed-size blocks and padded
- Uses a compression function f, which takes a chaining variable (of size of hash output) and a message block, and outputs the next chaining variable
- Final chaining variable is the hash value

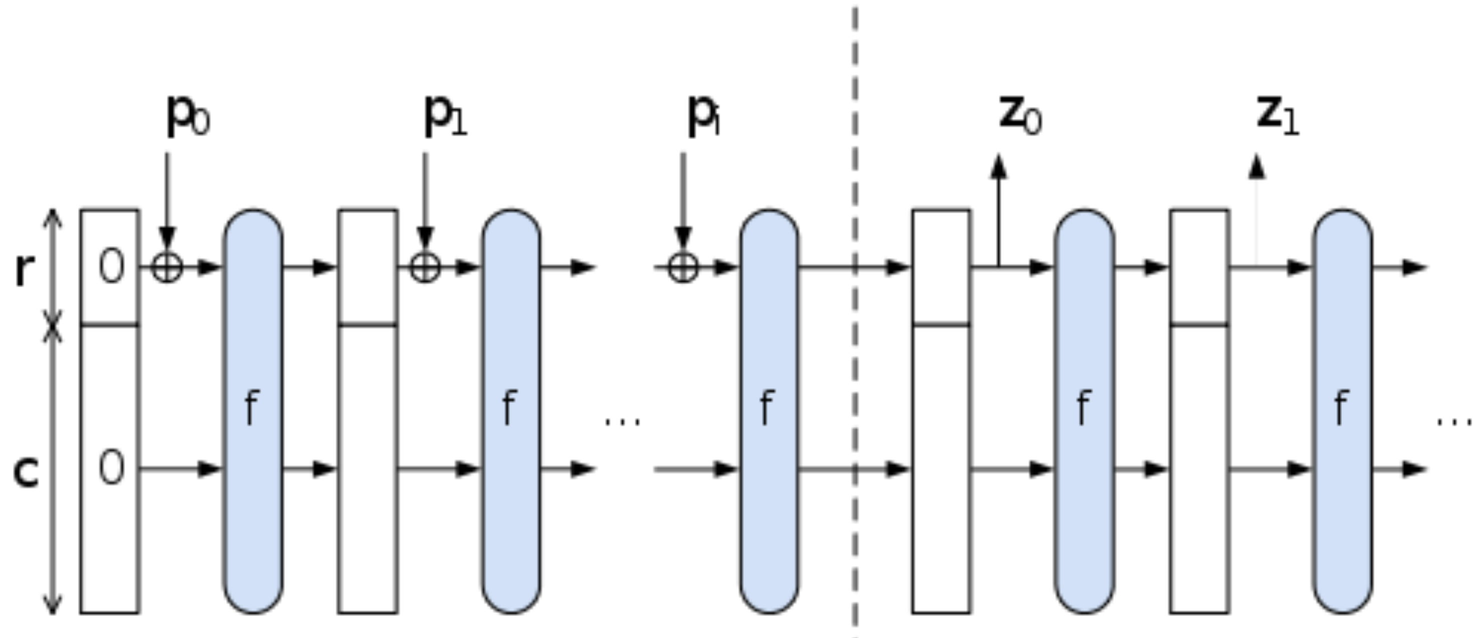


$$M=m_1m_2\ldots m_n;\ C_0=IV,\ C_{i+1}=f(C_i,m_i);\ H(M)=C_n$$

NIST SHA-3 Competition

- NIST is having an ongoing competition for SHA-3, the next generation of standard hash algorithms
- 2007: Request for submissions of new hash functions
- 2008: Submissions deadline. Received 64 entries. Announced first-round selections of 51 candidates.
- 2009: After First SHA-3 candidate conference in Feb, announced 14 Second Round Candidates in July.
- 2010: After one year public review of the algorithms, hold second SHA-3 candidate conference in Aug. Announced 5 Third-round candidates in Dec.
- 2011: Public comment for final round
- 2012: October 2, NIST selected SHA3
  - Keccak (pronounced “catch-ack”) created by Guido Bertoni, Joan Daemen and Gilles Van Assche, Michaël Peeters

The Sponge Construction: Used by SHA-3



- Each round, the next r bits of message is XOR'ed into the first r bits of the state, and a function f is applied to the state.
- After message is consumed, output r bits of each round as the hash output; continue applying f to get new states
- SHA-3 uses 1600 bits for state size

