Adin Ilfeld

COMS30023 / Cryptology

Problem Sheet 1 – Classical Ciphers and Blockcipher Construction

Dr François Dupressoir*

2023/24

This problem sheet first explores blockciphers a bit more in depth by considering classical designs and how they fail. This helps you practice attack finding skills on simple designs.

We will consider blockciphers for which $\mathcal{M} = \mathcal{C} = \{\mathtt{a}, \dots, \mathtt{z}\}^9$ (that is, plaintexts are 9-letter strings).

1. (a) $\star$ Determine $|\mathcal{M}|$ and estimate, to one decimal, $\lg(|\mathcal{M}|)$.

$|M| = 26^9 \approx \boxed{5.43 \times 10^{12}}$

$\lg(|M|) = \lg(26^9) \approx \boxed{42.3}$

(b) $\star$ Determine $|\mathrm{Perm}(\mathcal{M})|$. What can you say about $\lg(|\mathrm{Perm}(\mathcal{M})|)$? ($\mathrm{Perm}(\mathcal{X})$ is the set of *permutations* of $\mathcal{X}$.)

$|\mathrm{Perm}(M)| = |M|! = \boxed{26^9!}$

$\lg(|\mathrm{Perm}(M)|) = \lg(26^9!)$

$\approx 26^9 \cdot \lg(26^9) - 26^9 \lg(e)$ by Stirling's approximation

$= \boxed{2.2 \times 10^{14}}$ Very big!

(c) ⋆ What do those quantities represent, and why might we be interested in them?

$|\mathcal{M}|$ represents the total number of plaintexts possible, while $|\text{Perm}(\mathcal{M})|$ represents the total number of ways to map from plaintexts to ciphertexts; that is, the number of possible keys.

Computing them gives information about how hard these may be to crack by brute force.

$\lg|\mathcal{M}|$ = # of bits to represent elements in message space

$|\lg(\text{Perm}|\mathcal{M}|)|$ = # of bits to randomly draw a permutation from the set

2. You may have heard of transposition (or *shuffling*) ciphers (for example, columnar transposition), that operate by changing the *position* of letters in a text. For example, one could use the following table to define a shuffle on nine positions.

in	1	2	3	4	5	6	7	8	9
out	4	6	1	5	3	7	9	2	8

With this shuffle, `abcdefghi` would encipher to `cheadbfig`, so the first letter in the plaintext (a) goes to the fourth position in the ciphertext, and the first letter in the ciphertext (c) originates from the third position in the plaintext.

(a) ⋆ Decipher `vlooibuys`.

$\boxed{\text{o b v i o u s l y}}$

(b) ⋆ What is the keyspace $\mathcal{K}$ and how large are keys (in bits)?

$$\mathcal{K} = \{\text{permutations of } 1, \ldots, 9\}$$
$$= \boxed{\mathrm{Perm}(\{1, \ldots, 9\})}$$

There are $9!$ possible keys, so we can use $\lceil \lg(9!) \rceil = \boxed{19 \text{ bits}}$ to uniquely identify any key.

(c) ⋆ Find an adversary that distinguishes the shuffling cipher from a random permutation with an overwhelming advantage in a single query and minimal computation. (Calculate its advantage.)

Choose a message of repeating characters, e.g. $m =$ "aaaaaaaaa".

Then define

$$\mathcal{A}(c) = \begin{cases} 1 & \text{if } c \text{ contains all "a"s,} \\ 0 & \text{otherwise.} \end{cases}$$

Then $\Pr[\mathcal{A}(E_k(m)) = 1] = 1$, but

$$\Pr[\mathcal{A}(c^*) = 1] = \left(\frac{1}{26}\right)^9.$$

✓

the odds of a random 9-letter string containing only "a"

Thus $\mathrm{Adv}(\mathcal{A}) = 1 - \left(\frac{1}{26}\right)^{\cancel{8}\,9} \approx 1$.

(d) ⋆ Suppose the adversary is more ambitious than simply distinguishing and wants to recover the key using a chosen plaintext attack. Explain how you would recover the key. Try to maximize the key recovery advantage while minimizing the number of queries and adversarial runtime.

