Admin and The Shift Cipher

lecture 01
Welcome!

- Crypto is amazing
  - Can do things that initially seem impossible

- Crypto is important
  - It impacts us every day

- Crypto is fun!
  - Deep theory
  - Attackers’ mindset
Necessary administrative stuff

- Course webpage:

  - Prerequisites/information posted there
  - Syllabus posted there
  - HWs posted there
  - Announcements posted there
  - Midterm already scheduled- Oct 29 in class.
Necessary administrative stuff

- Canvas/ELMS or Gradescope (still working that out)
  - Used only to submit homework electronically-Must be Typed
  - Let me know if unable to access

- Piazza
  - Useful for discussions/questions
  - Preferable to email if you think others will have the same question
TAs

- Nathan Grammel
- Jeremy Klein
- Dan McVicker
- Jacob Prinz
- Jake Yamada
What You Need For This Class

- Mathematical prerequisites
  - Discrete math, probability, modular arithmetic
- Requires mathematical maturity
  - Proofs, abstraction
What You Need For This Class

- CS prerequisites
  - Binary, hex, pseudocode, algorithms, big-O notation

- Programming assignments
  - Hard part should not be the programming, but the thought behind it
  - Flexibility in choice of language
How to Get the Most Out of This Class

1. Read textbook and/or slides before class
   Note: On Slide Website it says on some line
   WHAT IS BELOW IS STILL A WORK IN PROGRESS.
   Should not read slides that are below that line.

2. Ask questions on Piazza and/or bring questions to class

3. This course will be taped so can catch up or review. Caution:
   3.1 If cut class and DO watch videos in sync, fine.
   3.2 If cut class and INTEND to watch videos insync, not fine.
HWs/exams

- HWs most weeks.
- Due Monday on before class begins.
- **Sick Cat Policy:** Can post Wed before class without penalty.
- **WARNING:** YOU have already been given an extension, HW solutions will be posted on Wed, so NO extensions past that.
- We will keep track of your lateness NOT for grade, but for letters.
- In-class midterm and final
Textbook


Can buy on Amazon used.

Don’t tell Katz I said so.
Laptops/electronics

- No laptops/electronics policy
  - Distracting to you
  - Distracting to others

- If you feel you need an exception, talk to me
How to contact Prof or TAs

▶ Prof email: gasarch@cs.umd.edu

▶ Please put “CMSC456” in subject line

▶ Prof Office hours MW 12-2, 3:30-5:00 or by Appt.

▶ Prof around a lot outside of office hours, feel free to drop in, but he will feel free to say Sorry, I’m busy.

▶ TA’s - email and office hours on syllabus.
Course goals

- Understand real-world crypto via a rigorous approach
- When you encounter crypto in your career
  - Understand the key terms
  - Understand the security guarantees provided
  - Know how to use crypto
  - Understand what goes on “under the hood”
- “Crypto mindset”
Course non-goals

▶ Designing your own crypto-schemes

▶ Implementing your own crypto for real-world use

▶ Course goal: Realize when to consult an expert!
A Personal Note

This is a theory course much of what we do has *direct* application!
A Personal Note

This is a theory course much of what we do has direct application!
I do not mind that, but I am not used to that.
A Personal Note

This is a theory course much of what we do has direct application! I do not mind that, but I am not used to that.

Last spring I taught
A Personal Note

This is a theory course much of what we do has direct application! I do not mind that, but I am not used to that.

Last spring I taught
CMSC 452: Elementary Theory of Computation

 taught what computer CAN’T do. Indirect applications.

And also
A Personal Note

This is a theory course much of what we do has direct application!
I do not mind that, but I am not used to that.

Last spring I taught
CMSC 452: Elementary Theory of Computation
which computer CAN'T do. Indirect applications.

And also
CMSC 858R: Ramsey Theory and its "Applications"
There were applications
A Personal Note

This is a theory course much of what we do has direct application!
I do not mind that, but I am not used to that.

Last spring I taught

CMSC 452: Elementary Theory of Computation

taught what computer CAN"T do. Indirect applications.

