Monoalphabetic Substitutions

Klaus Pommerening
Fachbereich Physik, Mathematik, Informatik
der Johannes-Gutenberg-Universität
Saarstraße 21
D-55099 Mainz

October 25, 1999—English version October 5, 2013—last change January 18, 2021

The subject of cryptography is the transformation of character strings.

The simplest example is a monoalphabetic substitution. It replaces each letter of a plaintext by another letter or symbol, depending only on the letter. The same plaintext letter is always replaced by the same ciphertext letter.

Amateurs tend to choose strange symbols as ciphertext letters instead of the characters of the ordinary alphabet `A...Z`. This has absolutely no effect on the security of the cipher but is a first example of an **illusory complication**. See the FAQ webpage: `http://www.staff.uni-mainz.de/pommeren/Cryptology/FAQ.html`

1 Mathematical Model of Cryptography

We want to give a formal definition of the following two concepts:

- *An encryption function transforms arbitrary character strings into other character strings.* (Where the strings are from a given alphabet.)
- *A cipher is a parametrized family of encryption functions. The parameter is called the key.* It determines the choice of a function from the family.

The purpose of this construct is that nobody can invert the encryption function except people who know the key. That is, an encrypted message (or a text, a file ...) is kept secret from third parties. These can see that there is a message, but they cannot read the contents of the message because they don't have the key and therefore don't know which of the functions from the family to invert.

Alphabets and Texts

Let Σ be a finite set, and call it **alphabet**. Call its elements **letters** (or **symbols**, or **characters**).

Examples. Here are some alphabets of cryptographic relevance:

- {A, B, ..., Z}, the standard 26 letter alphabet of classical cryptography.
- The 95 character alphabet of printable ASCII characters from "blank" to "tilde", including punctuation marks, numbers, lowercase, and uppercase letters.
- $\{0,1\} = \mathbb{F}_2$, the alphabet of bits, or the field of two elements. The earliest appearence (after Bauer[1]) is Bacon 1605.
- $\mathbb{F}_2^5$, the alphabet used for telegraphy code since Baudot (1874). It has 32 different symbols and also goes back to Bacon (after Bauer[1]).
- $\mathbb{F}_2^8$, the alphabet of bytes (correctly: octets, because in early computers bytes did not necessarily consist of exactly 8 bits). The earliest appearance seems to be at IBM around 1964.
- More generally $\mathbb{F}_2^l$, the alphabet of l-bit blocks. Often $l = 64$ (for example in DES or IDEA), or $l = 128$ (for example in AES). See Part II (on bitblock ciphers).

Often the alphabet Σ is equipped with a group structure, for example:

- $\mathcal{Z}_n$, the cyclic group of order $n = \#\Sigma$. Often we interpret the calculations in this group as arithmetic $\bmod n$, as in elementary number theory, and denote $\mathcal{Z}_n$ by $\mathbb{Z}/n\mathbb{Z}$, the residue class ring of integers $\bmod n$.
- $\mathbb{F}_2$ with the field addition $+$, as BOOLEan operator often denoted by XOR or $\oplus$. (Algebraists like to reserve the symbol $\oplus$ for direct sums. For this reason we'll rarely use it in the BOOLEan context.)
- $\mathbb{F}_2^l$ as l-dimensional vector space over $\mathbb{F}_2$ with vector addition, denoted by $+$, XOR, or $\oplus$.

For an alphabet Σ we denote by Σ^* the set of all finite sequences from Σ. These sequences are called **texts** (over Σ). A subset $M \subseteq \Sigma^*$ is called a **language** or **plaintext space**, and the texts in M are called meaningful texts or **plaintexts**.

Note that the extreme case $M = \Sigma^*$ is not excluded.

Ciphers

Let K be a set (finite or infinite), and call its elements **keys**.

Definition (i) An **encryption function** over Σ is an injective map $f\colon \Sigma^* \longrightarrow \Sigma^*$.

(ii) A **cipher** (also called encryption system or cryptosystem) over Σ with key space K is a family $F = (f_k)_{k\in K}$ of encryption functions over Σ.