✓

Simply choose an input with all distinct characters, e.g. "abcdefghi".

Then the encryption will be a permutation of these characters.

To recover the key, observe which position each of the original characters is mapped to.

The key recovery advantage is 1.

once we get the position for all 26 chars

3. Shuffling ciphers aren't very good. Another class of historical ciphers is known as *substitution* ciphers, where each letter of the alphabet is substituted by another one. For instance, one could use the following table to define the substitution.

in	`abcdefghijklmnopqrstuvwxyz`
out	`francoiszyxwvutqpmlkjhgedb`

(a) ⋆ Decipher `atvqjkcml`.

$\boxed{\text{computers}}$

(b) ⋆ What is the keyspace $\mathcal{K}$ and how large are keys (in bits)?

$\boxed{\mathcal{K} = \text{Perm}(\{a, b, c, \ldots, z\}).}$

$|\mathcal{K}| = 26!$, so each key can be represented with $\lceil \lg(26!) \rceil = \boxed{89 \text{ bits.}}$

(c) ⋆ Find a distinguishing attach on the substitution cipher. Try to maximize the distinguishing advantage while minimizing the number of queries and adversarial runtime.

We can do the exact same as in problem 2.

Choose a message of repeating characters, e.g. $m =$ "aaaaaaaaa".

Then define

$$A(c) = \begin{cases} 1 & \text{if } c \text{ contains all identical characters,} \\ 0 & \text{otherwise.} \end{cases}$$

Then $\Pr[A(E_k(m)) = 1] = 1$, but for $c^* \xleftarrow{\$} \mathcal{C}$,

$$\Pr[A(c^*) = 1] = \left(\frac{1}{26}\right)^8 = 4.8 \times 10^{-12}.$$

9?

Thus $\text{Adv}(A) = 1 - \left(\frac{1}{26}\right)^8 \approx 1.$

(d) ⋆⋆ Suppose the adversary is more ambitious and wants to recover the key using a chosen plaintext attack. Explain how you would recover the key. Try to maximize the key recovery advantage while minimizing the number of queries and adversarial runtime.

Make 3 queries, each containing a third of the alphabet (except the last will have a repeated character, since $26 = 9 + 9 + 8$).

Then one can observe what letter each character is mapped to, uniquely identifying the key with advantage $= 1$.

4. Shuffling once or substituting once is rubbish as an enciphering mechanism. But can we instead combine both operations, and iterate them a couple of times? Let's consider that we use P_k with $k \in \mathrm{Perm}(\{1, \ldots, 9\})$ to denote a shuffle, and S_k with $k \in \mathrm{Perm}(\{\mathsf{a}, \ldots, \mathsf{z}\})$ to denote a substitution. We can create an enciphering scheme E by composing shuffles and substitutions as follows.

Kg
$k_1 \leftarrow_\$ \mathrm{Perm}(\{1, \ldots, 9\})$
$k_2 \leftarrow_\$ \mathrm{Perm}(\{\mathsf{a}, \ldots, \mathsf{z}\})$
$k_3 \leftarrow_\$ \mathrm{Perm}(\{1, \ldots, 9\})$
$k_4 \leftarrow_\$ \mathrm{Perm}(\{\mathsf{a}, \ldots, \mathsf{z}\})$
return (k_1, k_2, k_3, k_4)

$\mathsf{E}_{(k_1,k_2,k_3,k_4)}(m)$
$c \leftarrow \mathsf{P}_{k_1}(\mathsf{S}_{k_2}(\mathsf{P}_{k_3}(\mathsf{S}_{k_4}(m))))$
return c

(a) ⋆ Explain how deciphering works.

Given a ciphertext c:

I was looking at Kg, not E

1) decrypt under the substitution k_4;
2) then decrypt the result under the shuffle k_3;
3) then the substitution k_2;
4) and finally, decrypting under the shuffle k_1 gives the result.

$$M \leftarrow S^{-1}_{k_4}(P^{-1}_{k_3}(S^{-1}_{k_2}(P^{-1}_{k_1}(c))))$$

(b) ⋆⋆ Argue (don't prove) that the repetition in the enciphering scheme is pointless, so we can consider only a two-key scheme $\mathsf{E}_{k_5,k_6} = \mathsf{P}_{k_5} \circ \mathsf{S}_{k_6}$ instead, without loss of generality (or security).

