Rudimentary Cryptanalysis of the Affine Cipher with Known Alphabet Size using the Brute-Force Algorithm

Noel Christoffel Simbolon - 13521096
Computer Science Study Program
School of Electrical Engineering and Informatics
Bandung Institute of Technology, Ganesha Street, no. 10, Bandung, 40132, Indonesia
13521096@std.stei.itb.ac.id

***Abstract*— The affine cipher, a classical encryption technique, is a simple substitution cipher that can be vulnerable to various cryptanalysis techniques. Although it was once considered a secure method for encryption, advancements in cryptanalysis have unveiled its susceptibility to attacks. This research paper delves into the in-depth investigation of the cryptanalysis of the affine cipher using the brute-force algorithm. By exhaustively trying all possible combinations of the multiplier and shift values, the brute-force approach aims to decrypt the ciphertext. We present an implementation of the brute-force algorithm and evaluate its effectiveness in decrypting affine cipher-encrypted text. Additionally, we discuss the theoretical foundation of the affine cipher, describe the implementation and methodology of the brute-force algorithm, and conclude with an analysis of the effectiveness of the approach.**

Keywords—cryptanalysis; affine cipher; brute-force algorithm

I. Introduction

In today's rapidly evolving world, the ubiquity of computing has resulted in an increasing interaction between humans and data [1]. Whether it's managing our financial assets through online banking or completing university assignments using word-processing software, these actions involve the transfer of data over various mediums, and we rely on the security of our data to carry out our activities with confidence [2]. However, ensuring the secure transfer of data is not always guaranteed, as there is a possibility of unwanted recipients gaining access to the transferred information. Therefore, the need for robust security measures, using cryptography, becomes paramount to safeguard our valuable information.

Cryptography has played a crucial role throughout history in protecting sensitive information [3]. From ancient civilizations using substitution ciphers to modern-day cryptographic algorithms, the primary goal has always been to ensure that only authorized parties can access and understand the contents of a message. With the advent of computers and the exponential growth of digital data, the development of more sophisticated encryption techniques has become necessary to safeguard our valuable information from malicious actors and unauthorized access.

As technology advances, the use of cryptography becomes even more significant. With the ever-increasing volume of data transmitted over networks, the potential for interception and unauthorized access grows. Encryption algorithms, including the affine cipher, form a critical line of defense, ensuring the confidentiality and integrity of our data. By gaining a deep understanding of these encryption methods, including their vulnerabilities and cryptanalysis techniques, we can continuously improve and adapt cryptographic systems to stay one step ahead of potential adversaries.

The affine cipher operates on the mathematical principle of modular arithmetic, providing a simple yet effective encryption method. It involves two parameters: a shift value (also known as the key) and a multiplier. These parameters are used to encrypt each character of the plaintext by substituting it with another character from the alphabet. The resulting ciphertext preserves the alphabetical order of the original plaintext, but the individual characters are transformed according to the chosen shift value and multiplier. The technical details of the affine cipher will be presented in detail in the second section of this paper.

While the affine cipher can offer a certain level of security, it is susceptible to cryptanalysis, particularly through the application of the brute-force algorithm. In the case of the affine cipher, this algorithm involves exhaustively searching through all possible shift values and multipliers to find the correct decryption key.

In this research paper, we aim to investigate the cryptanalysis of the affine cipher using the brute-force algorithm. Our objective is to develop an implementation that can effectively decrypt a ciphertext encrypted using the affine cipher with a known alphabet size. By conducting cryptanalysis of the affine cipher through the implementation of the brute-force algorithm, this research aims to contribute to the advancement of cryptanalysis techniques and foster the development of more robust encryption algorithms.

II. THEORETICAL FOUNDATION

A. Simple Substitusion Cipher

By definition, a simple substitution cipher is a cipher in which each letter is replaced by another letter (or some other type of symbol). The Caesar cipher is an example of a simple substitution cipher, but there are many simple substitution ciphers other than the Caesar cipher. In fact, a simple substitution cipher may be viewed as a rule or function

$$\{a, b, c, d, e, \dots, x, y, z\} \to \{A, B, C, D, E, \dots, X, Y, Z\}$$

assigning each plaintext letter in the domain a different ciphertext letter in the range [4]. We write the plaintext using lowercase letters and the ciphertext using uppercase letters for better distinguishability.