(iii) Let F be a cipher over Σ, and $\tilde{F} = \{f_k | k \in K\} \subseteq \mathrm{Map}(\Sigma^*, \Sigma^*)$ be the corresponding set of different encryption functions. Then $\log_2(\#K)$ is called the **key length**, and $d(F) = \log_2(\#\tilde{F})$, the **effective key length** of the cipher F.

Remarks

1. This is not the most general definition of an encryption function. One could also consider non-injective functions, or even relations that are not functions, or are not defined on all of Σ^*.
2. Strictly speaking, the encryption functions need to be defined only on the plaintext space M, however we almost always consider encryption functions that are defined on all of Σ^*.
3. The encryption functions f_k, $k \in K$, need not be pairwise different. Therefore in general $\#\tilde{F} \leq \#K$, and effective key length $\leq$ key length. If K is infinite, then $\tilde{F}$ can be finite or infinite. In general the key length

is easier to determine than the effective key length, however it is less useful.

4. The elements in the ranges $f_k(M)$ depend on the key k. They are called **ciphertexts**.

5. Note that the identification of the alphabet Σ with the integers $\bmod n$, $\mathbb{Z}/n\mathbb{Z}$, also defines a linear order on Σ. We often implicitely use this order. In some cases for clarity we must make it explicit.

2 Shift Ciphers

Assume that the alphabet is linearly ordered. A shift cipher replaces each letter of the plaintext by the letter that follows a certain number k of positions in the alphabet. If the end of the alphabet is reached, restart at the beginning. That means, we consider cyclic shifts. The number k is the key.

Decryption works in the reverse direction: Count backwards from the ciphertext letter.

Example 1: Original Caesar

Here $\Sigma = \{\text{A,...,Z}\} = \mathcal{Z}_{26}$, hence $A \leftrightarrow 0, B \leftrightarrow 1, ..., Z \leftrightarrow 25$. Caesar used the fixed key $k = 3$. Encryption looks like follows

```
C A E S A R  | +3  (plaintext)
-----------
F D H V D U        (ciphertext)
```

> Note that the original Roman alphabet had only 23 letters without J, U, W. However in this part of the lecture we (almost) always use the 26 letter alphabet.

As key space we could also take $K = \mathbb{Z}$. Then the key length is ∞. But effectively we only have 26 different encryption functions, one of them being trivial. Therefore the effective key length is only $\log_2(26) \approx 4.7$.

Example 2: Rot13

ROT13 is a shift cipher over the alphabet {A, ..., Z} that shifts each letter by 13 positions ahead in the alphabet. As mnemonic take the table

```
A B C D E F G H I J K L M
N O P Q R S T U V W X Y Z
```

As encryption function this is almost useless. Its purpose is hiding some texts, say of offensive content, from immediate recognition. The reader of the message can figure it out only by a conscious act.

Because 13 + 13 = 26, double encryption restores the plaintext. That is, ROT13 is an involution. Or in other words: encryption = decryption as functions.

Example 3: XOR

This example extends the notion of shift cipher towards the more general version given in the mathematical description below. In this sense XOR is a shift cipher on the space of l-bit blocks. Thus our alphabet is the l-dimensional vector space $\mathbb{F}_2^l$ over the two element field $\mathbb{F}_2$. The operation

XOR is the addition of vectors in this space (because XOR of bits is the addition in the field $\mathbb{F}_2$). The key is a fixed block k. Each plaintext block a is XORed with k bitwise, that is, "shifted" (or translated) by k.

Mathematical Description

Let the alphabet Σ be a finite group G with n elements and with group composition $*$. As key space also take $K = G$. For $k \in K$ let

$$f_k : \Sigma^* \longrightarrow \Sigma^*$$

be the continuation of the right translation $f_k(s) = s * k$ for $s \in \Sigma$, that is

$$f_k(a_1, \ldots, a_r) = (a_1 * k, \ldots, a_r * k) \quad \text{for } a = (a_1, \ldots, a_r) \in \Sigma^r.$$

The effective key length is $d(F) = \log_2(n)$. Thus the key space is quite small and is easily completely searched except when n is VERY LARGE. An example will follow in the next section.

