BILL RECORDED LECTURE

October 19, 2020

REVIEW FOR MIDTERM

October 19, 2020

SHIFT CIPHER

October 19, 2020

The Shift Cipher, Formally

- $\mathcal{M} = \{\text{all texts in lowercase English alphabet}\}$
  $\mathcal{M}$ for **Message space**.
  All arithmetic mod 26.
- Choose uniform $s \in \mathcal{K} = \{0, \ldots, 25\}$. $\mathcal{K}$ for **Keyspace**.
- Encode $(m_1 \ldots m_t)$ as $(m_1 + s \ldots m_t + s)$.
- Decode $(c_1 \ldots c_t)$ as $(c_1 - s \ldots c_t - s)$.
- Can verify that correctness holds.

Freq Vectors

Let T be a long text. Length N. May or may not be coded.

Let N_a be the number of $a's$ in T.
Let N_b be the number of $b's$ in T.
$\vdots$

Freq Vectors

Let T be a long text. Length N. May or may not be coded.

Let N_a be the number of $a's$ in T.
Let N_b be the number of $b's$ in T.
$\vdots$

The **Freq Vector of T** is

$$\vec{f_T} = \left(\frac{N_a}{N}, \frac{N_b}{N}, \cdots, \frac{N_z}{N} \right)$$

English Alphabet: $\{a, \ldots, z\}$

- English freq shifted by 0 is $\vec{f_0}$
- For $1 \le i \le 25$, English freq shifted by i is $\vec{f_i}$.

English Alphabet: $\{a, \ldots, z\}$

- English freq shifted by 0 is $\vec{f_0}$
- For $1 \le i \le 25$, English freq shifted by i is $\vec{f_i}$.

$\vec{f_0} \cdot \vec{f_0} \sim 0.065$

English Alphabet: $\{a, \ldots, z\}$

- English freq shifted by 0 is $\vec{f_0}$
- For $1 \le i \le 25$, English freq shifted by i is $\vec{f_i}$.

$\vec{f_0} \cdot \vec{f_0} \sim 0.065$

$\max_{1 \le i \le 25} \vec{f_0} \cdot \vec{f_i} \sim 0.038$

English Alphabet: $\{a, \ldots, z\}$

- English freq shifted by 0 is $\vec{f_0}$
- For $1 \le i \le 25$, English freq shifted by i is $\vec{f_i}$.

$\vec{f_0} \cdot \vec{f_0} \sim 0.065$

$\max_{1 \le i \le 25} \vec{f_0} \cdot \vec{f_i} \sim 0.038$

Upshot
$\vec{f_0} \cdot \vec{f_0}$ **big**
For $i \in \{1, \ldots, 25\}$, $\vec{f_0} \cdot \vec{f_i}$ **small**

English Alphabet: $\{a, \ldots, z\}$

- English freq shifted by 0 is $\vec{f_0}$
- For $1 \le i \le 25$, English freq shifted by i is $\vec{f_i}$.

$\vec{f_0} \cdot \vec{f_0} \sim 0.065$

$\max_{1 \le i \le 25} \vec{f_0} \cdot \vec{f_i} \sim 0.038$

Upshot
$\vec{f_0} \cdot \vec{f_0}$ **big**
For $i \in \{1, \ldots, 25\}$, $\vec{f_0} \cdot \vec{f_i}$ **small**

Henceforth $\vec{f_0}$ will be denoted $\vec{f_E}$. E is for *English*

Is English

We describe a way to tell if a text **Is English** that we will use throughout this course.

Is English

We describe a way to tell if a text **Is English** that we will use throughout this course.

1. Input(T) a text
2. Compute $\vec{f_T}$, the freq vector for T
3. Compute $\vec{f_E} \cdot \vec{f_T}$. If ≈ 0.065 then output YES, else NO

Is English

We describe a way to tell if a text **Is English** that we will use throughout this course.

1. Input(T) a text
2. Compute $\vec{f_T}$, the freq vector for T
3. Compute $\vec{f_E} \cdot \vec{f_T}$. If ≈ 0.065 then output YES, else NO

Note: What if $\vec{f_T} \cdot \vec{f_E} = 0.061$?

Is English

We describe a way to tell if a text **Is English** that we will use throughout this course.

1. Input(T) a text
2. Compute $\vec{f_T}$, the freq vector for T
3. Compute $\vec{f_E} \cdot \vec{f_T}$. If ≈ 0.065 then output YES, else NO

Note: What if $\vec{f_T} \cdot \vec{f_E} = 0.061$?

If shift cipher used, this will **never** happen.

Is English

We describe a way to tell if a text **Is English** that we will use throughout this course.

1. Input(T) a text
2. Compute $\vec{f_T}$, the freq vector for T
3. Compute $\vec{f_E} \cdot \vec{f_T}$. If ≈ 0.065 then output YES, else NO

Note: What if $\vec{f_T} \cdot \vec{f_E} = 0.061$?

If shift cipher used, this will **never** happen.

If 'simple' ciphers used, this will **never** happen.

Is English

We describe a way to tell if a text **Is English** that we will use throughout this course.

1. Input(T) a text
2. Compute $\vec{f_T}$, the freq vector for T
3. Compute $\vec{f_E} \cdot \vec{f_T}$. If ≈ 0.065 then output YES, else NO

Note: What if $\vec{f_T} \cdot \vec{f_E} = 0.061$?

If shift cipher used, this will **never** happen.

If 'simple' ciphers used, this will **never** happen.

If 'difficult' cipher used, we may use different IS-ENGLISH function.

Cracking Shift Cipher

- Given T a long text that you KNOW was coded by shift.

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.

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 **Is English**(T_s)=YES then output T_s and stop. Else try next value of s.

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 **Is English**(T_s)=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.

Speeding Up Cracking of Shift Cipher

In the last slide we tried *all* shifts in order.