And also

CMSC 858R: Ramsey Theory and its “Applications”

There were applications to other parts of pure mathematics.
Classical VS Modern cryptography

Classical: (1900 BCE?–1975)

2. WW II: They used people good at crossword puzzles.
3. Turing and others brought math into it, but not much math compared to Modern

Modern: (1976-today)

1. Lots of Math. Lots of Rigor.
2. The notion of Provably Secure important.

Note: The cutoff of 1975–1976 is approximate.
## Rough course outline

<table>
<thead>
<tr>
<th></th>
<th><strong>Secrecy</strong></th>
<th><strong>Integrity</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Private-key</strong></td>
<td>Private-key encryption</td>
<td>Message authentication codes</td>
</tr>
<tr>
<td><strong>setting</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Public-key</strong></td>
<td>Public-key encryption</td>
<td>Digital signatures</td>
</tr>
<tr>
<td><strong>setting</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Classical Cryptography

lecture 01
Motivation

- Allows us to “ease into things. . . ,”
- Shows why unprincipled approaches are dangerous (unprincipled means not-rigorous, not immoral)
- Illustrates why things are more difficult than they may appear
Alice, Bob, and Eve

- Alice sends a message to Bob in code.
- Eve overhears it.
- We want Eve to not be able to decode it.

This can mean one of two things:
- Eve does not have enough information to decode it. So even if Eve had unlimited computing power she could not decode.
- Assuming Eve can’t Factor quickly (or some other function) then Eve cannot break the code.
The First Step in Any Cipher-Spaces

I want to encode

Cryptography is an important part of security

Spaces give away information! For example, SHIFT-BY-1 yields:

Dszuphsbqiz jt bo jnqpsubou qbsu pg tfdvsjuz

Without any fancy math Eve knows that the second and third word are two letters long. That's information she can use!

What to do?
I want to encode

*Cryptography is an important part of security*

Break it up into blocks of 5:

*Cryptography is an important part of security*

However you code it, spaces will not give anything away.
The First Step in Any Cipher-Other Issues

I want to encode

Are my TAs for CMSC/MATH 456 awesome? YES!
The First Step in Any Cipher-Other Issues

I want to encode

Are my TAs for CMSC/MATH 456 awesome? YES!

1. Capital and small letters leak information.
The First Step in Any Cipher—Other Issues

I want to encode

*Are my TAs for CMSC/MATH 456 awesome? YES!*

1. Capital and small letters leak information.
   Map everything to Capitals.
The First Step in Any Cipher—Other Issues

I want to encode

Are my TAs for CMSC/MATH 456 awesome? YES!

2. Punctuation leaks information.
The First Step in Any Cipher-Other Issues

I want to encode

*Are my TAs for CMSC/MATH 456 awesome? YES!*

1. Capital and small letters leak information.  
   Map everything to Capitals.
2. Punctuation leaks information.  
   Get rid of all punctuation.
The First Step in Any Cipher—Other Issues

I want to encode

*Are my TAs for CMSC/MATH 456 awesome? YES!*

1. Capital and small letters leak information.
   Map everything to Capitals.
2. Punctuation leaks information.
   Get rid of all punctuation.
3. What to do about numbers?
The First Step in Any Cipher—Other Issues

I want to encode

*Are my TAs for CMSC/MATH 456 awesome? YES!*

1. Capital and small letters leak information.
   Map everything to Capitals.
2. Punctuation leaks information.
   Get rid of all punctuation.
3. What to do about numbers?
   Just like letters—alphabet is 36 characters.
The First Step in Any Cipher—Other Issues

I want to encode

*Are my TAs for CMSC/MATH 456 awesome? YES!*

1. Capital and small letters leak information.  
   Map everything to Capitals.
2. Punctuation leaks information.  
   Get rid of all punctuation.
3. What to do about numbers?  
   Just like letters—alphabet is 36 characters  
   More generally, set your mod equal to your alphabet size.
The First Step in Any Cipher—Other Issues

I want to encode

Are my TAs for CMSC/MATH 456 awesome? YES!

1. Capital and small letters leak information.
   Map everything to Capitals.

2. Punctuation leaks information.
   Get rid of all punctuation.

3. What to do about numbers?
   Just like letters- alphabet is 36 characters
   More generally, set your mod equal to your alphabet size.