3 Cryptanalysis of Shift Ciphers by Exhaustion

General Approach

The most primitive of all cryptanalytic attacks is **exhaustion**, also known as **brute force attack**. It consists of a complete key search—run through the complete key space K, and try key after key until you get a valid decryption. Assume that K is finite (as it is in all practical situations). Then the attacker needs $\#K$ steps in the worst case, and $\#K/2$ steps in the mean. *This method applies to all ciphers.* A precondition for the success is the redundancy of the plaintext language that allows distinguishing between meaningful text and nonsense character sequences. In general the solution is unique as soon as the length of the text exceeds the "unicity distance" of the cipher, see Chapter 10.

For distinguishing between meaningful and meaningless texts, algorithms that compute language statistics may be used, see Chapter 3.

Solving Shift Ciphers

```
FDHVDU
GEIWEV
HFJXFW
IGKYGX
JHLZHY
KIMAIZ
LJNBJA
MKOCKB
NLPDLC
OMQEMD
PNRFNE
QOSGOF
RPTHPG
SQUIQH
TRVJRI
USWKSJ
VTXLTK
WUYMUL
XVZNVM
YWAOWN
ZXBPXO
AYCQYP
BZDRZQ
CAESAR
DBFTBS
ECGUCT
```

This is an example for solving a shift cipher by exhaustion. The first row is the ciphertext from the last section. The following rows contain the candidate plaintexts for each possible key one after the other.

Only the row `CAESAR` makes sense as plaintext. Hence the ciphertext is decrypted and the key is 3.

Note that each column contains the standard alphabet, cyclically continued. From this observation a purely mechanical approach derives: Produce some vertical strips containing the alphabet twice, and arrange them beneath each other in such a way that one row contains the ciphertext. Then scan the other rows for meaningful plaintext.

> Because of this scheme the exhaustion method is sometimes called "generatrix method". This notation comes from an analogy with cipher cylinders, see Chapter 4.

Lessons Learned

1. Shift ciphers are solvable as soon as the attacker has some small amount of ciphertext, at least when the alphabet is not too large and the language is only a small part of all character sequences. (Later we'll express this as "high redundancy" or "low entropy", see Chapter 10.)

2. A cipher should use a large key space (or rather a large effective key length). But bear in mind:

   The effective key length measures the complexity of the exhaustion attack. But in general it is an insufficient measure of the complexity of the cryptanalysis of a cipher.

In other words: In many cases there are more efficient attacks against a cipher than exhaustion.

4 Monoalphabetic Substitution

Introductory Example

The key of a monoalphabetic substition is a permutation of the alphabet, for example:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
UNIVERSTABCDFGHJKLMOPQWXYZ
```

For encryption locate each letter of the plaintext in the first row of this table, and replace it by the letter below it. In our example this becomes:

```
ENGLI SHAST RONOM ERWIL LIAML ASSEL LDISC OVERE DTRIT ON
EGSDA MTUMO LHGHF ELWAD DAUFD UMMED DVAMI HQELE VOLAO HG
```

For decryption we use the inverse permutation, given by the table

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
IJKLEMNOCPQRSBTUVFGHADWXYZ
```

Mathematical Description

Let $\mathcal{S}(\Sigma)$ be the group of permutations of the alphabet Σ, that is the full symmetric group. See Appendix A for an introduction to permutations.

A monoalphabetic substitution consists of the elementwise application of a permutation $\sigma \in \mathcal{S}(\Sigma)$ to texts:

$$f_\sigma(a_1, \ldots, a_r) := (\sigma a_1, \ldots, \sigma a_r) \quad \text{for } (a_1, \ldots, a_r) \in \Sigma^r.$$

Definition A **monoalphabetic cipher** over the alphabet Σ with keyspace $K \subseteq \mathcal{S}(\Sigma)$ is a family $(f_\sigma)_{\sigma \in K}$ of monoalphabetic substitutions.