Speeding Up Cracking of Shift Cipher

In the last slide we tried *all* shifts in order.
Can do better: Most common letter is probably e. If not then 2nd most. . . .

Speeding Up Cracking of Shift Cipher

In the last slide we tried *all* shifts in order.
Can do better: Most common letter is probably e. If not then 2nd most....

- Given T a long text that you KNOW was coded by shift.

Speeding Up Cracking of Shift Cipher

In the last slide we tried *all* shifts in order.
Can do better: Most common letter is probably e. If not then 2nd most. . ..

- Given T a long text that you KNOW was coded by shift.
- Find frequencies of all letters, form vector $\vec{f}$.

Speeding Up Cracking of Shift Cipher

In the last slide we tried *all* shifts in order.
Can do better: Most common letter is probably e. If not then 2nd most....

- 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 σ_0, next is σ_1, etc.

Speeding Up Cracking of Shift Cipher

In the last slide we tried *all* shifts in order.
Can do better: Most common letter is probably e. If not then 2nd most....

- 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 σ_0, next is σ_1, etc.
- For $i = 0$ to 25

Speeding Up Cracking of Shift Cipher

In the last slide we tried *all* shifts in order.
Can do better: Most common letter is probably e. If not then 2nd most. . ..

- 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 σ_0, next is σ_1, etc.
- For $i = 0$ to 25
  - Create T_i which is T shifted as if σ_i maps to e.

Speeding Up Cracking of Shift Cipher

In the last slide we tried *all* shifts in order.
Can do better: Most common letter is probably e. If not then 2nd most....

- 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 σ_0, next is σ_1, etc.
- For $i = 0$ to 25
  - Create T_i which is T shifted as if σ_i maps to e.
  - Compute $\vec{g}$, the freq vector for T_i.

Speeding Up Cracking of Shift Cipher

In the last slide we tried *all* shifts in order.
Can do better: Most common letter is probably e. If not then 2nd most. . ..

- 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 σ_0, next is σ_1, etc.
- For $i = 0$ to 25
  - Create T_i which is T shifted as if σ_i maps to e.
  - Compute $\vec{g}$, the freq vector for T_i.
  - Compute $\vec{g} \cdot \vec{f_E}$. If ≈ 0.065 then stop: T_i is your text. Else try next value of i.

Speeding Up Cracking of Shift Cipher

In the last slide we tried *all* shifts in order.
Can do better: Most common letter is probably e. If not then 2nd most. . ..

- 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 σ_0, next is σ_1, etc.
- For $i = 0$ to 25
  - Create T_i which is T shifted as if σ_i maps to e.
  - Compute $\vec{g}$, the freq vector for T_i.
  - Compute $\vec{g} \cdot \vec{f_E}$. If ≈ 0.065 then stop: T_i is your text. Else try next value of i.

Note: Quite likely to succeed in the first try, or at least very early.

Variants of the Shift Cipher

1. $\Sigma = \{a, \ldots, z, 0, \ldots, 9, +, -, \times, \div\}$ (e.g., Math textbooks).
2. Σ is some other language (e.g., Greek, Russian).
3. $\Sigma = \{0, \ldots, 9\}$ (e.g, Credit Cards).
4. $\Sigma = \{0, 1\}^8$ (e.g., Ascii). Can use $\oplus$ instead of $+s$. Very fast!

These all have small key spaces and freq-of-symbol-use so can be cracked.

Include other symbols depending on the branch of math. E.g., $\wedge, \vee$ for logic.

Kerckhoff's principle

We made the comment **We KNOW that SHIFT was used.**
More generally we will always use the following assumption.
Kerckhoff's principle:

- Eve knows **The encryption scheme**.
- Eve knows **the alphabet and the language**.
- Eve does not know **the key**
- The key is chosen **at random**.

Definition of a Secure Crypto System

m will be a message.

Definition of a Secure Crypto System

m will be a message. c is what is sent.
If the following holds then the system is **secure**.

$$(\forall m, x, y, c)[\Pr(m = x | c = y) = \Pr(m = x)].$$

So seeing the y does not help Eve **at all**.

Definition of a Secure Crypto System

m will be a message. c is what is sent.
If the following holds then the system is **secure**.

$$(\forall m, x, y, c)[\Pr(m = x | c = y) = \Pr(m = x)].$$

So seeing the y does not help Eve **at all**.
Is this info-theoretic security or comp-security? Discuss

Definition of a Secure Crypto System

m will be a message. c is what is sent.
If the following holds then the system is **secure**.

$$(\forall m, x, y, c)[\Pr(m = x | c = y) = \Pr(m = x)].$$

So seeing the y does not help Eve **at all**.
Is this info-theoretic security or comp-security? Discuss
Info-Theoretic If Eve has unlimited computing power she still learns **nothing**.

One-Letter Shift is Secure!

Alphabet is $\{x, y\}$. $s \in \{0, 1\}$ randomly.
$\Pr(m = x) = p_x$. $\Pr(m = y) = p_y$.

One-Letter Shift is Secure!

Alphabet is $\{x, y\}$. $s \in \{0, 1\}$ randomly.
$\Pr(m = x) = p_x$. $\Pr(m = y) = p_y$. Eve knows this.

One-Letter Shift is Secure!

Alphabet is $\{x, y\}$. $s \in \{0, 1\}$ randomly.
$\Pr(m = x) = p_x$. $\Pr(m = y) = p_y$. Eve knows this.
Note that $p_x + p_y = 1$.

One-Letter Shift is Secure!

Alphabet is $\{x, y\}$. $s \in \{0, 1\}$ randomly.
$\Pr(m = x) = p_x$. $\Pr(m = y) = p_y$. Eve knows this.
Note that $p_x + p_y = 1$.

$$\Pr(m = x | c = x) = \frac{\Pr(m = x \wedge c = x)}{\Pr(c = x)}$$

One-Letter Shift is Secure!