Choosing the length of Hash outputs

- The Weakest Link Principle:
  - A system is only as secure as its weakest link.
- Hence all links in a system should have similar levels of security.
- Because of the birthday attack, the length of hash outputs in general should double the key length of block ciphers
  - SHA-224 matches the 112-bit strength of triple-DES (encryption 3 times using DES)
  - SHA-256, SHA-384, SHA-512 match the new key lengths (128,192,256) in AES

for Authentication

- Require an authentic channel to transmit the hash of a message
  - Without such a channel, it is insecure, because anyone can compute the hash value of any message, as the hash function is public
  - Such a channel may not always exist
- How to address this?
  - use more than one hash functions
  - use a key to select which one to use

Hash Family

- A hash family is a four-tuple (X, Y, K, H), where
  - X is a set of possible messages
  - Y is a finite set of possible message digests
  - K is the keyspace
  - For each $K \in K$, there is a hash function $h_K \in H$. Each h_K: $X \rightarrow Y$
- Alternatively, one can think of H as a function $K \times X \rightarrow Y$

Message Authentication Code

- A MAC scheme is a hash family, used for message authentication
- $MAC(K,M) = H_K(M)$
- The sender and the receiver share secret K
- The sender sends $(M, H_k(M))$
- The receiver receives (X,Y) and verifies that $H_K(X)=Y$, if so, then accepts the message as from the sender
- To be secure, an adversary shouldn't be able to come up with (X', Y') such that $H_K(X')=Y'$.

Security Requirements for MAC

- Resist the Existential Forgery under Chosen Plaintext Attack
  - Challenger chooses a random key K
  - Adversary chooses a number of messages M_1, M_2, .., M_n, and obtains t_j=MAC(K,M_j) for 1≤j≤n
  - Adversary outputs M' and t'
  - Adversary wins if ∀j M'≠M_j, and t'=MAC(K,M')

- Basically, adversary cannot create the MAC for a message for which it hasn't seen an MAC

Constructing MAC from Hash Functions

- Let h be a one-way hash function

- MAC(K,M) = h(K || M), where || denote concatenation
  - Insecure as MAC
  - Because of the Merkle-Damgard construction for hash functions, given M and t=h(K || M), adversary can compute M' =M||Pad(M)||X and t', such that h(K||M') = t'

Cryptographic Hash Functions

$$HMAC_K[M] = Hash[(K^+ \oplus opad) \,||\, Hash[(K^+ \oplus ipad)||M)]]$$

- K^+ is the key padded (with 0) to B bytes, the input block size of the hash function
- ipad = the byte 0x36 repeated B times
- opad = the byte 0x5C repeated B times.

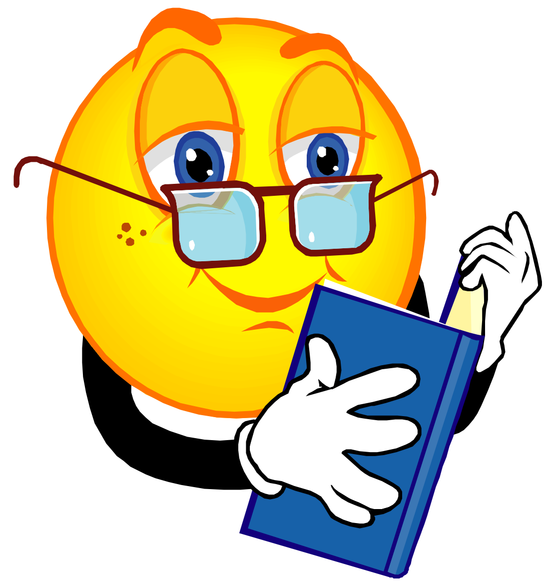
At high level, $HMAC_K[M] = H(K \,||\, H(K \,||\, M))$

HMAC Security

- If used with a secure hash functions (e.g., SHA-256) and according to the specification (key size, and use correct output), no known practical attacks against HMAC

Take home lessons

- Brute force against hash function for finding collision is $2^{m/2}$ where m is the output of the hash
- MD5 not secure as a cryptographic hash
- Recent attacks against SHA1 very strong
- The competition for a new hash function is over, SHA3 has been selected



5: Public key encryption and digital signatures

Readings for This Lecture

- Required: On Wikipedia
  - Public key cryptography
  - RSA
  - Diffie–Hellman key exchange
  - ElGamal encryption
- Required:
  - Differ & Hellman: "New Directions in Cryptography" IEEE Transactions on Information Theory, Nov 1976.