Note: In this class we will use 26-letter English only.
The Shift Cipher

lecture 01
The Shift Cipher

- Consider encrypting English text
- associate ‘a’ with 0; ‘b’ with 2; ...; ‘z’ with 25
- \( k \in \mathcal{K} = \{0, \ldots, 25\} \) (or could think of \( k \in \{a, \ldots, z\} \))
- To encrypt using key \( k \), shift every letter of the plaintext by \( k \) positions (with wraparound)
- Decryption just does the reverse

\[
\begin{align*}
\text{hello world} & \\
+22222 & 22222 \\
=\text{jgnnq yqtnf}
\end{align*}
\]
Modular arithmetic

- $x \equiv y \pmod{N}$ if and only if $N$ divides $x - y$.

- $[x \mod N] = \text{the remainder when } x \text{ is divided by } N$.
  - i.e. the unique value $y \in \{0, \ldots, N - 1\}$ such that $x \equiv y \pmod{N}$.

- $25 \equiv 35 \pmod{10}$

- $25 \neq [35 \mod 10]$

- $5 = [35 \mod 10]$
The Shift Cipher, Formally

- $\mathcal{M} = \{\text{all texts in lowercase English alphabet}\}$
  All arithmetic mod 26.

- Choose uniform $k \in \{0, \ldots, 25\}$

- Encode $(m_1 \ldots m_t)$ as $(m_1 + k, \ldots m_t + k)$

- Decode $(c_1 \ldots c_t)$ as $(c_1 - k, \ldots c_t - k)$

- Can verify that correctness holds.
Is the Shift Cipher Secure?

▶ No – only 26 possible keys!
  ▶ Given a ciphertext, try decrypting with every possible key
  ▶ Only one possibility will “make sense”

▶ Example of a “brute-force” or “exhaustive-search” attack
Example

- Ciphertext uryyb jbeyq
- Try every possible key...
  - tqxxa iadxp
  - spwwz hzcwo
  - ...
  - hello world

**Question:** We can tell that **hello world** is correct but how can a computer do that. Can we mechanize the process of picking out the right one?
Letter Frequencies

![Bar Chart showing letter frequencies]

- Letter a: 8.2%
- Letter b: 1.5%
- Letter c: 2.8%
- Letter d: 4.3%
- Letter e: 12.7%
- Letter f: 2.2%
- Letter g: 2.0%
- Letter h: 6.1%
- Letter i: 7.0%
- Letter j: 0.2%
- Letter k: 0.8%
- Letter l: 4.0%
- Letter m: 2.4%
- Letter n: 6.7%
- Letter o: 1.5%
- Letter p: 1.9%
- Letter q: 0.1%
- Letter r: 6.0%
- Letter s: 6.3%
- Letter t: 9.1%
- Letter u: 2.8%
- Letter v: 1.0%
- Letter w: 2.4%
- Letter x: 2.0%
- Letter y: 2.0%
- Letter z: 0.1%
Let $T$ be a long text of normal English. Let $\vec{f}$ be the freq vector of English. The components are all between 0 and 1 and add up to 1. We assume freq vector of $T$ is approx $\vec{f}$.

- One can compute that
  
  \[ \vec{f} \cdot \vec{f} \approx 0.065 \]

- Let $s \in \{1, \ldots, 25\}$. Let $T_s$ be the text shifted by $s$. Let $\vec{g}$ be the freq vector for $T_s$. One can compute that
  
  \[ \vec{f} \cdot \vec{g} \leq \approx 0.038 \]
Is English

We describe a way to tell if a text is English that we will use throughout this course. Let $\vec{f}$ be the freq vector for English.

1. Input($T$) a text
2. Compute $\vec{g}$, the freq vector for $T$
3. Compute $\vec{g} \cdot \vec{f}$. If $\approx 0.065$ then output YES, else NO
Cracking Shift Cipher

- Given $T$ a long text that you KNOW was coded by shift.
- For $s = 0$ to 25
  - Create $T_s$ which is $T$ shifted by $s$.
  - If $\text{Is English}(T_s) = \text{YES}$ then output $T_s$ and stop. Else try next value of $s$.

Note: No Near Misses. There will not be two values of $s$ that are both close to 0.065.

Pedagogical Note: Would normally have written Key instead of Note but the word Key is important in crypto so I can’t use it to say something is important. Oh Well.
A Note on Cracking Shift Cipher

In the last slide we tried all shifts in order. Can do better:

- Given $T$ a long text that you KNOW was coded by shift.
- Find frequencies of all letters, form vector $\vec{f}$
- Sort vector. So most common letter is $\sigma_1$, next is $\sigma_2$, etc.
- For $i = 0$ to 25
  - Create $T_s$ which is $T$ shifted as if $\sigma_i$ maps to $e$.
  - Compute $\vec{g}$, the freq vector for $T_s$
  - Compute $\vec{g} \cdot \vec{f}$. If $\approx 0.065$ then stop: $T_s$ is your text. Else try next value of $s$.