Alphabet is $\{x, y\}$. $s \in \{0, 1\}$ randomly.
$\Pr(m = x) = p_x$. $\Pr(m = y) = p_y$. Eve knows this.
Note that $p_x + p_y = 1$.

$$\Pr(m = x | c = x) = \frac{\Pr(m = x \wedge c = x)}{\Pr(c = x)}$$

$\Pr(m = x \wedge c = x) = \Pr(m = x \wedge s = 0) = p_x \times \frac{1}{2} = 0.5p_x$

One-Letter Shift is Secure!

Alphabet is $\{x, y\}$. $s \in \{0, 1\}$ randomly.
$\Pr(m = x) = p_x$. $\Pr(m = y) = p_y$. Eve knows this.
Note that $p_x + p_y = 1$.

$$\Pr(m = x | c = x) = \frac{\Pr(m = x \wedge c = x)}{\Pr(c = x)}$$

$\Pr(m = x \wedge c = x) = \Pr(m = x \wedge s = 0) = p_x \times \frac{1}{2} = 0.5p_x$
$\Pr(c = x) = \Pr(m = x)\Pr(s = 0) + \Pr(m = y)\Pr(s = 1) =$
$0.5p_x + 0.5p_y = 0.5(p_x + p_y)$

One-Letter Shift is Secure!

Alphabet is $\{x, y\}$. $s \in \{0, 1\}$ randomly.
$\Pr(m = x) = p_x$. $\Pr(m = y) = p_y$. Eve knows this.
Note that $p_x + p_y = 1$.

$$\Pr(m = x | c = x) = \frac{\Pr(m = x \wedge c = x)}{\Pr(c = x)}$$

$\Pr(m = x \wedge c = x) = \Pr(m = x \wedge s = 0) = p_x \times \frac{1}{2} = 0.5p_x$
$\Pr(c = x) = \Pr(m = x)\Pr(s = 0) + \Pr(m = y)\Pr(s = 1) = 0.5p_x + 0.5p_y = 0.5(p_x + p_y)$

$$\Pr(m = x | c = x) = \frac{0.5p_x}{0.5(p_x + p_y)} = p_x$$

One-Letter Shift is Secure! (cont)

Alphabet is $\{x, y\}$. $s \in \{0, 1\}$ randomly.
$\Pr(m = x) = p_x$. $\Pr(m = y) = p_y$.

One-Letter Shift is Secure! (cont)

Alphabet is $\{x, y\}$. $s \in \{0, 1\}$ randomly.
$\Pr(m = x) = p_x$. $\Pr(m = y) = p_y$. Eve knows this.

One-Letter Shift is Secure! (cont)

Alphabet is $\{x, y\}$. $s \in \{0, 1\}$ randomly.
$\Pr(m = x) = p_x$. $\Pr(m = y) = p_y$. Eve knows this.
Note that $p_x + p_y = 1$.
We showed

$$\Pr(m = x | c = x) = p_x$$

One-Letter Shift is Secure! (cont)

Alphabet is $\{x, y\}$. $s \in \{0, 1\}$ randomly.
$\Pr(m = x) = p_x$. $\Pr(m = y) = p_y$. Eve knows this.
Note that $p_x + p_y = 1$.
We showed

$$\Pr(m = x | c = x) = p_x$$

One can show:

$$\Pr(m = x | c = y) = p_x.$$

$$\Pr(m = y | c = x) = p_y.$$

$$\Pr(m = y | c = y) = p_y.$$

One-Letter Shift is Secure! (cont)

Alphabet is $\{x, y\}$. $s \in \{0, 1\}$ randomly.
$\Pr(m = x) = p_x$. $\Pr(m = y) = p_y$. Eve knows this.
Note that $p_x + p_y = 1$.
We showed

$$\Pr(m = x | c = x) = p_x$$

One can show:

$$\Pr(m = x | c = y) = p_x.$$

$$\Pr(m = y | c = x) = p_y.$$

$$\Pr(m = y | c = y) = p_y.$$

So seeing the ciphertext gives Eve **NO INFORMATION**.
Upshot The 1-letter shift **Information-Theoretic Secure**.

Is 2-letter Shift Uncrackable?

Is 2-letter Shift Uncrackable? Discuss.

Is 2-letter Shift Uncrackable?

Is 2-letter Shift Uncrackable? Discuss.
No. Alphabet is $\{X, Y\}$.

Is 2-letter Shift Uncrackable?

Is 2-letter Shift Uncrackable? Discuss.
No. Alphabet is $\{X, Y\}$.
If Eve sees **XX** then she knows that the original message was one of

$$\{XX, YY\}$$

So Eve has learned something. HW will make this rigorous.

Summary

Summary

- Alice and Bob use shift s unif, 1-letter.

Summary

- Alice and Bob use shift s unif, 1-letter. **Secure**

Summary

- Alice and Bob use shift s unif, 1-letter. **Secure**
- Alice and Bob use shift s bias, 1-letter.

Summary

- Alice and Bob use shift s unif, 1-letter. **Secure**
- Alice and Bob use shift s bias, 1-letter. **Insecure**

Summary

- Alice and Bob use shift s unif, 1-letter. **Secure**
- Alice and Bob use shift s bias, 1-letter. **Insecure**
- Alice and Bob use shift s unif, 2-letters.

Summary

- Alice and Bob use shift s unif, 1-letter. **Secure**
- Alice and Bob use shift s bias, 1-letter. **Insecure**
- Alice and Bob use shift s unif, 2-letters. **Insecure**

Summary

- Alice and Bob use shift s unif, 1-letter. **Secure**
- Alice and Bob use shift s bias, 1-letter. **Insecure**
- Alice and Bob use shift s unif, 2-letters. **Insecure**

Other Single Letter Ciphers

October 19, 2020

Affine Cipher

Def The Affine cipher with a, b:

1. Encrypt via $x \to ax + b \pmod{26}$. (a has to be rel prime to 26 so that $a^{-1} \pmod{26}$ exists.
2. Decrypt via $x \to a^{-1}(x - b) \pmod{26}$.