* should use commutativity and composition

Let's say we start with character c at position i.
Then the result of each transformation is:

- k_1: character c at position i'
- k_2: character c' at position i'
- k_3: character c' at position i''
- k_4: character c'' at position i''.

We can replicate this behavior with a single shuffle sending i → i'', followed by a substitution sending c → c''.

(c) ⋆⋆ Come up with a distinguishing attack.

Same as parts (2) and (3): choose $m =$ "aaaaaaaaa",
and define

$$A(c) = \begin{cases} 1 & \text{if } c \text{ contains all identical characters,} \\ 0 & \text{otherwise.} \end{cases}$$

Then

$$A(E_k(m)) = 1,$$

but for $c^* \xleftarrow{\$} \mathcal{C}$,

$$A(c^*) = \left(\frac{1}{26}\right)^8.$$

Thus $Adv(A) = 1 - \left(\frac{1}{26}\right)^8 \approx 1$.

(d) ⋆⋆⋆ Sticking to the simplification from (b), describe a key recovery attack under chosen plaintext attack that has advantage 1. You don't have to try to minimize the number of queries, but try to avoid exhaustive search, while still being guaranteed to recover the key (k_5, k_6).

We first find K_5 (key for the shuffle) by encrypting 9 strings, each of which consists of identical characters except for a different character at index i. For example,

$m_1 =$ "baaaaaaaa"
$m_2 =$ "aba ... a"
$\vdots$
$m_9 =$ "a ... ab"

Then the output $E_K(m_i)$ will likewise contain a single unique character, at an index j. This tells us that K_5 maps index i to index j.

To find k_6 (the substitution key), we use 26 different plaintexts, each consisting of a different repeated letter of the alphabet (say, α). Then the output will likewise contain a <u>different</u> repeated letter, for example β, which tells us the substitution key maps α to β.

5. Shuffling and substitution on their own, and taken together, are simply not good enough. We throw another ingredient into the mix. The Vigenère cipher is a generalization of Caesar's cipher, where letters of the alphabet are added together, identifying a with 1, b with 2, all the way up to identifying z with 0. Additions are done modulo 26. Given two words of the same length, we can add them letter by letter.

Kg
$k \leftarrow_\$ \{\mathsf{a}, \ldots, \mathsf{z}\}^9$ **return** k

$\mathsf{V}_k(m)$
$c \leftarrow m + k$ **return** c

(a) ⋆ Use Shannon's theorem to demonstrate that Vigenère's scheme is perfectly secret.

Recall:

Theorem 1.2 (Shannon's Theorem). *Let $E = (\mathsf{Kg}, \mathsf{E}, \mathsf{D})$ be an enciphering scheme with $\mathcal{K} = \mathcal{M}$. Then E is perfectly secure iff* Kg *draws from $\mathcal{K}$ uniformly at random and E satisfies that for all (m, c) pairs, there exists a unique key k such that $\mathsf{E}_k(m) = c$.*

Note that K_g draws from $\mathcal{K} = \{a, \ldots, z\}^9$ uniformly at random, so that part is satisfied.

Now consider some $m, c \in \{a, \ldots, z\}^9$.
Denote $m := m_1 m_2 \cdots m_9$ and $c := c_1 \cdots c_9$.
The only key which gives $E_k(m) = c$ is $k = k_1 \cdots k_9$, where
$$k_i = (c_i - m_i) \bmod 26.$$

i.e. $(\{a, \ldots, z\}^9, +)$ is a grp, so $(+)$ is invertible

Thus E is perfectly secure.

As mentioned, we'd like to combine the Vigenère cipher with shuffles and substitutions. The hope is that having three different mechanisms in play will work better than only the two. We first consider whether repetition helps, or whether it is as pointless as with substitutions and shuffles.