Note: Quite likely to succeed in the first try, or at least very early.
What if only transmit one letter?

**Odd Situation:** What if message is only one letter long?

**Discuss:** Can Eve crack a one-letter message?
What if only transmit one letter?

*Odd Situation:* What if message is only one letter long?  
*Discuss:* Can Eve crack a one-letter message?  
No (We will formalize this later.)
What if only transmit one letter?

**Odd Situation:** What if message is only one letter long?

**Discuss:** Can Eve crack a one-letter message?

No (We will formalize this later.)

**Discuss:** Can Eve learn from two 1-letter messages?
Odd Situation: What if message is only one letter long?
Discuss: Can Eve crack a one-letter message?
No (We will formalize this later.)
Discuss: Can Eve learn from two 1-letter messages?
Yes
Scenario:
In clear: Is Jacob a double agent working for the Klingons?
The answer comes via a shift cipher: A (which is either Y or N)
In clear: Is Jacob a double agent working for the Romulans?
The answer comes via a shift cipher: A (which is either Y or N)
What if only transmit one letter?

Odd Situation: What if message is only one letter long?
Discuss: Can Eve crack a one-letter message?
No (We will formalize this later.)
Discuss: Can Eve learn from two 1-letter messages?
Yes
Scenario:
In clear: Is Jacob a double agent working for the Klingons?
The answer comes via a shift cipher: A (which is either Y or N)
In clear: Is Jacob a double agent working for the Romulans?
The answer comes via a shift cipher: A (which is either Y or N)
Eve knows Jacob is working for either both or neither.
Eve Can Tell if Two Messages Are the Same or Not

**Issue:** If Eve sees two messages, will know if they are the same or different.

**Does this leak information:** Discuss
Eve Can Tell if Two Messages Are Same or Not

**Issue:** If Eve sees two messages, will know if they are the same or different.

**Does this leak information:** Discuss

**What to do about this?** Discuss
Eve Can Tell if Two Message Are Same or Not

**Issue:** If Eve sees two messages, will know if they are the same or different.

**Does this leak information:** Discuss

**What to do about this?** Discuss

**For Now Nothing** Will come back to this issue after a few more ciphers.

**For Now** A lesson in how even defining security and leak must be done carefully and rigorously.
Private-key encryption

\[ c := \text{Enc}_k(m) \]  \quad \text{message/plaintext}  

\[ m := \text{Dec}_k(c) \]  \quad \text{decryption}
Private-key encryption

\[ k \]
\[ m \]
\[ c := \text{Enc}_k(m) \]

\[ k \]
\[ m := \text{Dec}_k(c) \]
Private-key encryption

- A *private-key encryption scheme* is defined by a message space $\mathcal{M}$ and algorithms $(Gen, Enc, Dec)$:
  
  - *Gen* (key generation algorithm): outputs $k \in \mathcal{K}$
    (For SHIFT this is $k \in \{0, \ldots, 25\}$. Should 0 be included?)
  
  - *Enc* (encryption algorithm): takes key $k$ and message $m \in \mathcal{M}$ as input; outputs ciphertext $c$
    
    $$c \leftarrow Enc_k(m)$$
    
    (For SHIFT this is $Enc(m_1, \ldots, m_n) = (m_1 + k, \ldots, m_n + k)$.)
  
  - *Dec* (decryption algorithm): takes key $k$ and ciphertext $c$ as input; outputs $m$ or “error”
    
    $$m := Dec_k(c)$$
    
    (For SHIFT this is $Dec(c_1, \ldots, c_n) = (c_1 - k, \ldots, c_n - k)$.)

  $\forall k$ output by Gen $\forall m \in \mathcal{M}, Dec_k(Enc_k(m)) = m$

  (For SHIFT this is $(m + k) - k = m$)
Kerckhoffs’s principle

We made the comment We KNOW that SHIFT was used. More generally we use this principle.