Limit on Keys (a, b) must be such that a has an inverse. More on next page.
Easily cracked Only 312 keys. Use **Is-English** for each key.

The (a, b) for the Affine Cipher

Shift Cipher Key s could be **anything** in $\{0, \ldots, 25\}$. 26 keys.

The (a, b) for the Affine Cipher

Shift Cipher Key s could be **anything** in $\{0, \ldots, 25\}$. 26 keys.

Affine Cipher Key a has to be rel prime to 26, b can be anything.

The (a, b) for the Affine Cipher

Shift Cipher Key s could be **anything** in $\{0, \ldots, 25\}$. 26 keys.

Affine Cipher Key a has to be rel prime to 26, b can be anything.

If alphabet is size n then how many a's are usable?

The (a, b) for the Affine Cipher

Shift Cipher Key s could be **anything** in $\{0, \ldots, 25\}$. 26 keys.

Affine Cipher Key a has to be rel prime to 26, b can be anything.

If alphabet is size n then how many a's are usable?
The number of elts of $\{0, \ldots, n-1\}$ that are rel prime to n.
Do we have another name for this?

The (a, b) for the Affine Cipher

Shift Cipher Key s could be **anything** in $\{0, \ldots, 25\}$. 26 keys.

Affine Cipher Key a has to be rel prime to 26, b can be anything.

If alphabet is size n then how many a's are usable?
The number of elts of $\{0, \ldots, n-1\}$ that are rel prime to n.
Do we have another name for this? Yes: $\phi(n)$.

The (a, b) for the Affine Cipher

Shift Cipher Key s could be **anything** in $\{0, \ldots, 25\}$. 26 keys.

Affine Cipher Key a has to be rel prime to 26, b can be anything.

If alphabet is size n then how many a's are usable?
The number of elts of $\{0, \ldots, n-1\}$ that are rel prime to n.
Do we have another name for this? Yes: $\phi(n)$.

How to compute $\phi(n)$.

The (a, b) for the Affine Cipher

Shift Cipher Key s could be **anything** in $\{0, \ldots, 25\}$. 26 keys.

Affine Cipher Key a has to be rel prime to 26, b can be anything.

If alphabet is size n then how many a's are usable?
The number of elts of $\{0, \ldots, n-1\}$ that are rel prime to n.
Do we have another name for this? Yes: $\phi(n)$.

How to compute $\phi(n)$.

- n is small: list out all numbers $\leq n-1$ that are rel prime to n.

The (a, b) for the Affine Cipher

Shift Cipher Key s could be **anything** in $\{0, \ldots, 25\}$. 26 keys.

Affine Cipher Key a has to be rel prime to 26, b can be anything.

If alphabet is size n then how many a's are usable?
The number of elts of $\{0, \ldots, n-1\}$ that are rel prime to n.
Do we have another name for this? Yes: $\phi(n)$.

How to compute $\phi(n)$.

- n is small: list out all numbers $\leq n-1$ that are rel prime to n.
- If p is prime $\phi(p) = p - 1$.

The (a, b) for the Affine Cipher

Shift Cipher Key s could be **anything** in $\{0, \ldots, 25\}$. 26 keys.

Affine Cipher Key a has to be rel prime to 26, b can be anything.

If alphabet is size n then how many a's are usable?
The number of elts of $\{0, \ldots, n-1\}$ that are rel prime to n.
Do we have another name for this? Yes: $\phi(n)$.

How to compute $\phi(n)$.

- n is small: list out all numbers $\le n-1$ that are rel prime to n.
- If p is prime $\phi(p) = p - 1$.
  If p, q are prime then (HW 2, Prob 6) $\phi(pq) = \phi(p)\phi(q)$.

The (a, b) for the Affine Cipher

Shift Cipher Key s could be **anything** in $\{0, \ldots, 25\}$. 26 keys.

Affine Cipher Key a has to be rel prime to 26, b can be anything.

If alphabet is size n then how many a's are usable?
The number of elts of $\{0, \ldots, n-1\}$ that are rel prime to n.
Do we have another name for this? Yes: $\phi(n)$.

How to compute $\phi(n)$.

- n is small: list out all numbers $\leq n-1$ that are rel prime to n.
- If p is prime $\phi(p) = p - 1$.
  If p, q are prime then (HW 2, Prob 6) $\phi(pq) = \phi(p)\phi(q)$.
  Can extend to get a formula for $\phi(p_1^{a_1} \cdots p_k^{a_k})$.

The (a, b) for the Affine Cipher

Shift Cipher Key s could be **anything** in $\{0, \ldots, 25\}$. 26 keys.

Affine Cipher Key a has to be rel prime to 26, b can be anything.

If alphabet is size n then how many a's are usable?
The number of elts of $\{0, \ldots, n-1\}$ that are rel prime to n.
Do we have another name for this? Yes: $\phi(n)$.

How to compute $\phi(n)$.

- n is small: list out all numbers $\le n - 1$ that are rel prime to n.
- If p is prime $\phi(p) = p - 1$.
  If p, q are prime then (HW 2, Prob 6) $\phi(pq) = \phi(p)\phi(q)$.
  Can extend to get a formula for $\phi(p_1^{a_1} \cdots p_k^{a_k})$.
  Caveat: To really use it need to factor n.

The Quadratic Cipher

Def The Quadratic cipher with a, b, c: Encrypt via $x \to ax^2 + bx + c \pmod{26}$.

The Quadratic Cipher

Def The Quadratic cipher with a, b, c: Encrypt via
$x \rightarrow ax^2 + bx + c \pmod{26}$.
Does not work and was never used because:

No easy test for Invertibility (depends on def of easy).