Review of Secret Key (Symmetric) Cryptography

- Confidentiality
  - stream ciphers (uses PRNG)
  - block ciphers with encryption modes
- Integrity
  - Cryptographic hash functions
  - Message authentication code (keyed hash functions)
- Limitation: sender and receiver must share the same key
  - Needs secure channel for key distribution
  - Impossible for two parties having no prior relationship
  - Needs many keys for n parties to communicate

Concept of Public Key Encryption

- Each party has a pair (K, K^{-1}) of keys:
  - K is the **public** key, and used for encryption
  - K^{-1} is the **private** key, and used for decryption
  - Satisfies $\mathbf{D}_{K^{-1}}[\mathbf{E}_K[M]] = M$
- Knowing the public-key K, it is computationally infeasible to compute the private key K^{-1}
  - How to check (K,K^{-1}) is a pair?
  - Offers only computational security. Secure PK Encryption impossible when P=NP, as deriving K^{-1} from K is in NP.
- The public-key K may be made publicly available, e.g., in a publicly available directory
  - Many can encrypt, only one can decrypt
- Public-key systems aka *asymmetric* crypto systems

Public Key Cryptography Early History

- Proposed by Diffie and Hellman, documented in "New Directions in Cryptography" (1976)
  1. Public-key encryption schemes
  2. Key distribution systems
     - Diffie-Hellman key agreement protocol
  3. Digital signature

- Public-key encryption was proposed in 1970 in a classified paper by James Ellis
  - paper made public in 1997 by the British Governmental Communications Headquarters

