Introduction to Cryptography
CS 136
Computer Security
Peter Reiher
April 5, 2016

Outline

- What is data encryption?
- Cryptanalysis
- Basic encryption methods
  - Substitution ciphers
  - Permutation ciphers

Introduction to Encryption

- Much of computer security is about keeping secrets
- One method is to make the secret hard for others to read
- While (usually) making it simple for authorized parties to read

Encryption

- Encryption is the process of hiding information in plain sight
- Transform the secret data into something else
- Even if the attacker can see the transformed data, he can't understand the underlying secret

Encryption and Data Transformations

- Encryption is all about transforming the data
- One bit or byte pattern is transformed to another bit or byte pattern
- Usually in a reversible way

Encryption Terminology

- Encryption is typically described in terms of sending a message
  - Though it's used for many other purposes
- The sender is S
- The receiver is R
- And the attacker is O

More Terminology

- *Encryption* is the process of making message unreadable/unalterable by *O*
- *Decryption* is the process of making the encrypted message readable by *R*
- A system performing these transformations is a *cryptosystem*
  - Rules for transformation sometimes called a *cipher*

Plaintext and Ciphertext

- *Plaintext* is the original form of the message (often referred to as *P*)

```
Transfer
$100 to my
savings
account
```

- *Ciphertext* is the encrypted form of the message (often referred to as *C*)

```
Sqzmredq
#099 sn lx
rzuhmfr
zbbntms
```

Very Basics of Encryption Algorithms

- Most algorithms use a *key* to perform encryption and decryption
  - Referred to as *K*
- The key is a secret
- Without the key, decryption is hard
- With the key, decryption is easy

Terminology for Encryption Algorithms

- The encryption algorithm is referred to as *E()*
- $C = E(K,P)$
- The decryption algorithm is referred to as *D()*
  – Sometimes the same algorithm as *E()*
- The decryption algorithm also has a key

Symmetric and Asymmetric Encryption Systems

- Symmetric systems use the same keys for E and D :

  $P = D(K, C)$

  Expanding, $P = D(K, E(K,P))$

- Asymmetric systems use different keys for E and D:

  $C = E(K_E,P)$

  $P = D(K_D,C)$

Characteristics of Keyed Encryption Systems

- If you change only the key, a given plaintext encrypts to a different ciphertext
  - Same applies to decryption
- Decryption should be hard without knowing the key

Cryptanalysis

- The process of trying to break a cryptosystem
- Finding the meaning of an encrypted message without being given the key
- To build a strong cryptosystem, you must understand cryptanalysis

Forms of Cryptanalysis

- Analyze an encrypted message and deduce its contents
- Analyze one or more encrypted messages to find a common key
- Analyze a cryptosystem to find a fundamental flaw

Breaking Cryptosystems

- Most cryptosystems are breakable
- Some just cost more to break than others
- The job of the cryptosystem designer is to make the cost infeasible
  - Or incommensurate with the benefit extracted

Types of Attacks on Cryptosystems

- Ciphertext only
- Known plaintext
- Chosen plaintext
  - Differential cryptanalysis
- Algorithm and ciphertext
  - Timing attacks
- In many cases, the intent is to guess the key

Ciphertext Only

- No *a priore* knowledge of plaintext
- Or details of algorithm
- Must work with probability distributions, patterns of common characters, etc.
- Hardest type of attack

Known Plaintext

- Full or partial
- Cryptanalyst has matching sample of ciphertext and plaintext
- Or may know something about what ciphertext represents
  - E.g., an IP packet with its headers

Chosen Plaintext

- Cryptanalyst can submit chosen samples of plaintext to the cryptosystem
- And recover the resulting ciphertext
- Clever choices of plaintext may reveal many details
- Differential cryptanalysis iteratively uses varying plaintexts to break the cryptosystem
  - By observing effects of controlled changes in the offered plaintext

Algorithm and Ciphertext

- Cryptanalyst knows the algorithm and has a sample of ciphertext
- But not the key, and cannot get any more similar ciphertext
- Can use "exhaustive" runs of algorithm against guesses at plaintext
- Password guessers often work this way
- *Brute force attacks* – try every possible key to see which one works

Timing Attacks

- Usually assume knowledge of algorithm
- And ability to watch algorithm encrypting/decrypting
- Some algorithms perform different operations based on key values
- Watch timing to try to deduce keys
- Successful against some smart card crypto
- Similarly, observe power use by hardware while it is performing cryptography

Basic Encryption Methods

- Substitutions
  - Monoalphabetic
  - Polyalphabetic
- Permutations

Substitution Ciphers

- Substitute one or more characters in a message with one or more different characters
- Using some set of rules
- Decryption is performed by reversing the substitutions

Example of a Simple Substitution Cipher

How did this transformation happen?

```
Sqzmredq
#099 sn lx
rzuhmfr
zbbntms
```

```
Sqzmredq
#099 sn lx
rzuhmfr
zbbntms
```

Every letter was changed to the “next lower” letter

Caesar Ciphers

- A simple substitution cipher like the previous example
  - Supposedly invented by Julius Caesar
- Translate each letter a fixed number of positions in the alphabet
- Reverse by translating in opposite direction

Is the Caesar Cipher a Good Cipher?