To understand this concept better, take a look at Table I below.

Table I. Simple substitution cipher table

Plain	Encrypted
a	C
b	I
c	S
d	Q
e	V
f	N
g	F
h	O
i	W
j	A
k	X
l	M
m	T
n	G
o	U
p	H
q	P
r	B
s	K
t	L
u	R
v	E
w	Y
x	D
y	Z
z	J

Now, suppose that we receive the encrypted message

TVVLTVCLLOVHCBX

and that we know that it was encrypted using Table I. We can reverse the encryption process by finding each ciphertext letter in the second column of Table I and writing down the corresponding letter from the top row, revealing the message.

MEETMEATTHEPARK

which can be interpreted as “meet me at the park.”

B. Divisibility and Greatest Common Divisors

Much of classical cryptography is built on the foundations of algebra and number theory. So, before we explore further we shall discuss important terms and definitions of these mathematical concepts.

Divisibility — Let a and b be integers with $b \neq 0$. We say that b divides a, or that a is divisible by b, if there is an integer c such that

$$a = bc.$$

We write $b \mid a$ to indicate that b divides a. If b does not divide a, then we write $b \nmid a$.

Greatest Common Divisor — A *common divisor* of two integers a and b is a positive integer d that divides both of them. The *greatest common divisor* of a and b is, as its name suggests, the largest positive integer d such that $d \mid a$ and $d \mid b$. The greatest common divisor of a and b is denoted $gcd(a, b)$. If there is no possibility of confusion, it is also sometimes denoted by (a, b). (If a and b are both 0, then $gcd(a, b)$ is not defined.)

Division With Remainder — Let a and b be positive integers. Then we say that a *divided by* b *has quotient* q *and remainder* r if

$$a = b \cdot q + r \quad with\ 0 \leq r < b.$$

The values of q and r are uniquely determined by a and b.

The Euclidean Algorithm — Let a and b be positive integers with $a \geq b$. The following algorithm computes $gcd(a, b)$ in a finite number of steps.

(1) $Let\ r_0 = a\ and\ r_1 = b.$

(2) $Set\ i = 1.$

(3) $Divide\ r_{i+1} by\ r_i\ to\ get\ a\ quotient\ q_i\ and\ remainder\ r_{i+1},$

$$r_{i-1} = r_i \cdot q_i + r_{i+1} \quad with\ 0 \leq r_{i+1} < r_i.$$

(4) $If\ the\ remainder\ r_{i+1} = 0, then\ r_i = \gcd(a, b)\ and\ the\ algorithm\ nates.$

(5) $Otherwise, r_{i+1} > 0, so\ set\ i = i + 1\ and\ go\ to\ Step\ 3.$

The division step (Step 3) is executed at most

$$2 \log_2(b) + 2\ times.$$

Extended Euclidean Algorithm — Let a and b be positive integers. Then the equation

$$au + bv = \gcd\ (a, b)$$

always has a solution in integers u and v.

Coprime Integers — Let a and b be integers. We say that a and b are *coprime* (or *relatively prime*) if $\gcd(a, b) = 1$.

C. Modular Arithmetic and Prime Numbers

Understanding modular arithmetic is essential to grasp the inner workings of the affine cipher. It enables us to handle wraparound behavior within the alphabet's range and express key operations using modular equations. This section outlines the basic concepts of the modular arithmetics needed to understand the affine cipher.

Congruence — Let $m \geq 1$ be an integer. We say that the integers a and b are *congruent modulo* m if their difference $a - b$ is divisible by m. We write

$$a \equiv b \ (mod\ m)$$

to indicate that a and b are congruent modulo m. The number m is called the *modulus*.

For example, we have

$$17 \equiv 7 \ (mod\ 5),$$

since 5 divides $10 = 17 - 7$. On the other hand,

$$19 \not\equiv 6 \ (mod\ 11),$$

since 11 does not divide $13 = 19 - 6$.