- Concept of digital signature is still originally due to Diffie & Hellman

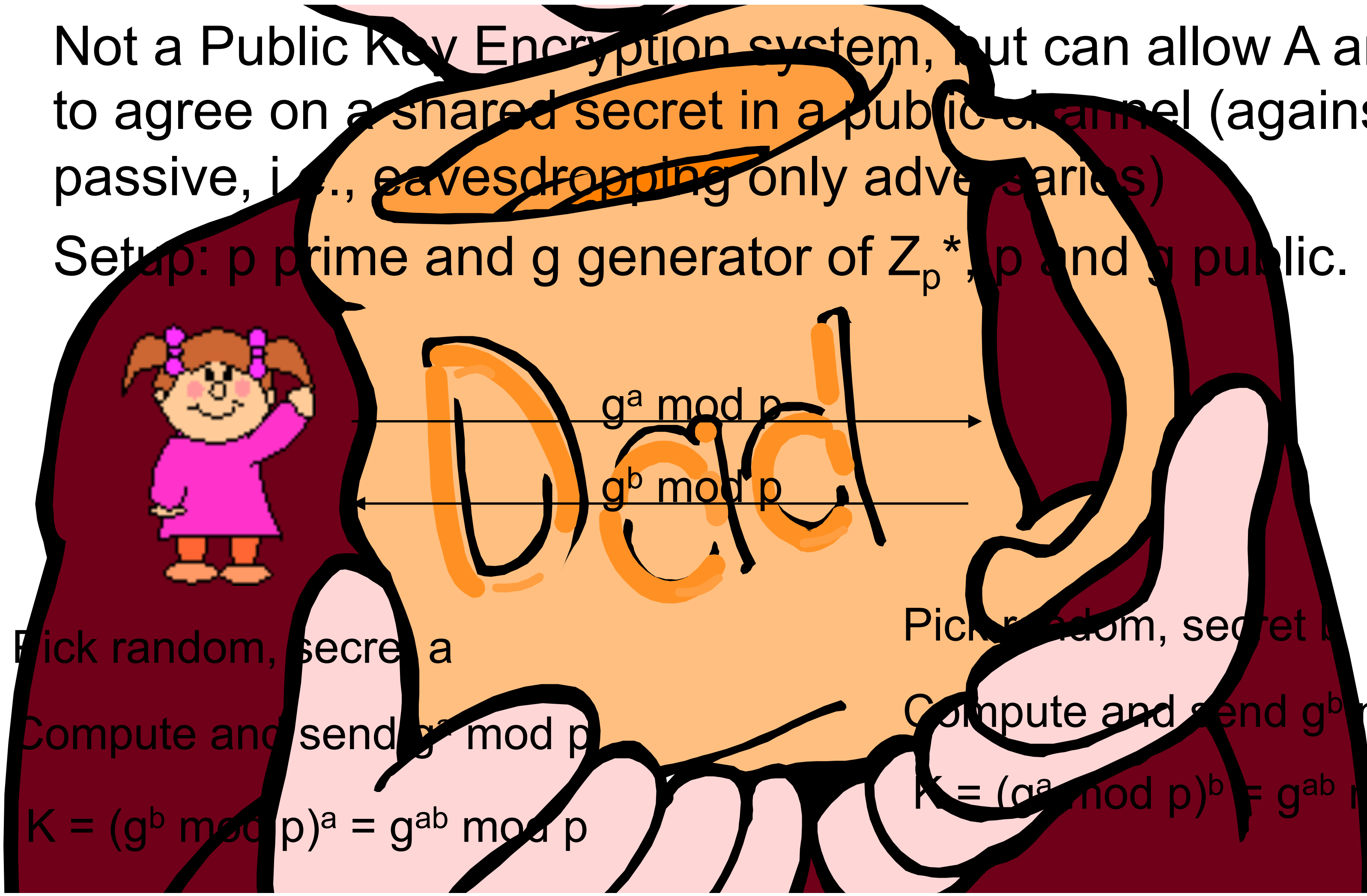
Public Key Encryption Algorithms

- Almost all public-key encryption algorithms use either number theory and modular arithmetic, or elliptic curves
- RSA
  - based on the hardness of factoring large numbers
- El Gamal
  - Based on the hardness of solving discrete logarithm
  - Use the same idea as Diffie-Hellman key agreement

Diffie-Hellman Key Agreement Protocol

Not a Public Key Encryption system, but can allow A and B to agree on a shared secret in a public channel (against passive, i.e., eavesdropping only adversaries)

Setup: p prime and g generator of Z_p^*, p and g public.



$g^a \bmod p$ →

← $g^b \bmod p$

Pick random, secret a

Compute and send $g^a \bmod p$

$K = (g^b \bmod p)^a = g^{ab} \bmod p$

Pick random, secret b

Compute and send $g^b \bmod p$

$K = (g^a \bmod p)^b = g^{ab} \bmod p$

Diffie-Hellman

- Example: Let p=11, g=2, then

a	1	2	3	4	5	6	7	8	9	10	11
g^a	2	4	8	16	32	64	128	256	512	1024	2048
g^a mod p	2	4	8	5	10	9	7	3	6	1	2

A chooses 4, B chooses 3, then shared secret is

$$(2^3)^4 = (2^4)^3 = 2^{12} = 4 \pmod{11}$$

Adversaries sees 2^3=8 and 2^4=5, needs to solve one of 2^x=8 and 2^y=5 to figure out the shared secret.

Three Problems Believed to be Hard to Solve

- Discrete Log (DLG) Problem: Given <g, h, p>, computes a such that g^a = h mod p.
- Computational Diffie Hellman (CDH) Problem: Given <g, g^a mod p, g^b mod p> (without a, b) compute g^{ab} mod p.
- Decision Diffie Hellman (DDH) Problem: distinguish (g^a,g^b,g^{ab}) from (g^a,g^b,g^c), where *a*,*b*,*c* are randomly and independently chosen

- If one can solve the DL problem, one can solve the CDH problem. If one can solve CDH, one can solve DDH.

Assumptions

- DDH Assumption: DDH is hard to solve.
- CDH Assumption: CDH is hard to solve.
- DLG Assumption: DLG is hard to solve

- DDH assumed difficult to solve for large p (e.g., at least 1024 bits).
- Warning:
  - New progress by Joux means solving discrete log for p values with some property can be done quite fast.
  - Look out when you need to use/implement public key crypto
  - May want to consider Elliptic Curve-based algorithms

ElGamal Encryption

- Public key <g, p, $h=g^a$ mod p>
- Private key is a
- To encrypt: chooses random b, computes
  C=[g^b mod p, g^{ab} * M mod p].
  - Idea: for each M, sender and receiver establish a shared secret g^{ab} via the DH protocol. The value g^{ab} hides the message M by multiplying it.
- To decrypt C=[c_1,c_2], computes M where
  - ((c_1^a mod p) * M) mod p = c_2.
    - To find M for x * M mod p = c_2, compute z s.t. x*z mod p =1, and then M = C_2*z mod p
- CDH assumption ensures M cannot be fully recovered.
- IND-CPA security requires DDH.