Examples 1. The shift cipher where $K =$ the set of right translations.

2. The general monoalphabetic cipher where $K = \mathcal{S}(\Sigma)$. Here $\#K = n!$ with $n = \#\Sigma$.

The Effective Key Length

The general monoalphabetic cipher F defeats the exhaustion attack, even with computer help. The $n!$ different keys define $n!$ different encryption functions. Therefore

$$d(F) = \log_2(n!) \geq n \cdot [\log_2(n) - \log_2(e)] \approx n \cdot \log_2(n)$$

by Stirling's formula, see Appendix B. For $n = 26$ we have for example

$$n! \approx 4 \cdot 10^{26}, \quad d(F) \approx \log_2(26!) \approx 88.38.$$

Note that for a ciphertext that doesn't contain all letters of the alphabet the search is somewhat faster because the attacker doesn't need to determine the entire key.

5 Algorithms and Programming in Perl

See the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/1_Monoalph/MonoPerl.html`

6 Cryptanalysis of Monoalphabetic Substitution

General Approach

The cryptanalysis of the monoalphabetic substitution makes use of its *invariants*, that is of properties of a text that remain unchanged under encryption:

1. The distribution of the frequencies of single characters is invariant.
   - This means that a letter in the ciphertext occurs exactly as many times as the corresponding letter in the plaintext.
   - The same is true for bigrams (= pairs of letters), trigrams, ..., n-grams.
2. Repeated patterns in the plaintext show up also in the ciphertext.

Both of these invariant properties suggest cryptanalytic approaches:

1. Statistical analysis
2. Pattern recognition (for example matching with the words of a dictionary)

Often the cryptanalyst combines both of these approaches, and supplements them with systematic guesses:

- *Cryptography is Mathematics.*
- *Cryptanalysis is struggling, using all available aids.*

Only in rare situations cryptanalysis is completely algorithmic. But no matter which method applies and how clean its theoretical basis is, the successful solution legitimates the cryptanalyst.

7 Statistical Analysis of Ciphertext

Character Frequencies

Natural languages such as German, English, Russian, ..., and also artificial languages such as MS-DOS-EXE, ..., Pascal, ..., MS-Word, ..., show typical character frequencies that are

- nonuniformly distributed,
- characteristic for the language.

Texts of about 500 or 1000 letters in a natural language rareley show a significant deviation from the typical frequencies.

This allows automating the cryptanalysis based on letter frequencies to a large extent. The web offers several such programs, for example see the ACA Crypto Dropbox `[http://www.und.nodak.edu/org/crypto/crypto/]`.

Mathematical Model

The simplest mathematical model for statistical analysis of ciphertext is a probability distribution on the underlying (finite) alphabet Σ with atomic probabilities $p(s)$ for all letters $s \in \Sigma$. Thus we assume that plaintexts are streams of independent (but not uniformly distributed) random letters.

A closer approximation to the truth would account for dependencies of letters from their predecessors according to the typical bigram distribution.

There are further possible refinements, for example the most frequent initial letter of a word in English is T, in German, D.

Example: Byte Frequencies in MS-Word Files

Byte	**Frequency**
`00`	ca 7-70%
`01`	ca 0.8-17%
`20` = space	ca 0.8-12%
`65` = e	ca 1-10%
`FF`	ca 1-10%

Observations

- The variability is rather large, unexpected peaks occur frequently.
- The distribution depends on the software version.
- All bytes `00-FF` occur.
- We see long sequences of zero bytes. If the file is encrypted by XOR, large parts of the key shine through.

The last remark yields an efficient method for analysis of the XOR encryption of a WORD file with periodically repeated key. This not exactly a statistical cryptanalysis, it only uses the frequency of a single byte. To start with, pairwise add the blocks. If one of the plaintext blocks essentially consists of zeroes, then the sum is readable plaintext:

Plaintext	...	a_1	...	a_s	...	0	...	0	...
Key (repeated)	...	k_1	...	k_s	...	k_1	...	k_s	...
Ciphertext	...	c_1	...	c_s	...	c'_1	...	c'_s	...

where $c_i = a_i + k_i$ in the first block, and $c'_i = 0 + k_i$ in the second block for $i = 1, ..., s$ (s the blocksize).