Notice that the numbers satisfying

$$a \equiv 0 \ (mod\ m)$$

are the numbers that are divisble by m, i.e., the multiples of m.

Prime Numbers — An integer p is called a prime if $p \geq 2$ and if the only positive integers dividing p are 1 and p.

The Fundamental Theorem of Arithmetic — Let $a \geq 2$ be an integer. Then a can be factored as a product of prime numbers

$$a = p_1^{e_1} \cdot p_2^{e_2} \cdot p_3^{e_3} \cdots p_r^{e_r}.$$

Further, other than rearranging the order of the primes, this factorization into prime powers is unique [4].

Modular Multiplicative Inverse — A modular multiplicative inverse of an integer a is an integer x such that the product ax is congruent to 1 with respect to the modulus m. In the standard notation of modular arithmetic this congruence is written as

$$ax \equiv 1 \ (mod\ m),$$

which is the shorthand way of writing the statement that m divides (evenly) the quantity $ax - 1$, or, put another way, the remainder after dividing ax by the integer m is 1.

D. The Affine Cipher

The affine cipher is a type of simple substitution cipher, where each letter in an alphabet is mapped to its numeric equivalent, encrypted using a simple mathematical function, and converted back to a letter. The formula used means that each letter encrypts to one other letter, and back again, meaning the cipher is essentially a standard substitution cipher with a rule governing which letter goes to which [5].

The key for an affine cipher consists of two integers $k = (k_1, k_2)$. The encryption function is defined by

$$e_k(m) \equiv k_1 \cdot m + k_2 \ (\text{mod}\ p).$$

where modulus p is the size of the alphabet. The value a must be chosen such that k_1 and p are coprime. Here, m can be interpreted as the plaintext that is to be encrypted by the cipher. The value of m is the numerical equivalent of the given plaintext letter.

The decryption function is defined by

$$d_k(c) \equiv k_1' \cdot (c - k_2) \ (\text{mod}\ p)$$

Where k_1' is the modular multiplicative inverse of k_1 modulo p, i.e., it satisfies the equation

$$1 = k_1 \cdot k_1' \ (mod\ p).$$

The multiplicative inverse of k_1 only exists if k_1 and p are coprime. Hence without the restriction on k_1, decryption might not be possible. It can be shown as follows that decryption function is the inverse of the encryption function.

$$\begin{aligned} & d_k\big(e_k(m)\big) = k_1' \cdot (e_k(m) - k_2) \ (mod\ p) \\ & = k_1' \cdot ((k_1 \cdot m + k_2 \ (\text{mod}\ p)) - k_2) \ (mod\ p) \\ & = k_1' \cdot (k_1 \cdot m + k_2 - k_2) \ (mod\ p) \\ & = k_1' \cdot k_1 \cdot m \ (mod\ p) \\ & = m \ (mod\ p) \end{aligned}$$

To understand the affine cipher better, let's take a look at the following example where the alphabet is in English which has the letters A through Z. The letters have the corresponding numberical values found in the following table.

Table II. Numerical values of the letters in the English alphabet

Letter	Numerical Value (m)
a	0
b	1
c	2
d	3
e	4
f	5
g	6
h	7
i	8
j	9
k	10
l	11
m	12
n	13
o	14
p	15
q	16
r	17
s	18
t	19
u	20
v	21
w	22
x	23
y	24
z	25

In this example, the plaintext to be encrypted is "algorithm" using the table mentioned above for the numeric values of each letter, taking k_1 to be 7, k_2 to be 5, and p to be 26 since there are 26 characters in the alphabet being used. The two values k_1 and k_2 can be thought of the multiplier and the amount of character to shift, respectively. Only the value of k_1 has a restriction since it has to be coprime with 26. The possible values that k_1 could be are $1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23,$ and 25. The value for k_2 can be arbitrary as long as k_1 does not equal 1 since this is the shift of the cipher. Thus, the encryption function for this example will be $e_k(m) \equiv 7m + 5 \pmod{26}$.

The table below shows the complete table for encrypting the message in the affine cipher.

