Math 7H		Professor: Padraic Bartlett
	Lecture 3: Cryptography	
Week 3		*UCSB 2015*

1 Cryptography: Classical Approaches

Definition. An **encryption algorithm**, or **cipher**, is a method that allows us to turn normal, plainly-readable text into difficult-to-read **ciphertext**, via the use of a secret key. Formally, we write

$$enc(k, \text{plaintext}) = \text{ciphertext},$$
$$dec(k, \text{ciphertext}) = \text{plaintext},$$

where

- plaintext is a message we want to encrypt,
- k is a key,
- *enc* is a function that takes in a unencrypted message and a key, and outputs an encrypted version of that message, and
- *dec* is a function that takes in an encrypted message and a key, and outputs the unencrypted version of that message.

In this system, the functions enc, dec are assumed to be widely known. There are cryptographic systems that do not assume this, but they are widely considered to be "bad." There are a number of reasons for why this is believed to be true (see security through obscurity and Kerckhoffs's principle for a wider discussion of this philosophy;) one of the simpler arguments is that there are many more ways to determine what an algorithm does (intimidate/bribe any of a number of engineers, steal a copy of the code, pose as a legitimate user and buy the algorithm, etc.) than to steal a specific key (which can only be done by attacking the specific user of the key you want.)

Typically, in cryptography, we refer to the two communicating parties as Alice and Bob (A and B for short, but seriously everyone uses Alice and Bob,) and the hypothetical third-party eavesdropper as Eve (E.) Given any encryption system, we typically evaluate its strength by looking at it in the following three situations:

1. The attacker, Eve, has access to a number of cipher texts.
2. The attacker, Eve, has access to a number of cipher texts and their original plaintexts.
3. The attacker, Eve, has the ability to generate cipher texts for whatever plaintext inputs they choose.

The first situation is the most common one: it is typically assumed that the attacker Eve has access to most, if not all, of the encrypted traffic that Alice and Bob send back and forth to each other. The second is stronger, but not unreasonable to expect; in many situations (see: WWII and the Enigma machine for some great stories) the attacker will be able to "steal" some unencrypted messages via subterfuge, and it would be great if an encryption method could still work even with this occasionally happening. The third is stronger yet: it basically is a kind of system that can withstand most anything, as long as the keys remain hidden. Strong cryptographical systems work in all of these systems, and are what we want to find.

People have been creating encryption schemes for thousands of years; essentially, as long as there have been people with secrets to keep, there have been ways to keep things secret. We study several of these here:

Algorithm. The Caesar-shift cipher. The Caesar-shift cipher, whose first recorded use was by Julius Caesar to protect various military secrets, is the following encryption scheme. Given a plaintext message m and a key k, "encrypt" m by doing the following: one-by-one, take each character of m and circularly shift it over k places to the right in the alphabet. Caesar historically used this cipher with a shift of three: i.e. $A \mapsto D$, $B \mapsto E$, $\ldots W \mapsto Z$, $X \mapsto A$, $Y \mapsto B$, $Z \mapsto C$. The decryption scheme is similar: take your encrypted message and character-by-character, circularly shift each letter over k places to the left in the alphabet.

This cipher, with a key of 13, is known as "ROT13" and is frequently used on the Internet to hide spoilers; this cipher-key combo is particularly convenient, because its encryption and decryption functions are identical (shifting right by 13 and left by 13 are the same in a 26-character alphabet.)

Example. Take the message[1]

```
"Just the place for a Snark!" the Bellman cried,
As he landed his crew with care;
Supporting each man on the top of the tide
By a finger entwined in his hair.
```

If we applied a Caesar shift with key 4, we would get the message

```
"Nywx xli tpegi jsv e Wrevo!" xli Fippqer gvmih,
Ew li perhih lmw gvia amxl gevi;
Wyttsvxmrk iegl qer sr xli xst sj xli xmhi
Fc e jmrkiv irxamrih mr lmw lemv.
```

Weaknesses. This is a very weak cipher. In particular, it is easy to beat with brute-force approaches: for example, suppose we saw the text

```
Ns ymj gjlnssnsl ymjwj bfx stymnsl, bmnhm jcuqtiji.
```

we could simply just go through values of k until we got something that looked like a promising translation:

[1]From **The Hunting of the Snark**, by Lewis Carroll. It's pretty great.

```
Mr xli fikmrrmrk xlivi aew rsxlmrk, almgl ibtpshih.
Lq wkh ehjlqqlqj wkhuh zdv qrwklqj, zklfk hasorghg.
Kp vjg dgikppkpi vjgtg ycu pqvjkpi, yjkej gzrnqfgf.
Jo uif cfhjoojoh uifsf xbt opuijoh, xijdi fyqmpefe.
In the beginning there was nothing, which exploded.
```

Given that there are only 26 values of k to pick, this should be something we can do relatively quickly. Moreover, we could just do this to a small sample of text if it took us too long to translate everything, as there's usually only one shift that's going to make our text look readable.

Algorithm. Simple substition ciphers. One key issue with the algorithm above was that the range of possible key choices was far too small: we could simply adopt a brute-force approach and look at all possible outputs of our decryption function under different keys, and discover the original plaintext in this way.

A solution to this was the idea of a simple substition cipher, which is defined as follows: first, write down the alphabet. Then, write down some permutation ρ of that alphabet: i.e.

`ABCDEFGHIJKLMNOPQRSTUVWXYZ`

This permutation ρ is the key for an encryption scheme defined as follows: take a plaintext message, and character-by-character replace each letter in the plaintext message with a character from the permutation. For example, if we used the permutation described above, we would have $A \mapsto E, B \mapsto J, C \mapsto W, D \mapsto D, \ldots$

This algorithm avoids the weakness we've noticed that Caesar-shift is weak to: where the Caesar shift only had 26 keys, this algorithm has as many keys as there are ways to permute the characters of the alphabet. If we count, we can see that there are 26! ways in which to do this: this is because in creating a permutation we are choosing one of our 26 characters for A to map to, any of the remaining 25 characters for B to map to, and so on/so forth until we have have one last character for Z to map to.

26! is a much larger search space than 26: it's roughly $4 \cdot 10^{26}$. By comparison, the fastest supercomputer on record (as of November, 2013, and as far as I know) can perform roughly $34 * 10^{15}$ calculations per second; if it could check whether one given permutation was a viable interpretation of our encrypted text per calculation, it would require about 373 years to test all possible calculations. So, at the least, brute-force is not always the *best* strategy to use...

Example. Under the permutation

`ABCDEFGHIJKLMNOPQRSTUVWXYZ`

the phrase

is transformed into the phrase

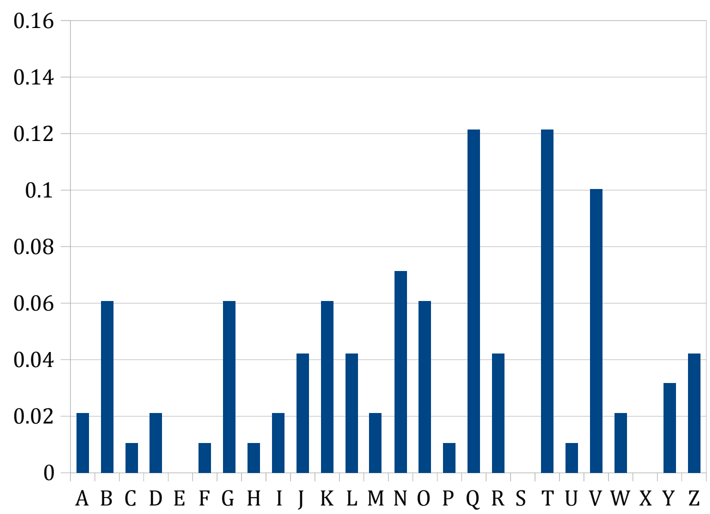
Weaknesses. While this algorithm is pretty much immune to brute-force attacks, it is still remarkably easy to crack, even by hand. We can do this by studying the underlying structure of the plaintext message sent — i.e. the structure of the English language itself! — and use this information to crack the algorithm.