- *The encryption scheme* is not secret
  - Eve knows the encryption scheme
  - The only secret is the key
  - The key must be chosen at random; kept secret

- Some arguments in favor of this principle
  - Easier to keep *key* secret than *algorithm*
  - Easier to change *key* than to change *algorithm*
  - Standardization
    - Ease of deployment
    - Public validation
Byte-wise Shift Cipher

lecture 01
Byte-wise Shift Cipher

- Instead of $a, b, c, d, \ldots, z$ have (for example) 0000, 0001, \ldots, 1111.
- Works for an alphabet of *bytes* rather than (English, lowercase) *letters*
  - Data in a computer is stored this way anyway. So works natively for arbitrary data!

- Use XOR instead of modular addition. Fast!
- Decode and Encode are both XOR.
  - Essential properties still hold
# Hexadecimal (base 16)

<table>
<thead>
<tr>
<th>Hex</th>
<th>Bits (&quot;nibble&quot;)</th>
<th>Decimal</th>
<th>Hex</th>
<th>Bits (&quot;nibble&quot;)</th>
<th>Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
<td>0</td>
<td>8</td>
<td>1000</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
<td>1</td>
<td>9</td>
<td>1001</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
<td>2</td>
<td>A</td>
<td>1010</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>0011</td>
<td>3</td>
<td>B</td>
<td>1011</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>0100</td>
<td>4</td>
<td>C</td>
<td>1100</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>0101</td>
<td>5</td>
<td>D</td>
<td>1101</td>
<td>13</td>
</tr>
<tr>
<td>6</td>
<td>0110</td>
<td>6</td>
<td>E</td>
<td>1110</td>
<td>14</td>
</tr>
<tr>
<td>7</td>
<td>0111</td>
<td>7</td>
<td>F</td>
<td>1111</td>
<td>15</td>
</tr>
</tbody>
</table>
Hexadecimal (base 16)

Notation: 0x before a string of \{0, 1, \ldots, 9, A, B, C, D, E, F\} means that the string will be base 16.

▶ 0x10

▶ 0x10 = 16*1 + 0 = 16
▶ 0x10 = 0001 0000

▶ 0xAF

▶ 0xAF = 16*A + F = 16*10 + 15 = 175
▶ 0xAF = 1010 1111
ASCII

- Characters (often) represented in ASCII with TWO hex-digits.
- Potentially 256 characters via \( \{0, \ldots, 9, A, \ldots, F\} \times \{0, \ldots, 9, A, \ldots, F\} \)
- Only use 128 characters via \( \{0, \ldots, 8\} \times \{0, \ldots, 9, A, \ldots, F\} \)
<table>
<thead>
<tr>
<th>Hex</th>
<th>Dec</th>
<th>Character</th>
<th>Hex</th>
<th>Dec</th>
<th>Character</th>
<th>Hex</th>
<th>Dec</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>0</td>
<td>NULL</td>
<td>0x01</td>
<td>1</td>
<td>SOH</td>
<td>0x02</td>
<td>2</td>
<td>STX</td>
</tr>
<tr>
<td>0x03</td>
<td>3</td>
<td>ETX</td>
<td>0x04</td>
<td>4</td>
<td>EOT</td>
<td>0x05</td>
<td>5</td>
<td>ENQ</td>
</tr>
<tr>
<td>0x06</td>
<td>6</td>
<td>ACK</td>
<td>0x07</td>
<td>7</td>
<td>BELL</td>
<td>0x08</td>
<td>8</td>
<td>BS</td>
</tr>
<tr>
<td>0x09</td>
<td>9</td>
<td>TAB</td>
<td>0x0A</td>
<td>10</td>
<td>LF</td>
<td>0x0B</td>
<td>11</td>
<td>VT</td>
</tr>
<tr>
<td>0x0C</td>
<td>12</td>
<td>FF</td>
<td>0x0D</td>
<td>13</td>
<td>CR</td>
<td>0x0E</td>
<td>14</td>
<td>SO</td>
</tr>
<tr>
<td>0x0F</td>
<td>15</td>
<td>SI</td>
<td>0x10</td>
<td>16</td>
<td>DLE</td>
<td>0x11</td>
<td>17</td>
<td>DC1</td>
</tr>
<tr>
<td>0x12</td>
<td>18</td>
<td>DC2</td>
<td>0x13</td>
<td>19</td>
<td>DC3</td>
<td>0x14</td>
<td>20</td>
<td>DC4</td>
</tr>
<tr>
<td>0x15</td>
<td>21</td>
<td>NAK</td>
<td>0x16</td>
<td>22</td>
<td>SYN</td>
<td>0x17</td>
<td>23</td>
<td>ETB</td>
</tr>
<tr>
<td>0x18</td>
<td>24</td>
<td>CAN</td>
<td>0x19</td>
<td>25</td>
<td>EM</td>
<td>0x1A</td>
<td>26</td>
<td>SUB</td>
</tr>
<tr>
<td>0x1B</td>
<td>27</td>
<td>FSC</td>
<td>0x1C</td>
<td>28</td>
<td>FS</td>
<td>0x1D</td>
<td>29</td>
<td>GS</td>
</tr>
<tr>
<td>0x1E</td>
<td>30</td>
<td>RS</td>
<td>0x1F</td>
<td>31</td>
<td>US</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ASCII