Table III. Affine cipher table for encrypting "algorithm"

Plaintext	m	$7m + 5$	$e_k(m)$	Ciphertext
a	0	5	5	F
l	11	82	4	E
g	6	47	21	V
o	14	103	25	Z
r	17	124	20	U
i	8	61	9	J
t	19	138	8	I
h	7	54	2	C
m	12	89	11	L

Next, we will try to decrypt the ciphertext from the above example. The corresponding decryption function is $d_k(c) \equiv 15(c - 5) \pmod{26}$, where k_1' is calculated to be 15, and k_2 is 5.

The table below shows the complete table for decrypting the message in the affine cipher.

Table IV. Affine cipher table for decrypting "FEVZUJICL"

Ciphertext	c	$15(c - 5)$	$d_k(m)$	Plaintext
F	5	0	0	a
E	4	-15	11	l
V	21	240	6	g
Z	25	500	14	o
U	20	225	17	r
J	9	60	8	i
I	8	45	19	t
C	2	-45	7	h
L	11	90	12	m

E. The Brute-Force Algorithm

Brute force is a straightforward approach to solving a problem, usually directly based on the problem statement and definitions of the concepts involved.

The "force" implied by the strategy's definition is that of a computer and not that of one's intellect. "Just do it!" would be another way to describe the prescription of the brute-force approach. And often, the brute-force strategy is indeed the one that is easiest to apply [6].

Exhaustive Search — Many important problems require finding an element with a special property in a domain that grows exponentially (or faster) with an instance size. Typically, such problems arise in situations that involve—explicitly or implicitly—combinatorial objects such as permutations, combinations, and subsets of a given set. Many such problems are optimization problems: they ask to find an element that maximizes or minimizes some desired characteristic such as a path length or an assignment cost.

Exhaustive search is simply a brute-force approach to combinatorial problems. It suggests generating each and every element of the problem domain, selecting those of them that satisfy all the constraints, and then finding a desired element (e.g., the one that optimizes some objective function). Note that although the idea of exhaustive search is quite straightforward, its implementation typically requires an algorithm for generating certain combinatorial objects.

In the context of this research, we use exhaustive search by trying all possible combinations of the multiplier and shift values, i.e., the key to the affine cipher $k = (k_1, k_2)$ with known alphabet size p.

III. Implementation, Cryptanalysis, and Evaluation

Throughout this section, we shall overview of the brute-force implementation written in the Python programming language. Additionally, we will perform cryptanalysis using the implementation. The source code can be found in the remote Git repository linked here.

A. Brute-Force Algorithm Implementation

The implementation of the brute-force algorithm is pretty straightforward. The systematic approach is outlined below.

(1) The program takes three inputs: the ciphertext to be decrypted, a path to a word bank file, and an integer p representing the alphabet size.

    The word bank file contains valid words for the alphabet used in the plaintext. These words are useful to rank the possible plaintext and to determine the accepted decrypted solution.

(2) The program then generates all possible combinations of the multiplier and shift values, i.e., the key to the affine cipher $k = (k_1, k_2)$ with known alphabet size p.

(3) For each key combination, the program decrypts the ciphertext generating possible plaintext.

(4) The possible plaintext is ranked by calculating scores based on the number of valid words, determined through membership testing against the word bank.

(5) The program finds the highest score and its corresponding plaintext and accepts it as the solution. If there is more than one plaintext with the highest score, all of them are accepted as the solution.