To get an idea of how this would work, consider the following longer sample of ciphertext:

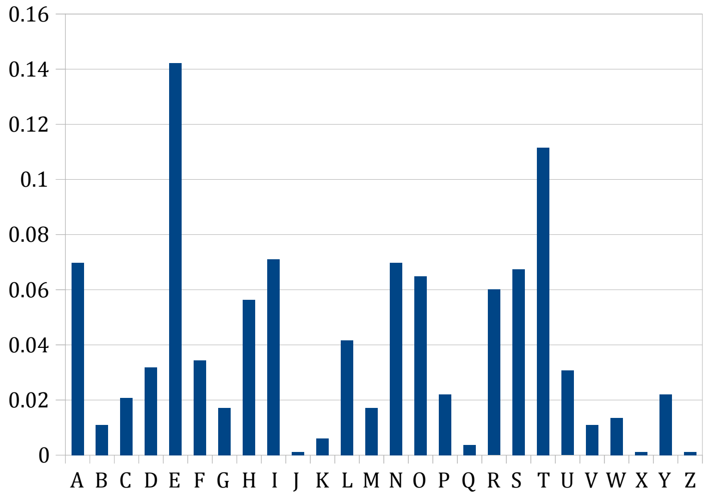
```
Hvni tobnwk mqnt gn vt Ynkqqn Ynbuqjkbti vnr, jqwjqttvdli,
tqvzobnw ovr tg dq gnq ga toqh.  Toq avzylti ovr lgnw vwg
zgnajgntqr tobk avzt vnr ovr cqjaqztqr uvjbgyk rqubzqk agj
vugbrbnw bt.  Dyt tobk mvk cqjaqztli vll jbwot dqzvykq, tg
dq avbj, kg ovr toq ktyrqntk.  Toq kiktqh mgjkqr fybtq mqll
vnr, vk ovccqnk bn kyzo zvkqk, ovr tvkqn gn toq ktvtyk ga
tjvrbtbgn.  Lqztyjqk zlqvjli tggk clvzq, dqzvykq toqi mqjq
rgmn toqjq gn toq tbhqtvdlq bn dlvzk vnr mobtq.  Toq avzt tovt
ng-gnq vttqnrqr mvk vn bjjqlquvnt rqtvbl.  Bt mvk gzzvkbgnvlli
hvbntvbnqr tovt tobk hqvnt tovt toq lqztyjqk rbr ngt bn avzt
ovccqn vt vll, dyt ng-gnq quqj vttqnrqr toqh tg abnr gyt
ba tobk mvk tjyq.  Vnimvi, bt mvk vjwyqr (di toq Jqvrqj bn
Mgglli Tobnkbnw  mobzo bk lbkq Ayppi Lgwbz, gnli lqkk kg) tovt
lqztyjqk ovr tvkqn clvzq bn qkkqnzq, kg tovt mvk vll jbwot,
tgg.
```

On its surface, this doesn't look much like English anymore. However, this doesn't stop us from a more fundamental method of attack — character frequency!

Specifically: notice that the characters in the above text occur with the following frequencies:



Now, notice that if you were to take a sufficiently large sample of works in English — say, a handful of Iain M. Banks novels, or the works of Raymond Chandler — you also have certain character frequencies! Intuitively, this makes sense: we're much more likely to see the letter “a” than the letter “q,” for example. In general, English text tends to have the following character distributions:



This gives us the following observations:

- The most frequently occurring characters in our ciphertext are the characters t (12%,) q (10%,) v (7%,) n, k, o and g (all at 6%.) Conversely, the most commonly-occurring characters in the English language are (in order) e (13%,) t (9%,) a (8%,) o (8%,) i and n(all at 7%). Consequently, it is likely that most of our frequently-occurring characters in our ciphertext correspond to these common characters in English!
- Moreover, we can scan our text for commonly-occurring three-letter strings. We see the following strings occurring multiple times in our text:

  1. toq (12 times)
  2. tob (6 times)
  3. tov (6 times)
  4. mvk (6 times)
  5. ovr (6 times)
  6. qzt (5 times)
  7. tto (5 times)
  8. obk (5 times)
  9. ovt (5 times)
  10. vtt (4 times)
  11. vll (4 times)
  12. rto (4 times)
  13. bmw (4 times)
  14. vnr (4 times)
  15. vrt (4 times)
  16. avz (4 times)