(b) ⋆⋆⋆ Argue that when combining Vigenère with shuffles, repetition is pointless. That is, for all k_1, k_2, k_3, k_4, there exist k_5 and k_6 such that

$$\mathsf{P}_{k_5} \circ \mathsf{V}_{k_6} = \mathsf{P}_{k_1} \circ \mathsf{V}_{k_2} \circ \mathsf{P}_{k_3} \circ \mathsf{V}_{k_4}$$

Suppose $k_4 = v_1 \cdots v_q$ and $k_2 = v_1' \cdots v_q'$.
Also, denote P_{k_3} as sending index $p(i)$ to index i and P_{k_1} as sending index $p'(i)$ to index i.
Then for an arbitrary $m = m_1 \cdots m_q \in \mathcal{M}$,

$$(P_{k_1} \circ V_{k_2} \circ P_{k_3} \circ V_{k_4})(m_1 \cdots m_q)$$
$$= (P_{k_1} \circ V_{k_2} \circ P_{k_3})\ (m_1 + v_1) \cdots (m_q + v_q)$$
$$= (P_{k_1} \circ V_{k_2})\ (m_{p(1)} + v_{p(1)}) \cdots (m_{p(q)} + v_{p(q)})$$
$$= P_{k_1}\left((m_{p(1)} + v_{p(1)} + v_1') \cdots (m_{p(q)} + v_{p(q)} + v_q')\right)$$
$$\overset{(\star)}{=} (m_{p'(p(1))} + v_{p'(p(1))} + v'_{p'(1)}) \cdots (m_{p'(p(q))} + v_{p'(p(q))} + v'_{p'(q)})$$

Now define $k_6 = \underbrace{(v_{p(1)} + v_1')}_{\text{first position}} \cdots \underbrace{(v_{p(q)} + v_q')}_{q^{th}\text{ position}}$

and $P_{k_5} = P_{k_1}$.

Then

$$(P_{k_5} \circ V_{k_6})m = P_{k_5}\ (m_1 + v_{p(1)} + v_1') \cdots (m_q + v_{p(q)} + v_q')$$
$$= (m_{p'(p(1))} + v_{p'(p(1))} + v'_{p'(1)}) \cdots (m_{p'(p(q))} + v_{p'(p(q))} + v'_{p'(q)})$$

This is equivalent to $(\star)$ above, so the 2 ciphers are equal.

(c) ⋆⋆ When combining Vigenère with substitutions, repetitions *do* in fact add complexity. However, you can still find a distinguishing attack. Do so.

~~Same as previous questions.~~
~~Choose a repeating character, like $m =$ "aa...a".~~
~~Then Vignère and substitutions, no matter how many are applied, will output a ciphertext with a single repeating character.~~

this isn't true, since Vignère depends on position of each character as well

~~Thus, as above,~~

$$\cancel{\mathcal{A}(E_k(m)) = 1,}$$

~~but for~~ $\cancel{c^* \xleftarrow{\$} \mathcal{C},}$

$$\cancel{\mathcal{A}(c^*) = \left(\frac{1}{26}\right)^8,}$$

~~so~~ $\cancel{\mathrm{Adv}(\mathcal{A}) = 1 - \left(\frac{1}{26}\right)^8 \approx 1.}$

(d) ⋆ ⋆ ⋆ Suppose that a cipher consists of ten repetitions, or rounds, each consisting of a substitution followed by Vigenère—all with independent keys. Describe an efficient chosen plaintext attack that recovers a complete description of the keyed encryption and decryption algorithms. (As lookup tables or functions—this is easier than recovering all 20 subkeys!)
(Hint 1: Which letters of the plaintext does the ith letter of the ciphertext depend on? — Answering this will help you understand how you can recover an algorithm to decrypt without recovering the entire key.)
(Hint 2: 26 chosen plaintexts will suffice. Attacks with a lot less exist for those of you who have nothing better to do.)

abbbbbbbb

↓

bc ... c

↓

def ... z

<u>Observation</u>: $c[i]$ depends only on $m[i]$, since the letters never change places.

Thus we choose 26 plaintexts:
"aa...a", "bb...b", ..., "zz...z".

Then the outputs describe what each letter, in each possible position, becomes.

For example, if a plaintext m starts with "xy..." then $E_K(m)$ will start with $E_K(\text{"xx...x"})_1$ (the first character of the ciphertext), followed by $E_K(\text{"yy...y"})_2$.