RSA Algorithm

- Invented in **1978** by Ron **R**ivest, Adi **S**hamir and Leonard **A**dleman
  - Published as R L Rivest, A Shamir, L Adleman, "*On Digital Signatures and Public Key Cryptosystems*", Communications of the ACM, vol 21 no 2, pp120-126, Feb 1978
- Security relies on the difficulty of factoring large composite numbers
- Essentially the same algorithm was discovered in 1973 by Clifford Cocks, who works for the British intelligence

RSA Public Key Crypto System

Key generation:

1. Select 2 large prime numbers of about the same size, p and q

   Typically each p, q has between 512 and 2048 bits

2. Compute $n = pq$, and $\Phi(n) = (q-1)(p-1)$
3. Select e, $1<e< \Phi(n)$, s.t. $\gcd(e, \Phi(n)) = 1$

   Typically e=3 or e=65537

4. Compute d, $1< d< \Phi(n)$ s.t. $ed \equiv 1 \bmod \Phi(n)$

   Knowing $\Phi(n)$, d easy to compute.

Public key: (e, n)

Private key: d

RSA Description (cont.)

Encryption

Given a message M, 0 < M < n $M \in Z_n - \{0\}$

use public key (e, n)

compute $C = M^e \bmod n$ $C \in Z_n - \{0\}$

Decryption

Given a ciphertext C, use private key (d)

Compute $C^d \bmod n = (M^e \bmod n)^d \bmod n = M^{ed} \bmod n = M$

RSA Example

- $p = 11$, $q = 7$, $n = 77$, $\Phi(n) = 60$
- $d = 13$, $e = 37$ ($ed = 481$; $ed \bmod 60 = 1$)
- Let $M = 15$. Then $C \equiv M^e \bmod n$
  - $C \equiv 15^{37} \pmod{77} = 71$
- $M \equiv C^d \bmod n$
  - $M \equiv 71^{13} \pmod{77} = 15$

RSA Example 2

- Parameters:
  - $p = 3, q = 5, n = pq = 15$
  - $\Phi(n) = ?$
- Let $e = 3$, what is d?
- Given $M=2$, what is C?
- How to decrypt?

Hard Problems RSA Security Depends on

$C = M^e \bmod (n=pq)$ →

Plaintext: M

Ciphertext: C

← $C^d \bmod n$

1. Factoring Problem: Given n=pq, compute p,q
2. Finding RSA Private Key: Given (n,e), compute d s.t. $ed = 1 \pmod{\Phi(n)}$.
   - Known to be equivalent to Factoring problem.
   - Implication: cannot share n among multiple users
3. RSA Problem: From (n,e) and C, compute M s.t. $C = M^e$
   - Aka computing the e'th root of C.
   - Can be solved if n can be factored

RSA Security and Factoring

- Security depends on the difficulty of factoring n
  - Factor n $\Rightarrow$ compute $\Phi(n)$ $\Rightarrow$ compute d from (e, n)
  - Knowing e, d such that ed = 1 (mod $\Phi(n)$) $\Rightarrow$ factor n
- The length of n=pq reflects the strength
  - 700-bit n factored in 2007
  - 768 bit factored in 2009
- RSA encryption/decryption speed is quadratic in key length
- 1024 bit for minimal level of security today
  - likely to be breakable in near future
- Minimal 2048 bits recommended for current usage
- NIST suggests 15360-bit RSA keys are equivalent in strength to 256-bit
- Factoring is easy to break with quantum computers
- Recent progress on Discrete Logarithm may make factoring much faster

RSA Encryption & IND-CPA Security

- The RSA assumption, which assumes that the RSA problem is hard to solve, ensures that the plaintext cannot be fully recovered.

- Plain RSA does not provide IND-CPA security.
  - For Public Key systems, the adversary has the public key, hence the initial training phase is unnecessary, as the adversary can encrypt any message he wants to.

  - How to break IND-CPA security?