  Consequently, if we looked through our text and picked out the most frequently-occurring three-letter cipherphrase, “toq”, we could assume that this matches up with one of the most frequently-occurring three-letter objects in English! Again, studying a large body of work reveals that the most frequently occurring trigrams in English, in order, are

  1. the (3.51%)
  2. and (1.59%)
  3. ing (1.14%)
  4. her (0.82%)
  5. hat (0.65%)
  6. his (0.59%)
  7. tha (0.59%)
  8. ere (0.56%)
  9. for (0.55%)
  10. ent (0.53%)
  11. ion (0.51%)
  12. ter (0.46%)

13. was (0.46%)
14. you (0.43%)
15. ith (0.43%)
16. ver (0.43%)
17. all (0.42%)
18. wit (0.40%)
19. thi (0.39%)
20. tio (0.38%)

Of these, “the” is by far and away the most common. So, it is fairly likely that “toq” in our ciphertext corresponds to “the” in English! This is backed up by looking at frequencies of individual characters; t and q are the two most common characters in our ciphertext, while t and e are the two most common characters in English.

- If we apply this observation — that it is likely that $T \mapsto T, O \mapsto H$, and $Q \mapsto E$ — we get

```
Hvni thbnwk ment gn vt Ynkeen Ynbuejkbti vnr, jewjettvdli,
tevzhbnw hvr tg de gne ga theh.  The avzylti hvr lgnw vwg
zgnajgnter thbk avzt vnr hvr cejaezter uvjbgyk reubzek agj
vugbrbnw bt.  Dyt thbk mvk cejaeztli vll jbwht dezvyke, tg
de avbj, kg hvr the ktyrentk.  The kikteh mgjker fybte mell
vnr, vk hvccenk bn kyzh zvkek, hvr tvken gn the ktvtyk ga
tjvrbtbgn.  Leztyjek zlevjli tggk clvze, dezvyke thei meje
rgmn theje gn the tbhetvdle bn dlvzk vnr mhbte.  The avzt thvt
ng-gne vttenrer mvk vn bjjeleuvnt retvbl.  Bt mvk gzzvkbgnvlli
hvbntvbner thvt thbk hevnt thvt the leztyjek rbr ngt bn avzt
hvccen vt vll, dyt ng-gne euej vttenrer theh tg abnr gyt
ba thbk mvk tjye.  Vnimvi, bt mvk vjwyer (di the Jevrej bn
Mgglli Thbnkbnw  mhbzh bk lbke Ayppi Lgwbz, gnli lekk kg) thvt
leztyjek hvr tvken clvze bn ekkenze, kg thvt mvk vll jbwht,
tgg.
```

(The symbols we “guessed” are in red.

- This guess lets us make more guesses. For example, we can see lots of repeated two-letter words of the form “tg”. There is only one common two-letter English word that starts with a t: “to.” So, it is a reasonable guess that $G \mapsto O$, and therefore that

```
Hvni thbnwk ment on vt Ynkeen Ynbuejkbti vnr, jewjettvdli,
tevzhbnw hvr to de one oa theh.  The avzylti hvr lonw vwo
zonajonter thbk avzt vnr hvr cejaezter uvjboyk reubzek aoj
vuobrbnw bt.  Dyt thbk mvk cejaeztli vll jbwht dezvyke, to
de avbj, ko hvr the ktyrentk.  The kikteh mojker fybte mell
vnr, vk hvccenk bn kyzh zvkek, hvr tvken on the ktvtyk oa
tjvrbtbon.  Leztyjek zlevjli took clvze, dezvyke thei meje
romn theje on the tbhetvdle bn dlvzk vnr mhbte.  The avzt thvt
no-one vttenrer mvk vn bjjeleuvnt retvbl.  Bt mvk ozzvkbonvlli
hvbntvbner thvt thbk hevnt thvt the leztyjek rbr not bn avzt
hvccen vt vll, dyt no-one euej vttenrer theh to abnr oyt
ba thbk mvk tjye.  Vnimvi, bt mvk vjwyer (di the Jevrej bn
Moolli Thbnkbnw  mhbzh bk lbke Ayppi Lowbz, onli lekk ko) thvt
leztyjek hvr tvken clvze bn ekkenze, ko thvt mvk vll jbwht,
too.
```