B. Cryptanalysis

In this section, we shall perform cryptanalysis of the affine cipher using the implemented brute-force algorithm. Suppose that we are given the following ciphertext that is known to be encrypted using the affine cipher and is originally written in English.

```
FQQHI LZWZX YZFYD ELQOO HQQDI LMPWL MIOZW
WFZXQ DQGWL FQDIL MBQYD ILMFW KJZIL MFDQY
AILMF DQYAO LWAWD ZYPQV QDFYD QFZWF DQYAJ
QBWDQ JKZZX QOIPQ LUQGY OKLJD WEQLY LFZXQ
OZIPP LQOOM YVQLW ZWEQL YLFZX QWLPC GWDFZ
XQDQO HWEQL GYOZX QGXIO HQDQF GWDFP QLWDQ
ZXIOI GXIOH QDQFY LFYLQ UXWAK DAKDQ FJYUE
ZXQGW DFPQL WDQAQ DQPCZ XIOYL FLWZX ILMAW
DQJYU EILZW ZXQUX YAJQD ZKDLI LMYPP ACOWK
PGIZX ILAQJ KDLIL MOWWL YMYIL IXQYD FYZYH
HILMO WAQGX YZPWK FQDZX YLJQB WDQOK DQPCO
YIFIO KDQPC ZXYZI OOWAQ ZXILM YZACG ILFWG
PYZZI UQPQZ AQOQQ ZXQLG XYZZX QDQYZ IOYLF
ZXIOA COZQD CQRHP WDQPQ ZACXQ YDZJQ OZIPP
YAWAQ LZYLF ZXIOA COZQD CQRHP WDQZI OZXQG
ILFYL FLWZX ILMAW DQWHQ LXQDQ IBPKL MZXQO
XKZZQ DGXQL GIZXA YLCYB PIDZY LFBPK ZZQDI
LZXQD QOZQH HQFYO ZYZQP CDYVQ LWBZX QOYIL
ZPCFY COWBC WDQLW ZZXQP QYOZW JQIOY LUQAY
FQXQL WZYAI LKZQO ZWHHQ FWDOZ YCQFX QJKZG
IZXAI QLWBP WDFWD PYFCH QDUXQ FYJWV QACUX
YAJQD FWWDH QDUXQ FKHWL YJKOZ WBHYP PYOTK
OZYJW VQACU XYAJQ DFWWD HQDUX QFYLF OYZYL
FLWZX ILMAW DQZXQ LZXIO QJWLC JIDFJ QMKIP
ILMAC OYFBY LUCIL ZWOAI PILMJ CZXQM DYVQY
LFOZQ DLFQU WDKAW BZXQU WKLZQ LYLUQ IZGWD
QZXWK MXZXC UDQOZ JQOXW DLYLF OXYVQ LZXWK
IOYIF YDZOK DQLWU DYVQL MXYOZ PCMDI AYLFY
LUIQL ZDYVQ LGYLF QDILM BDWAZ XQLIM XZPCO
XWDQZ QPPAQ GXYZZ XCPWD FPCLY AQIOW LZXQL
IMXZO HPKZW LIYLO XWDQS KWZXZ XQDYV QLLQV
QDAWD QAKUX IAYDV QPPQF ZXIOK LMYIL PCBWG
PZWXQ YDFIO UWKDO QOWHP YILPC ZXWKM XIZOY
LOGQD PIZZP QAQYL ILMPI ZZPQD QPQVY LUCJW
DQBWD GQUYL LWZXQ PHYMD QQILM ZXYZL WPIVI
LMXKA YLJQI LMQVQ DCQZG YOJPQ OOQFG IZXOQ
QILMJ IDFYJ WVQXI OUXYA JQDFW WDJID FWDJQ
YOZKH WLZXQ OUKPH ZKDQF JKOZY JWVQX IOUXY
AJQDF WWDGI ZXOKU XLYAQ YOLQV QDAWD Q
```