General Substitution Cipher

Def **Gen Sub Cipher** with perm f on $\{0, \ldots, 25\}$.

1. Encrypt via $x \to f(x)$.
2. Decrypt via $x \to f^{-1}(x)$.

PRO Very Large Key Space: 26!, so brute force not an option.
CON 100 years ago Hard to use, so we will look at alternatives that take a short seed and get a **random looking** perm.
CON today Crackable. We discuss how later.

Keyword-Shift Cipher. Key is (Word,Shift)

$\Sigma = \{a, \ldots, k\}$. **Key:** (jack, 4).

Keyword-Shift Cipher. Key is (Word,Shift)

$\Sigma = \{a, \ldots, k\}$. **Key:** (jack, 4).
Alice then does the following:

Keyword-Shift Cipher. Key is (Word,Shift)

$\Sigma = \{a, \ldots, k\}$. **Key:** (jack, 4).
Alice then does the following:

1. List out the key word and then the remaining letters:

j	a	c	k	b	d	e	f	g	h	i

Keyword-Shift Cipher. Key is (Word,Shift)

$\Sigma = \{a, \ldots, k\}$. **Key:** (jack, 4).
Alice then does the following:

1. List out the key word and then the remaining letters:

j	a	c	k	b	d	e	f	g	h	i

2. Now do Shift 4 on this:

f	g	h	i	j	a	c	k	b	d	e

   This is where $a, b, c, \ldots$ go, so:

a	b	c	d	e	f	g	h	i	j	k
f	g	h	i	j	a	c	k	b	d	e

UPSHOT

1. From short key got random looking perm (in its day, not now).
2. Keyword Mixed cipher is similar, probably better, but we skip.

Keyword-Shift vs Truly Random

Alice and Eve play the following game:

Keyword-Shift vs Truly Random

Alice and Eve play the following game:
Game: $\Sigma = \{a, b, \ldots, z\}$. L is length of keyword, $L = 6$.

Keyword-Shift vs Truly Random

Alice and Eve play the following game:

Game: $\Sigma = \{a, b, \ldots, z\}$. L is length of keyword, $L = 6$.

1. Alice flips a fair coin.

Keyword-Shift vs Truly Random

Alice and Eve play the following game:

Game: $\Sigma = \{a, b, \ldots, z\}$. L is length of keyword, $L = 6$.

1. Alice flips a fair coin.

   If T then Alice gen rand perm of Σ and sends to Eve.

Keyword-Shift vs Truly Random

Alice and Eve play the following game:

Game: $\Sigma = \{a, b, \ldots, z\}$. L is length of keyword, $L = 6$.

1. Alice flips a fair coin.

   If T then Alice gen rand perm of Σ and sends to Eve.
   If H then Alice gen rand word $w \in \Sigma^6$, with 6 **diff** letters, rand $s \in \mathbb{Z}_{25}$, creates a perm using Keyword-Shift with w, s, and sends to Eve.

Keyword-Shift vs Truly Random

Alice and Eve play the following game:

Game: $\Sigma = \{a, b, \ldots, z\}$. L is length of keyword, $L = 6$.

1. Alice flips a fair coin.

   If T then Alice gen rand perm of Σ and sends to Eve.

   If H then Alice gen rand word $w \in \Sigma^6$, with 6 **diff** letters, rand $s \in \mathbb{Z}_{25}$, creates a perm using Keyword-Shift with w, s, and sends to Eve.

2. Eve says RP (Rand Perm) if she thinks Alice flipped T, KS (Keyword-Shift) if she thinks Alice flipped H. If Eve is correct she wins! If not then Alice wins!

Alice has no strategy in this game.

Keyword-Shift vs Truly Random

Alice and Eve play the following game:

Game: $\Sigma = \{a, b, \ldots, z\}$. L is length of keyword, $L = 6$.

1. Alice flips a fair coin.

   If T then Alice gen rand perm of Σ and sends to Eve.
   If H then Alice gen rand word $w \in \Sigma^6$, with 6 **diff** letters, rand $s \in \mathbb{Z}_{25}$, creates a perm using Keyword-Shift with w, s, and sends to Eve.

2. Eve says RP (Rand Perm) if she thinks Alice flipped T, KS (Keyword-Shift) if she thinks Alice flipped H. If Eve is correct she wins! If not then Alice wins!

Alice has no strategy in this game.
Eve can have a strategy. If Eve is unlimited then she can do quite well.

Keyword-Shift vs Truly Random

Alice and Eve play the following game:

Game: $\Sigma = \{a, b, \ldots, z\}$. L is length of keyword, $L = 6$.

1. Alice flips a fair coin.

   If T then Alice gen rand perm of Σ and sends to Eve.
   If H then Alice gen rand word $w \in \Sigma^6$, with 6 **diff** letters, rand $s \in \mathbb{Z}_{25}$, creates a perm using Keyword-Shift with w, s, and sends to Eve.

2. Eve says RP (Rand Perm) if she thinks Alice flipped T, KS (Keyword-Shift) if she thinks Alice flipped H. If Eve is correct she wins! If not then Alice wins!

Alice has no strategy in this game.

Eve can have a strategy. If Eve is unlimited then she can do quite well.

We measure how good the Keyword-Shift is by the probability that an optimal Eve can win.

Unlimited Eve Strategy

Assume Eve has unlimited computational power.

Unlimited Eve Strategy

Assume Eve has unlimited computational power.
Before Eve plays the game she does the following:

Unlimited Eve Strategy

Assume Eve has unlimited computational power.
Before Eve plays the game she does the following:

- For every word $w \in \Sigma^6$ (all diff letters) and shift $s \in \{0, \ldots, 25\}$ find the perm generated by keyword-Shift.