- We can continue in this fashion; for example, we might look at the multiple "thvt" instances, and decide that the character "v" is likely to correspond to "a", as that's the only vowel that makes this a word.

```
Hani thbnwk ment on at Ynkeen Ynbuejkbti anr, jewjettadli,
teazhbnw har to de one oa theh.  The aazylti har lonw awo
zonajonter thbk aazt anr har cejaezter uajboyk reubzek aoj
auobrbnw bt.  Dyt thbk mak cejaeztli all jbwht dezayke, to
de aabj, ko har the ktyrentk.  The kikteh mojker fybte mell
anr, ak haccenk bn kyzh zakek, har taken on the ktatyk oa
tjarbtbon.  Leztyjek zleajli took claze, dezayke thei meje
romn theje on the tbhetadle bn dlazk anr mhbte.  The aazt that
no-one attenrer mak an bjjeleuant retabl.  Bt mak ozzakbonalli
habntabner that thbk heant that the leztyjek rbr not bn aazt
haccen at all, dyt no-one euej attenrer theh to abnr oyt
ba thbk mak tjye.  Animai, bt mak ajwyer (di the Jearej bn
Moolli Thbnkbnw  mhbzh bk lbke Ayppi Lowbz, onli lekk ko) that
leztyjek har taken claze bn ekkenze, ko that mak all jbwht,
too.
```

- We are nearly done. We can now start exploiting the longer words in our sentence: for instance, consider the word "attenrer." If you search for "atte**e*" on onelook.com, you'll come across only one word that has identical third-to-last and last characters: "attended."

  Therefore, we can guess that $N \mapsto N$ and $R \mapsto D$.

```
Hani thbnwk ment on at Ynkeen Ynbuejkbti and, jewjettadli,
teazhbnw had to de one oa theh.  The aazylti had lonw awo
zonajonted thbk aazt and had cejaezted uajboyk deubzek aoj
auobdbnw bt.  Dyt thbk mak cejaeztli all jbwht dezayke, to
de aabj, ko had the ktydentk.  The kikteh mojked fybte mell
and, ak haccenk bn kyzh zakek, had taken on the ktatyk oa
tjadbtbon.  Leztyjek zleajli took claze, dezayke thei meje
domn theje on the tbhetadle bn dlazk and mhbte.  The aazt that
no-one attended mak an bjjeleuant detabl.  Bt mak ozzakbonalli
habntabned that thbk heant that the leztyjek dbd not bn aazt
haccen at all, dyt no-one euej attended theh to abnd oyt
ba thbk mak tjye.  Animai, bt mak ajwyed (di the Jeadej bn
Moolli Thbnkbnw  mhbzh bk lbke Ayppi Lowbz, onli lekk ko) that
leztyjek had taken claze bn ekkenze, ko that mak all jbwht,
too.
```

  Repeating this process a few more times ("Jeadej" should be "reader," so $J \mapsto R$"; "habntabned" should map to "maintained," so $H \mapsto M$, $B \mapsto I$; "haccen" maps to "happen", so $C \mapsto p$; etc) will eventually give us the following as our source text:

```
Many things went on at Unseen University and, regrettably,
teaching had to be one of them.  The faculty had long ago
confronted this fact and had perfected various devices for
avoiding it.  But this was perfectly all right because, to
be fair, so had the students.  The system worked quite well
and, as happens in such cases, had taken on the status of
tradition.  Lectures clearly took place, because they were
down there on the timetable in black and white.  The fact that
no-one attended was an irrelevant detail.  It was occasionally
maintained that this meant that the lectures did not in fact
happen at all, but no-one ever attended them to find out if
this was true.  Anyway, it was argued (by the Reader in Woolly
Thinking --- which is like Fuzzy Logic, only less so) that
lectures had taken place in essence, so that was all right,
too.
```

Cryptography, now with Terry Pratchett![2]

Some of the weaknesses in this algorithm can be fixed, as we illustrate below:

Algorithm. Vigenère ciphers. In a sense, the main weakness of the simple substitution cipher is that each plaintext letter always corresponded to the same encrypted symbol: this allowed us to use our knowledge of English to break the code. We can fix this, however, by using a Vigenère cipher!