- Well, it worked great 2000 years ago
- It's simple, but
- It's simple
- Fails to conceal many important characteristics of the message
- Which makes cryptanalysis easier
- Limited number of useful keys

How Would Cryptanalysis Attack a Caesar Cipher?

- Letter frequencies
- In English (and other alphabetic languages), some letters occur more frequently than others
- Caesar ciphers translate all occurrences of a given plaintext letter into the same ciphertext letter
- All you need is the offset

More On Frequency Distributions

- In most languages, some letters used more than others
  - In English, “e,” “t,” and “s” are common
- True even in non-natural languages
  - Certain characters appear frequently in C code
  - Zero appears often in numeric data

Cryptanalysis and Frequency Distribution

- If you know what kind of data was encrypted, you can (often) use frequency distributions to break it
- Especially for Caesar ciphers
  - And other simple substitution-based encryption algorithms

Breaking Caesar Ciphers

- Identify (or guess) the kind of data
- Count frequency of each encrypted symbol
- Match to observed frequencies of unencrypted symbols in similar plaintext
- Provides probable mapping of cipher
- The more ciphertext available, the more reliable this technique

Example

- With ciphertext “Sqzmredq #099 sn lx rzuhmfr zbbntms”
- Frequencies -

a	0	b	2	c	0	d	1	e	1
f	1	g	0	h	1	i	0	j	0
k	0	l	1	m	3	n	2	o	0
p	0	q	2	r	3	s	3	t	1
u	1	v	0	w	0	x	1	y	0
z	3								

Applying Frequencies To Our Example

a	0	b	2	c	0	d	1	e	1
f	1	g	0	h	1	i	0	j	0
k	0	l	1	m	3	n	2	o	0
p	0	q	2	r	3	s	3	t	1
u	1	v	0	w	0	x	1	y	0
z	3								

- The most common English letters are typically “e,” “t,” “a,” “o,” and “s”
- Four out of five of the common English letters in the plaintext map to these letters

Cracking the Caesar Cipher

- Since all substitutions are offset by the same amount, just need to figure out how much
- How about +1?
  - That would only work for a=>b
- How about -1?
  - That would work for t=>s, a=>z, o=>n, and s=>r
  - Try it on the whole message and see if it looks good

More Complex Substitutions

- Monoalphabetic substitutions
  - Each plaintext letter maps to a single, unique ciphertext letter
- Any mapping is permitted
- Key can provide method of determining the mapping
  - Key could <u>be</u> the mapping

Are These Monoalphabetic Ciphers Better?

- Only a little
- Finding the mapping for one character doesn't give you all mappings
- But the same simple techniques can be used to find the other mappings
- Generally insufficient for anything serious

Codes and Monoalphabetic Ciphers

- Codes are sometimes considered different than ciphers
- A series of important words or phrases are replaced with meaningless words or phrases
- E.g., “Transfer $100 to my savings account” becomes
  - “The hawk flies at midnight”

Are Codes More Secure?

- Frequency attacks based on letters don't work
- But frequency attacks based on phrases may
- And other tricks may cause problems
- In some ways, just a limited form of substitution cipher
- Weakness based on need for codebook
  - Can your codebook contain all message components?

Superencipherment

- First translate message using a code book
- Then encipher the result
- If opponent can't break the cipher, great
- If he can, he still has to break the code
- Depending on several factors, may (or may not) be better than just a cipher
- Popular during WWII (but the Allies still read Japan's and Germany's messages)

Polyalphabetic Ciphers

- Ciphers that don't always translate a given plaintext character into the same ciphertext character
- For example, use different substitutions for odd and even positions

Example of Simple Polyalphabetic Cipher

- Move one character "up" in even positions, one character "down" in odd positions
- Note that same character translates to different characters in some cases

```
Transfer
$100 to my
savings
account
```

```
Sszorgds
%019 sp nx
tbujmhr
zdbptos
```

Are Polyalphabetic Ciphers Better?

- Depends on how easy it is to determine the pattern of substitutions
- If it’s easy, then you’ve gained little

Cryptanalysis of Our Example

- Consider all even characters as one set
- And all odd characters as another set
- Apply basic cryptanalysis to each set
- The transformations fall out easily
- How did you know to do that?
  - You guessed
  - Might require several guesses to find the right pattern

How About For More Complex Patterns?

- Good if the attacker doesn't know the choices of which characters get transformed which way
- Attempt to hide patterns well
- But known methods still exist for breaking them

Methods of Attacking Polyalphabetic Ciphers

- Kasiski method tries to find repetitions of the encryption pattern
- Index of coincidence predicts the number of alphabets used to perform the encryption
- Both require lots of ciphertext

How Does the Cryptanalyst "Know" When He's Succeeded?

- Every key translates a message into something
- If a cryptanalyst thinks he's got the right key, how can he be sure?
- Usually because he doesn't get garbage when he tries it
- He almost certainly will get garbage from any other key
- Why?