Therefore $c_i + c'_i = a_i + k_i + k_i = a_i$,—one block of plaintext revealed and identified—; and $k_i = c'_i$—the key revealed.

If the addition of two cipher text blocks yields a zero block, then with high probability both plaintext blocks are zero blocks (or with small probability are identical nonzero blocks). Also in this case the key is revealed.

8 Example of a Statistical Cryptanalysis

See web pages `http://www.staff.uni-mainz.de/pommeren/Kryptologie/Klassisch/1_Monoalph/Beispiel.html` (in German) or `http://www.staff.uni-mainz.de/pommeren/Kryptologie/Klassisch/0_Unterhaltung/Lit/Goldbug_Crypto.html` (in English)

9 Pattern Search

Word Lists

The second basic approach to cryptanalysis of the monoalphabetic substitution is the search for patterns in the ciphertext that correspond to the patterns of

- supposed words (probable words),
- words from a list.

This method is cumbersome if done by hand but easy with computer support that completely searches lists of several 100000 words in a few seconds.

Searching for a probable word is a variant of pattern search. We search for the pattern of a word that we suspect from knowledge of the context as occuring in the plaintext.

Numerical Patterns for Strings

To normalize letter patterns we describe them by numbers. Here is an example: The word "`statistics`" defines the pattern `1232412451`. The general procedure is: Replace the first letter by 1. Then replace each following letter by

- the number that was assigned to this letter before,
- the next unused number, if the letter occurs for the first time.

Here is a formal definition:

Definition Let Σ be an alphabet. Let $a_1, \ldots, a_q$ be letters from Σ. The **pattern** belonging to the string $(a_1, \ldots, a_q)$ ist the q-tuple $(n_1, \ldots, n_q) \in \mathbb{N}^q$ of numbers that is defined recursively by

- $n_1 := 1$.
- For $k = 2, \ldots, q$:
  If there is an i with $1 \leq i < k$ and $a_k = a_i$, then $n_k := n_i$,
  else $n_k := 1 + \max\{n_i \mid 1 \leq i < k\}$.

Remarks

1. $n_i = n_j \iff a_i = a_j$ for $1 \leq i \leq j \leq q$.
2. $\{n_1, \ldots, n_q\} = [1 \ldots m]$ where $m = \#\{a_1, \ldots, a_q\}$ (= number of different letters in $(a_1, \ldots, a_q)$).

Algorithmic Description

Goal: Determine the numerical pattern of a string.

Input: The string as a list `string =` $(a_1, \ldots, a_q)$.

Output: The numerical pattern as a list `pattern =` $(n_1, \ldots, n_q)$.

Initial value: `pattern = empty list`.

Auxiliary variables:

- n = current number, initial value = 0.
- `assoc =` list of processed letters.
  The index i belongs to the letter `assoc[i]`.
  Initial value: `assoc = empty list`.

Procedure: Loop over the letters in `string`. The current letter is `x`.

If there is an i with `x = assoc[i]`, then append i to `pattern`,

else increment n, append n to `pattern`, append `x` to `assoc`.

For a Perl program that implements this algorithm see the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/1_Monoalph/PattPerl.html`

10 Example of Cryptanalysis by Pattern Search

See the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/1_Monoalph/Puzzle.html`

11 Known Plaintext Attack

See the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/1_Monoalph/knownplain.html`

12 Early History of Cryptology

See the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/1_Monoalph/EarlyHist.html`

13 Variants of Cryptographic Procedures

Some Definitions

Substitution: Letters or groups of letters are replaced by other ones.

Monoalphabetic substitution: Each letter is replaced by another letter that is always the same.