Specifically: the key to a Vigenère cipher is a code word k that is some string of characters. To illustrate, suppose that the code word is "bah." To encrypt some message, like (for example) "Friendship is Magic!," we use the code word as a way to "shift" the letters of the codephrase as follows:

1. Write down your message. Below it, write a number of copies of the codeword so that each character in our message is matched up with a character from the codeword, as below:

```
Friendship is Magic!
bahbahbahb ah bahba
```

2. Then, take each character in the message, and "shift" it by the codeword character corresponding to it! In other words, take each codeword character, interpret it as a number (i.e. $a \mapsto 1, b \mapsto 2, \ldots$), and circularly shift the message character to the right that many places. For example, our message becomes

```
Hsqgoluiqr ja Obokd!
```

Example. We provide another example here, this one a bit more long-form. Consider the codeword "bode," and the message to be encoded[3]

[2] This quote of his is not a wholly accurate sentiment for your UCSB teachers. We like teaching you! Usually.

[3] **The Wasp**, a poem by Ogden Nash. He's great.

```
The wasp and all his numerous family
I look upon as a major calamity.
He throws open his nest with prodigality,
But I distrust his waspitality.
```

In this setting, we would form the four lines

```
The wasp and all his numerous family
bod ebod ebo deb ode bodebode bodebo
I look upon as a major calamity.
d ebod ebod eb o debod ebodebod
He throws open his nest with prodigality,
eb odebod ebod ebo debo debo debodebodeb
But I distrust his waspitality.
ode b odebodeb ode bodebodebod
```

which results in the text

```
Vwi bcht fps eqn wmx pjqjtdyx hpqnnn
M qqdo zrdr fu p qfldv hcaerkic.
Mg ilwqlw trtr mkh rjui anvw twqsmlcamya,
Qyy K smxvgyxv wmx ypwukieqkic.
```

Weaknesses. This algorithm, given a sufficiently long codephrase, can avoid all of the issues that came up with our earlier work: given a codephrase that's like a few sentences long, then we would expect any given letter in our message to be shifted by many different values, and therefore that analyzing the character distributions will be useless.

And this is true! However, it is still weak to attacks that exploit the underlying structure of the English language. Specifically, there is a technique, called the **Kasiski test**, that we can use to break this code.

Roughly speaking, the Kasiski test is centered around the following observations:

- English has a lot of repeated sets of characters. We used this structure to great success in our earlier problem: we broke our earlier code largely by looking for repeated triples of characters and matching them up to known English characters.
- It is likely that our source message, being originally some large English text, has a number of often-repeated sequences. While it is possible that not all of those repeats will be preserved by the Vigenère cipher, (i.e. in our earlier text sample, the triple "him" occurred above the triple "ebo" the second time and "ode" the third time,) it is certainly likely that **some** triples will line up nicely with our code word, and therefore still occur as triples (for example, the first and second occurrences of "him" are matched with the same triple "ode.")
- Why do we care about these repeated phrases in our encrypted text? Well: if they correspond to repeated phrases in the original text (and aren't there just by accident,) then they tell us something about our codeword! In particular, they tell us that if our codeword sent those two repeated phrases to the same phrases, then both of those phrases were lined up over the "same" portion of our codeword!

- Therefore, there must be a whole number of copies of the codeword separating these two phrases, in order for them to line up! For example, using this observation on our text above with the repeated "his" phrase, if we count the number of letters between the first occurrence of "his" and the second occurrence of "his," we get 88. This tells us that it is likely that our codephrase is a divisor of 88; i.e. one of 2,4,8 or 11. Further analysis, by looking for more of these repeated sets, can eliminate other options, and tell us the precise length of the codeword we're studying.

- From there, we simply need to find the elements of the codeword! We can do this just like we did for the simple substitution cipher, basically. To be specific: break our cipher text into groups, each corresponding to the character of the code word that translated it. I.e. if we had the text from our earlier example and had deduced that the code word was length 4, we could then simply group our cipher text into four groups, each corresponding to the characters in the ciphertext that were shifted by a fixed codeletter.

  On each of these groups, we can then perform the character analysis we did before, and deduce one-by-one the codeword's characters. We omit a worked example here, but it is entirely within the reader's powers to solve one in the HW!

Next week, we'll talk about how to create a "better" encryption scheme, and what some of the encryption schemes used today are!