Consider A Caesar Cipher

- There are 25 useful keys (in English)
- The right one will clearly yield meaningful text
- What's the chances that any of the other 24 will?
  - Pretty poor
- So if the decrypted text makes sense, you've got the key

The More General Case

- Let's say the message is N bits long
  - So there are 2^N possible messages
  - But many of those make no sense
- Let's say the key is m bits long ($m << N$)
  - So there are 2^m keys
- So each N bit encrypted message could be decrypted 2^m ways
  - But that leaves 2^{N-m} possible messages it couldn't be

Why Does That Help?

- What if only only 2^k of the possible messages make sense?
  - $2^k << 2^N$
  - That would be the case if the message was English text, e.g.
- Assuming everything is random (and a good encryption algorithm tries to be)
  - For each wrong key, the chance it decrypts to something sensible is around $2^k/2^N = 1/2^{N-k}$
  - The chance any of the other m-1 keys give sensible output is thus $(2^m-1) * 1/2^{N-k} \sim= 1/2^{N-k+m}$

The Unbreakable Cipher

- There is a “perfect” substitution cipher
- One that is theoretically (and practically) unbreakable without the key
- And you can’t guess the key
  - If the key was chosen in the right way . . .

One-Time Pads

- Essentially, use a new substitution alphabet for <u>every</u> character
- Substitution alphabets chosen purely at random
  - These constitute the key
- Provably unbreakable without knowing this key

Example of One Time Pads

- Usually explained with bits, not characters
- We shall use a highly complex cryptographic transformation:
  - XOR
- And a three bit message
  - 010

One Time Pads at Work

0	1	0

Flip some coins to get random numbers

Apply our sophisticated cryptographic algorithm

0	0	1

0	1	1

We now have an unbreakable cryptographic message

What's So Secure About That?

- Any key was equally likely
- Any plaintext could have produced this message with one of those keys
- Let's look at our example more closely

Why Is the Message Secure?

Let's say there are only two possible meaningful messages

0 1 1

There's a key that works for each

And they're equally likely

0 1 0 | 0 0 1

Could the message decrypt to either or both of these?

0 0 0 | 0 1 1

Security of One-Time Pads

- If the key is truly random, provable that it can't be broken without the key
- But there are problems
- Need one bit of key per bit of message
- Key distribution is painful
- Synchronization of keys is vital
- A good random number generator is hard to find

One-Time Pads and Cryptographic Snake Oil

- Companies regularly claim they have “unbreakable” cryptography
- Usually based on one-time pads
- But typically misused
  - Pads distributed with some other crypto mechanism
  - Pads generated with non-random process
  - Pads reused

Permutation Ciphers

- Instead of substituting different characters, scramble up the existing characters
- Use algorithm based on the key to control how they're scrambled
- Decryption uses key to unscramble

Characteristics of Permutation Ciphers

- Doesn't change the characters in the message
  - Just where they occur
- Thus, character frequency analysis doesn't help cryptanalyst

Columnar Transpositions

- Write the message characters in a series of columns
- Copy from top to bottom of first column, then second, etc.

Example of Columnar Substitution

How did this transformation happen?

T	r	a	n	s	f
e	r		$	1	0
0		t	o		m
y		s	a	v	i
n	g	s		a	c
c	o	u	n	t	

T	e	0	y	n	c
r	r			g	o
a		t	s	s	u
n	$	o	a		n
s	1		v	a	t
f	0	m	i	c	

Looks a lot more cryptic written this way:

Te0yncrr goa tssun$oa ns1 vatf0mic

Attacking Columnar Transformations

- The trick is figuring out how many columns were used
- Use information about digrams, trigrams, and other patterns
- Digrams are pairs of letters that frequently occur together (“re”, “th”, “en”, e.g.)
- For each possibility, check digram frequency

For Example,

4 5 6 1 2 3 4 5 6 1 2 3 4 5 6 1 2 3

Te0yncrr goa tssun$oa ns1 vatf0mic

$ 1 0 0

In our case, the presence of dollar signs and numerals in the text is suspicious

- Maybe they belong together?

Umm, maybe there's 6 columns?

Double Transpositions

- Do it twice
- Using different numbers of columns
- How do you break it?
  - Find pairs of letters that probably appeared together in the plaintext
  - Figure out what transformations would put them in their positions in the ciphertext
- Can transform more than twice, if you want

Generalized Transpositions

- Any algorithm can be used to scramble the text
- Usually somehow controlled by a key
- Generality of possible transpositions makes cryptanalysis harder

Which Is Better, Transposition or Substitution?

- Well, neither, really
- Strong modern ciphers tend to use both
- Transposition scrambles text patterns
- Substitution hides underlying text characters/bits
- Combining them can achieve both effects
  - If you do it right . . .

Quantum Cryptography

- Using quantum mechanics to perform crypto
  - Mostly for key exchange
- Rely on quantum indeterminacy or quantum entanglement
- Existing implementations rely on assumptions
  - Quantum hacks have attacked those assumptions
- Not ready for real-world use, yet
- Quantum computing (to attack crypto) even further off