Real World Usage of Public Key Encryption

- Often used to encrypt a symmetric key
  - To encrypt a message M under an RSA public key (n,e), generate a new AES key K, compute [K^e mod n, AES-CBC_K(M)]
- One often needs random padding.
  - Given M, chooses random r, and generates F(M,r), and then encrypts as $F(M,r)^e$ mod n
  - From F(M,r), one should be able to recover M
  - This provides randomized encryption

Digital Signatures: The Problem

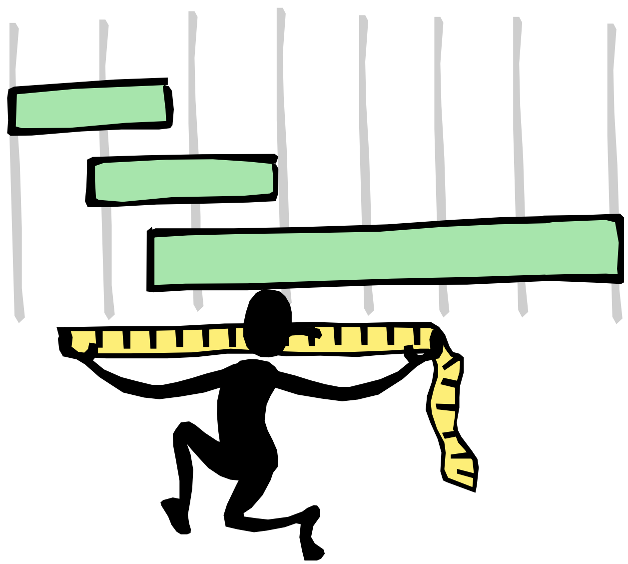
- Consider the real-life example where a person pays by credit card and signs a bill; the seller verifies that the signature on the bill is the same with the signature on the card
- Contracts are valid if they are signed.
- Signatures provide non-repudiation.
  - ensuring that a party in a dispute cannot repudiate, or refute the validity of a statement or contract.
- Can we have a similar service in the electronic world?
  - Does Message Authentication Code provide non-repudiation? Why?

Digital Signatures

- MAC: One party generates MAC, one party verifies integrity.
- Digital signatures: One party generates signature, many parties can verify.
- Digital Signature: a data string which associates a message with some originating entity.
- Digital Signature Scheme:
  - a signing algorithm: takes a message and a (private) signing key, outputs a signature
  - a verification algorithm: takes a (public) verification key, a message, and a signature
- Provides:
  - Authentication, Data integrity, Non-Repudiation

Digital Signatures and Hash

- Very often digital signatures are used with hash functions, hash of a message is signed, instead of the message.
- Hash function must be:
  - Strong collision resistant



RSA Signatures

Key generation (as in RSA encryption):

- Select 2 large prime numbers of about the same size, p and q
- Compute $n = pq$, and $\Phi = (q - 1)(p - 1)$
- Select a random integer e, $1 < e < \Phi$, s.t. $\gcd(e, \Phi) = 1$
- Compute d, $1 < d < \Phi$ s.t. $ed \equiv 1 \bmod \Phi$

Public key: (e, n) **used for verification**

Private key: d, **used for generation**

RSA Signatures with Hash (cont.)

Signing message M

- Verify $0 < M < n$
- Compute $S = h(M)^d \bmod n$

Verifying signature S

- Use public key (e, n)
- Compute $S^e \bmod n = (h(M)^d \bmod n)^e \bmod n = h(M)$

Non-repudiation

- Nonrepudiation is the assurance that someone cannot deny something. Typically, nonrepudiation refers to the ability to ensure that a party to a contract or a communication cannot deny the authenticity of their signature on a document or the sending of a message that they originated.
- Can one deny a signature one has made?
- Does email provide non-repudiation?

The Big Picture

	Secret Key Setting	Public Key Setting
Secrecy / Confidentiality	Stream ciphers Block ciphers + encryption modes	Public key encryption: RSA, El Gamal, etc.
Authenticity / Integrity	Message Authentication Code	Digital Signatures: RSA, DSA, etc.

Take home lessons

- RSA the most well-known PKI encryption and digital signature
- RSA is secure when used with proper key sizes 1024 bits and higher
- Digital certificates push entities to trust CAs
- If a CA is compromised certificates must be revoked