Unlimited Eve Strategy

Assume Eve has unlimited computational power.
Before Eve plays the game she does the following:

- For every word $w \in \Sigma^6$ (all diff letters) and shift $s \in \{0, \ldots, 25\}$ find the perm generated by keyword-Shift.
- Store all $L = 26 \times 25 \times 24 \times 23 \times 22 \times 21 \times 26$ perms: $\sigma_1, \sigma_2, \ldots, \sigma_L$.

Unlimited Eve Strategy

Assume Eve has unlimited computational power.
Before Eve plays the game she does the following:

- For every word $w \in \Sigma^6$ (all diff letters) and shift $s \in \{0, \ldots, 25\}$ find the perm generated by keyword-Shift.
- Store all $L = 26 \times 25 \times 24 \times 23 \times 22 \times 21 \times 26$ perms: $\sigma_1, \sigma_2, \ldots, \sigma_L$.
- Note that the number of perms is $\sim 10^9$.

Unlimited Eve Strategy

Assume Eve has unlimited computational power.
Before Eve plays the game she does the following:

- For every word $w \in \Sigma^6$ (all diff letters) and shift $s \in \{0, \ldots, 25\}$ find the perm generated by keyword-Shift.
- Store all $L = 26 \times 25 \times 24 \times 23 \times 22 \times 21 \times 26$ perms: $\sigma_1, \sigma_2, \ldots, \sigma_L$.
- Note that the number of perms is $\sim 10^9$.
- Note that $26! \sim 10^{26}$.

Unlimited Eve Strategy

Assume Eve has unlimited computational power.
Before Eve plays the game she does the following:

- For every word $w \in \Sigma^6$ (all diff letters) and shift $s \in \{0, \ldots, 25\}$ find the perm generated by keyword-Shift.
- Store all $L = 26 \times 25 \times 24 \times 23 \times 22 \times 21 \times 26$ perms: $\sigma_1, \sigma_2, \ldots, \sigma_L$.
- Note that the number of perms is $\sim 10^9$.
- Note that $26! \sim 10^{26}$.

Eve's strategy:

Unlimited Eve Strategy

Assume Eve has unlimited computational power.
Before Eve plays the game she does the following:

- For every word $w \in \Sigma^6$ (all diff letters) and shift $s \in \{0, \ldots, 25\}$ find the perm generated by keyword-Shift.
- Store all $L = 26 \times 25 \times 24 \times 23 \times 22 \times 21 \times 26$ perms: $\sigma_1, \sigma_2, \ldots, \sigma_L$.
- Note that the number of perms is $\sim 10^9$.
- Note that $26! \sim 10^{26}$.

Eve's strategy:
Alice gives Eve perm τ. If τ is one of the σ_i then Eve says KS, otherwise Eve says RP.

Unlimited Eve Analysis

Unlimited Eve Analysis

- If KS then Eve will guess it correctly.

Unlimited Eve Analysis

- If KS then Eve will guess it correctly.
- If RP then the prob Eve gets it wrong is the prob that perm just happens to be one of the σ_i:

$$\sim \frac{10^9}{10^{26}} = \frac{1}{10^{17}}$$

Unlimited Eve Analysis

- If KS then Eve will guess it correctly.
- If RP then the prob Eve gets it wrong is the prob that perm just happens to be one of the σ_i:

$$\sim \frac{10^9}{10^{26}} = \frac{1}{10^{17}}$$

Prob Eve right is $1 - \frac{1}{10^{17}} = 0.99999999999999999 = L$.

Unlimited Eve Analysis

- If KS then Eve will guess it correctly.
- If RP then the prob Eve gets it wrong is the prob that perm just happens to be one of the σ_i:

$$\sim \frac{10^9}{10^{26}} = \frac{1}{10^{17}}$$

Prob Eve right is $1 - \frac{1}{10^{17}} = 0.9999999999999999 = L$.

Prob Eve wins is
$\Pr(KS) \times 1 + \Pr(RP) \times L = \frac{1}{2} \times 1 + \frac{1}{2} \times L = \frac{1}{2}(1 + L) = L'$
which is very close to 1.

Upshot Unlimited Eve wins most of the time.

Strategy for Comp Limited Eve

Strategy for Comp Limited Eve

1. Eve gets τ.

Strategy for Comp Limited Eve

1. Eve gets τ.
2. If τ has 3 consecutive letters (e.g., p, q, r) then say KS, else say RP. (We do not count wrap around.)

Prob that Limited Eve Wins

If KS then Eve is correct (we omit this part).

Prob that Limited Eve Wins

If KS then Eve is correct (we omit this part).
If RP then prob Eve wrong is prob a rand perm has 3 cons lets.

- Number of perms: 26!
- Number of perms with 3 consecutive letters:

Prob that Limited Eve Wins

If KS then Eve is correct (we omit this part).
If RP then prob Eve wrong is prob a rand perm has 3 cons lets.

- Number of perms: 26!
- Number of perms with 3 consecutive letters:

  Pick the space to begin the 3 cons lets: 24 $(a, \ldots, x)$

Prob that Limited Eve Wins

If KS then Eve is correct (we omit this part).
If RP then prob Eve wrong is prob a rand perm has 3 cons lets.

- Number of perms: 26!
- Number of perms with 3 consecutive letters:

  Pick the space to begin the 3 cons lets: 24 $(a, \ldots, x)$

  Pick the let to put there (also determines the next 2 lets): 26

Prob that Limited Eve Wins

If KS then Eve is correct (we omit this part).
If RP then prob Eve wrong is prob a rand perm has 3 cons lets.

- Number of perms: 26!
- Number of perms with 3 consecutive letters:

  Pick the space to begin the 3 cons lets: 24 ($a, \ldots, x$)

  Pick the let to put there (also determines the next 2 lets): 26

  Permute remaining 23 letters in remaining 23 places: 23!

Prob that Limited Eve Wins

If KS then Eve is correct (we omit this part).
If RP then prob Eve wrong is prob a rand perm has 3 cons lets.