Polyalphabetic substitution: Each letter is replaced—depending on its position in the text—by another letter. (The most important method of classical cryptography in the 20th century up to the sixties)

Monographic substitution: Letters are replaced by symbols one at a time.

Polygraphic substitution: In each step one or more letters are replaced by several symbols.

Homophonic substitution: For some plaintext letters or groups there are several choices of ciphertext symbols.

A mathematical model uses a probability space Ω and considers encryption functions of the type

$$f_k : M \times \Omega \longrightarrow \Sigma^*.$$

This is called **probabilistic encryption**.

Transposition: The letters of the plaintext are permuted.

Codebook: Letter groups of various lengths (for example entire words) are replaced by other ones according to a list. Since the Renaissance this was in use under the denomination **Nomenclator**. It was the most used encryption method even in the 20th Century, especially by diplomats.

Source coding (superencrypted code): The plaintext is transformed with a codebook, and the resulting "intermediate text" is encrypted by some kind of substitution.

Book cipher: Plaintext words or letters are looked up in a certain book. As ciphertext one takes the position of the word or letter in the book, for example page number, line number, number of the word (or number of the letter).

Block cipher: In each step a fixed number of letters is substituted at once.

Stream cipher: In each step a single letter is substituted, each time in another way, depending on its position in the plaintext.

Product cipher: A sequence of several transpositions and block substitutions is applied one after the other (also called cipher cascade).

Polygraphic Substitution

For a fixed l in each step an l-gram (block of l letters) is encrypted at once.

As simplest nontrivial example we consider **bigraphic substitution**. Here pairs of letters are encrypted together. The easiest description of the cipher is by a large square of sidelength $n = \#\Sigma$. An example for the standard alphabet:

	a	**b**	**c**	**d**	**...**
a	CA	FN	BL	...	...
b	SK	WM	...	...	...
c	HP	...	...	...	...
d	...	...	...	...	...
...	...	...	...	...	...

With this table BA is encrypted as SK .

The earliest historical example was given by Porta in 1563. His bigram table however contained strange symbols meeting the spirit of the time. A picture is on the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/1_Monoalph/PortaBi.gif`

Properties of the Polygraphic Substitution

1. The key space of a bigraphic substitution is the set $\mathcal{S}(\Sigma^2)$ of all permutations of the Cartesian product $\Sigma \times \Sigma$. It contains the huge number of $n^2!$ keys. (Of course one also could restrict the keys to a subspace.) The effective keylength is

$$d(F) = \log_2(n^2!) \approx n^2 \cdot \log_2(n^2) = 2 \cdot n^2 \cdot \log_2(n).$$

   For $n = 26$ this amounts to about 4500. Exhaustion surpasses all present or future computer capacity.

2. Compared with a monoalphabetic (and monographic) substitution the frequency distribution of single letters is flattened down. A statistical analysis therefore must resort to bigram frequencies and is a lot harder. Pattern recognition and search for probable words also is harder, but not so much. Also more general attacks with known plaintext are feasible.

3. We may interpret a polygraphic substitution of l-grams as a monographic substitution over the alphabet $\tilde{\Sigma} = \Sigma^l$ of l-grams. The larger

l, the more complicated is the cryptanalysis. However for the *general* polygraphic substitution also the complexity of specifying the key grows with n^l, that is exponentially with l. Therefore this encryption method is useful only with a restricted keyspace. That means we need to fix a class of substitutions $\Sigma^l \longrightarrow \Sigma^l$ whose description is much shorter than the complete value table of n^l entries.

A bigraphic example from history is the PLAYFAIR cipher, invented by WHEATSTONE.

4. Polygraphic substitutions are the predecessors of modern block ciphers.

Codebooks

See the web page `http://www.staff.uni-mainz.de/pommeren/Cryptology/Classic/1_Monoalph/Codebook.html`

References

[1] Bauer, F. L. *Decrypted Secrets; Methods and Maxims of Cryptology.* Berlin: Springer 1997.

[2] Deavours, C. A., Kruh, L. *Machine Cryptography and Modern Cryptanalysis.* Norwood: Artech House 1985.