- ‘1’ = 0x31 = 0011 0001
- ‘F’ = 0x46 = 0100 0110
Useful observations

- Only 128 valid ASCII chars (128 bytes invalid)
- 0x20-0x7E printable
- 0x41-0x7A includes upper/lowercase letters
  - Uppercase letters begin with 0x4 or 0x5
  - Lowercase letters begin with 0x6 or 0x7
Byte-wise shift cipher

- $\mathcal{M} = \{\text{strings of bytes}\}$

- $Gen$: choose uniform byte $k \in K = \{0, \ldots, 255\}$

- $Enc_k(m_1 \ldots m_t)$: output $c_1 \ldots c_t$, where $c_i := m_i \oplus k$

- $Dec_k(c_1 \ldots c_t)$: output $m_1 \ldots m_t$, where $m_i := c_i \oplus k$

- Verify that correctness holds...
Example

Key is 11001110.
Alice wants to send 00011010, 11100011, 00000000
She sends

\[00011010 \oplus 11001110, 11100011 \oplus 11001110, 00000000 \oplus 11001110\]

\[= 11010100, 00101101, 11001110\]
Example

Key is \textbf{11001110}.
Alice wants to send \textbf{00011010, 11100011, 00000000}
She sends

\begin{align*}
00011010 \oplus 11001110, & \quad 11100011 \oplus 11001110, \quad 00000000 \oplus 11001110 \\
= 11010100, & \quad 00101101, \quad 11001110
\end{align*}

\textbf{Question}: Should it worry Alice and Bob that the key itself was transmitted? \textbf{Discuss}
Example

Key is 11001110.
Alice wants to send 00011010, 11100011, 00000000
She sends

\[
\begin{align*}
00011010 \oplus 11001110, & \quad 11100011 \oplus 11001110, \quad 00000000 \oplus 11001110 \\
= 11010100, & \quad 00101101, \quad 11001110
\end{align*}
\]

**Question:** Should it worry Alice and Bob that the key itself was transmitted?  **Discuss**
No. Eve has no way of knowing that.
Is this cipher secure?

- No – only 256 possible keys!
  - Given a ciphertext, try decrypting with every possible key
  - If ciphertext is long enough, only one plaintext will “look like English” (use the vector method of the last set of slides).

- Can further optimize
  - First nibble of plaintext likely 0x4, 0x5, 0x6, 0x7 (assuming letters only)
  - Can reduce exhaustive search to 26 keys (how?)
  - Talk to your friends or blood enemies about this.
Sufficient key space principle

- The key space must be large enough to make exhaustive-search attacks impractical
  - How large do you think that is?

- Note: this makes some assumptions...
  - English-language plaintext
  - Ciphertext sufficiently long so only one valid plaintext
Is this cipher secure if we are transmitting numbers?

If Alice sends Bob a Document in English via Byte-Shift then insecure!

What if Alice sends Bob a credit card number? Discuss
Is this cipher secure if we are transmitting numbers?

If Alice sends Bob a Document in English via Byte-Shift then insecure!

What if Alice sends Bob a credit card number? Discuss Credit Card Numbers also have patterns:

1. Visa cards always begin with 4
2. American Express always begins 34 or 37
3. Mastercard starts with 51 or 52 or 53 or 54.

Upshot: If Eve knows what kind of information is being transmitted (English, Credit Card Numbers, numbers on checks) she can use this to make any cipher with a small key space insecure.