- Number of perms: 26!
- Number of perms with 3 consecutive letters:

  Pick the space to begin the 3 cons lets: 24 $(a, \ldots, x)$

  Pick the let to put there (also determines the next 2 lets): 26

  Permute remaining 23 letters in remaining 23 places: 23!

  We have counted some perms ≥ 2 times. So
  Numb of perms with 3 cons lets is $\leq 24 \times 26 \times 23!$.

Prob that Limited Eve Wins

If KS then Eve is correct (we omit this part).
If RP then prob Eve wrong is prob a rand perm has 3 cons lets.

- Number of perms: 26!
- Number of perms with 3 consecutive letters:

  Pick the space to begin the 3 cons lets: 24 ($a, \ldots, x$)

  Pick the let to put there (also determines the next 2 lets): 26

  Permute remaining 23 letters in remaining 23 places: 23!

  We have counted some perms ≥ 2 times. So
  Numb of perms with 3 cons lets is $\leq 24 \times 26 \times 23!$.

  Prob that Alice picks perm with 3 cons lets is

$$\leq \frac{24 \times 26 \times 23!}{26!} = \frac{1}{25} = 0.04$$

Prob that Limited Eve Wins

If KS then Eve is correct (we omit this part).
If RP then prob Eve wrong is prob a rand perm has 3 cons lets.

- Number of perms: 26!
- Number of perms with 3 consecutive letters:

  Pick the space to begin the 3 cons lets: 24 $(a, \ldots, x)$

  Pick the let to put there (also determines the next 2 lets): 26

  Permute remaining 23 letters in remaining 23 places: 23!

  We have counted some perms ≥ 2 times. So
  Numb of perms with 3 cons lets is $\leq 24 \times 26 \times 23!$.

  Prob that Alice picks perm with 3 cons lets is

$$\leq \frac{24 \times 26 \times 23!}{26!} = \frac{1}{25} = 0.04$$

  Prob that Eve wins is $\geq 1 - 0.04 = 0.96$.

Prob Eve wins is $\frac{1}{2} \times 1 + \frac{1}{2} \times 0.096 = 0.98$

Terminology: 1-Gram, 2-Gram, 3-Gram

Notation Let T be a text.

Terminology: 1-Gram, 2-Gram, 3-Gram

Notation Let T be a text.

1. The **1-grams** of T are just the letters in T, counting repeats.

Terminology: 1-Gram, 2-Gram, 3-Gram

Notation Let T be a text.

1. The **1-grams** of T are just the letters in T, counting repeats.
2. The **2-grams** of T are just the contiguous pairs of letters in T, counting repeats. Also called **bigrams**.

Terminology: 1-Gram, 2-Gram, 3-Gram

Notation Let T be a text.

1. The **1-grams** of T are just the letters in T, counting repeats.
2. The **2-grams** of T are just the contiguous pairs of letters in T, counting repeats. Also called **bigrams**.
3. The **3-grams** of T you can guess. Also called **trigrams**.

Terminology: 1-Gram, 2-Gram, 3-Gram

Notation Let T be a text.

1. The **1-grams** of T are just the letters in T, counting repeats.
2. The **2-grams** of T are just the contiguous pairs of letters in T, counting repeats. Also called **bigrams**.
3. The **3-grams** of T you can guess. Also called **trigrams**.
4. One usually talks about the freq of n-grams.

Notation and Parameter for a Family of Algorithms

Notation Let σ be a perm and T a text.

Notation and Parameter for a Family of Algorithms

Notation Let σ be a perm and T a text.

1. f_E is freq of n-grams. It is a 26^n long vector. (Formally we should use $f_E(n)$. We omit the n. The value of n will be clear from context.)

Notation and Parameter for a Family of Algorithms

Notation Let σ be a perm and T a text.

1. f_E is freq of n-grams. It is a 26^n long vector. (Formally we should use $f_E(n)$. We omit the n. The value of n will be clear from context.)
2. $\sigma(T)$ is taking T and applying σ to it. If σ^{-1} was used to encrypt, then $\sigma(T)$ will be English!

Notation and Parameter for a Family of Algorithms

Notation Let σ be a perm and T a text.

1. f_E is freq of n-grams. It is a 26^n long vector. (Formally we should use $f_E(n)$. We omit the n. The value of n will be clear from context.)
2. $\sigma(T)$ is taking T and applying σ to it. If σ^{-1} was used to encrypt, then $\sigma(T)$ will be English!
3. $f_{\sigma(T)}$ is the 26^n-long vector of freq's of n-grams in $\sigma(T)$.

Notation and Parameter for a Family of Algorithms

Notation Let σ be a perm and T a text.

1. f_E is freq of n-grams. It is a 26^n long vector. (Formally we should use $f_E(n)$. We omit the n. The value of n will be clear from context.)
2. $\sigma(T)$ is taking T and applying σ to it. If σ^{-1} was used to encrypt, then $\sigma(T)$ will be English!
3. $f_{\sigma(T)}$ is the 26^n-long vector of freq's of n-grams in $\sigma(T)$.
4. I and R will be parameters we discuss later.

Notation and Parameter for a Family of Algorithms

Notation Let σ be a perm and T a text.

1. f_E is freq of n-grams. It is a 26^n long vector. (Formally we should use $f_E(n)$. We omit the n. The value of n will be clear from context.)
2. $\sigma(T)$ is taking T and applying σ to it. If σ^{-1} was used to encrypt, then $\sigma(T)$ will be English!
3. $f_{\sigma(T)}$ is the 26^n-long vector of freq's of n-grams in $\sigma(T)$.
4. I and R will be parameters we discuss later.
   I stands for $\mathrm{Iterations}$ and will be large (like 2000).

Notation and Parameter for a Family of Algorithms

Notation Let σ be a perm and T a text.

1. f_E is freq of n-grams. It is a 26^n long vector. (Formally we should use $f_E(n)$. We omit the n. The value of n will be clear from context.)
2. $\sigma(T)$ is taking T and applying σ to it. If σ^{-1} was used to encrypt, then $\sigma(T)$ will be English!
3. $f_{\sigma(T)}$ is the 26^n-long vector of freq's of n-grams in $\sigma(T)$.
4. I and R will be parameters we discuss later.
   I stands for $\mathrm{Iterations}$ and will be large (like 2000).
   R stands for Redos and will be small (like 5).

n-Gram Algorithm

Input T. Find Freq of 1-grams and n-grams.

n-Gram Algorithm

Input T. Find Freq of 1-grams and n-grams.
σ_{init} is perm that maps most freq to e, etc. Uses 1-gram freq.

n-Gram Algorithm

Input T. Find Freq of 1-grams and n-grams.
σ_{init} is perm that maps most freq to e, etc. Uses 1-gram freq.

For $r = 1$ to R (R is small, about 5)

n-Gram Algorithm

Input T. Find Freq of 1-grams and n-grams.
$\sigma_{\rm init}$ is perm that maps most freq to e, etc. Uses 1-gram freq.

For $r = 1$ to R (R is small, about 5)

$\sigma_r \leftarrow \sigma_{\rm init}$

n-Gram Algorithm

Input T. Find Freq of 1-grams and n-grams.
$\sigma_{\rm init}$ is perm that maps most freq to e, etc. Uses 1-gram freq.

For $r = 1$ to $\rm R$ ($\rm R$ is small, about 5)

$\sigma_r \leftarrow \sigma_{\rm init}$
For $i = 1$ to $\rm I$ ($\rm I$ is large, about 2000)

n-Gram Algorithm

Input T. Find Freq of 1-grams and n-grams.
$\sigma_{\rm init}$ is perm that maps most freq to e, etc. Uses 1-gram freq.

For $r = 1$ to R (R is small, about 5)

  $\sigma_r \leftarrow \sigma_{\rm init}$
  For $i = 1$ to I (I is large, about 2000)
    Pick $j, k \in \{0, \ldots, 25\}$ at Random.

n-Gram Algorithm

Input T. Find Freq of 1-grams and n-grams.
σ_{init} is perm that maps most freq to e, etc. Uses 1-gram freq.

For $r = 1$ to R (R is small, about 5)

> $\sigma_r \leftarrow \sigma_{\mathrm{init}}$
> For $i = 1$ to I (I is large, about 2000)
>
> > Pick $j, k \in \{0, \ldots, 25\}$ at Random.
> > Let σ' be σ_r with j, k swapped

n-Gram Algorithm

Input T. Find Freq of 1-grams and n-grams.
$\sigma_{\rm init}$ is perm that maps most freq to e, etc. Uses 1-gram freq.

For $r = 1$ to $\rm R$ ($\rm R$ is small, about 5)

$\sigma_r \leftarrow \sigma_{\rm init}$
For $i = 1$ to $\rm I$ ($\rm I$ is large, about 2000)
Pick $j, k \in \{0, \ldots, 25\}$ at Random.
Let σ' be σ_r with j, k swapped
If $f_{\sigma'(T)} \cdot f_E > f_{\sigma_r(T)} \cdot f_E$ then $\sigma_r \leftarrow \sigma'$

n-Gram Algorithm

Input T. Find Freq of 1-grams and n-grams.
$\sigma_{\rm init}$ is perm that maps most freq to e, etc. Uses 1-gram freq.

For $r = 1$ to $\rm R$ ($\rm R$ is small, about 5)

$\sigma_r \leftarrow \sigma_{\rm init}$
For $i = 1$ to $\rm I$ ($\rm I$ is large, about 2000)
Pick $j, k \in \{0, \ldots, 25\}$ at Random.
Let σ' be σ_r with j, k swapped
If $f_{\sigma'(T)} \cdot f_E > f_{\sigma_r(T)} \cdot f_E$ then $\sigma_r \leftarrow \sigma'$

Candidates for σ are $\sigma_1, \ldots, \sigma_{\rm R}$

n-Gram Algorithm

Input T. Find Freq of 1-grams and n-grams.
$\sigma_{\rm init}$ is perm that maps most freq to e, etc. Uses 1-gram freq.

For $r = 1$ to $\rm R$ ($\rm R$ is small, about 5)

$\sigma_r \leftarrow \sigma_{\rm init}$
For $i = 1$ to $\rm I$ ($\rm I$ is large, about 2000)
Pick $j, k \in \{0, \ldots, 25\}$ at Random.
Let σ' be σ_r with j, k swapped
If $f_{\sigma'(T)} \cdot f_E > f_{\sigma_r(T)} \cdot f_E$ then $\sigma_r \leftarrow \sigma'$

Candidates for σ are $\sigma_1, \ldots, \sigma_{\rm R}$
Pick the σ_r with min good_r or have human look at all $\sigma_r(T)$

n-Gram Algorithm

Input T. Find Freq of 1-grams and n-grams.
$\sigma_{\rm init}$ is perm that maps most freq to e, etc. Uses 1-gram freq.

For $r = 1$ to R (R is small, about 5)

$\sigma_r \leftarrow \sigma_{\rm init}$
For $i = 1$ to I (I is large, about 2000)
Pick $j, k \in \{0, \ldots, 25\}$ at Random.
Let σ' be σ_r with j, k swapped
If $f_{\sigma'(T)} \cdot f_E > f_{\sigma_r(T)} \cdot f_E$ then $\sigma_r \leftarrow \sigma'$

Candidates for σ are $\sigma_1, \ldots, \sigma_{\mathrm{R}}$
Pick the σ_r with min good_r or have human look at all $\sigma_r(T)$
The parameters R and I need to be picked carefully.