With that information, we can use the alphabet size $p = 26$ and the English word bank to decrypt the ciphertext using the implemented brute-force algorithm. The following text is the accepted plaintext solution as an output of the program.

```
DEEPI NTOTH ATDAR KNESS PEERI NGLON GISTO
ODTHE REWON DERIN GFEAR INGDO UBTIN GDREA
MINGD REAMS NOMOR TALEV ERDAR EDTOD REAMB
EFORE BUTTH ESILE NCEWA SUNBR OKENA NDTHE
STILL NESSG AVENO TOKEN ANDTH EONLY WORDT
HERES POKEN WASTH EWHIS PERED WORDL ENORE
THISI WHISP EREDA NDANE CHOMU RMURE DBACK
THEWO RDLEN OREME RELYT HISAN DNOTH INGMO
REBAC KINTO THECH AMBER TURNI NGALL MYSOU
LWITH INMEB URNIN GSOON AGAIN IHEAR DATAP
PINGS OMEWH ATLOU DERTH ANBEF ORESU RELYS
AIDIS URELY THATI SSOME THING ATMYW INDOW
LATTI CELET MESEE THENW HATTH EREAT ISAND
THISM YSTER YEXPL ORELE TMYHE ARTBE STILL
AMOME NTAND THISM YSTER YEXPL ORETI STHEW
INDAN DNOTH INGMO REOPE NHERE IFLUN GTHES
HUTTE RWHEN WITHM ANYAF LIRTA NDFLU TTERI
NTHER ESTEP PEDAS TATEL YRAVE NOFTH ESAIN
TLYDA YSOFY ORENO TTHEL EASTO BEISA NCEMA
DEHEN OTAMI NUTES TOPPE DORST AYEDH EBUTW
ITHMI ENOFL ORDOR LADYP ERCHE DABOV EMYCH
AMBER DOORP ERCHE DUPON ABUST OFPAL LASJU
STABO VEMYC HAMBE RDOOR PERCH EDAND SATAN
DNOTH INGMO RETHE NTHIS EBONY BIRDB EGUIL
INGMY SADFA NCYIN TOSMI LINGB YTHEG RAVEA
NDSTE RNDEC ORUMO FTHEC OUNTE NANCE ITWOR
ETHOU GHTHY CREST BESHO RNAND SHAVE NTHOU
ISAID ARTSU RENOC RAVEN GHAST LYGRI MANDA
NCIEN TRAVE NWAND ERING FROMT HENIG HTLYS
HORET ELLME WHATT HYLOR DLYNA MEISO NTHEN
IGHTS PLUTO NIANS HOREQ UOTHT HERAV ENNEV
ERMOR EMUCH IMARV ELLED THISU NGAIN LYFOW
LTOHE ARDIS COURS ESOPL AINLY THOUG HITSA
NSWER LITTL EMEAN INGLI TTLER ELEVA NCYBO
REFOR WECAN NOTHE LPAGR EEING THATN OLIVI
NGHUM ANBEI NGEVE RYETW ASBLE SSEDW ITHSE
EINGB IRDAB OVEHI SCHAM BERDO ORBIR DORBE
ASTUP ONTHE SCULP TURED BUSTA BOVEH ISCHA
MBERD OORWI THSUC HNAME ASNEV ERMOR E
```

Upon inspection, it seems like the plaintext is taken from the famous poem “The Raven” by Edgar Allan Poe. The following text is the original piece of the poem.

> Deep into that darkness peering, long I stood there wondering, fearing,
> Doubting, dreaming dreams no mortal ever dared to dream before;
> But the silence was unbroken, and the stillness gave no token,
> And the only word there spoken was the whispered word, "Lenore?"
> This I whispered, and an echo murmured back the word, "Lenore!"—
> Merely this and nothing more.
>
> Back into the chamber turning, all my soul within me burning,
> Soon again I heard a tapping somewhat louder than before.
> "Surely," said I, "surely that is something at my window lattice;
> Let me see, then, what thereat is, and this mystery explore—
> Let my heart be still a moment and this mystery explore;—
> 'Tis the wind and nothing more!"
>
> Open here I flung the shutter, when, with many a flirt and flutter,
> In there stepped a stately Raven of the saintly days of yore;
> Not the least obeisance made he; not a minute stopped or stayed he;
> But, with mien of lord or lady, perched above my chamber door—
> Perched upon a bust of Pallas just above my chamber door—
> Perched, and sat, and nothing more.
>
> Then this ebony bird beguiling my sad fancy into smiling,
> By the grave and stern decorum of the countenance it wore,
> "Though thy crest be shorn and shaven, thou," I said, "art sure no craven,
> Ghastly grim and ancient Raven wandering from the Nightly shore—
> Tell me what thy lordly name is on the Night's Plutonian shore!"

Quoth the Raven "Nevermore."

Much I marvelled this ungainly fowl to hear discourse so plainly,
Though its answer little meaning—little relevancy bore;
For we cannot help agreeing that no living human being
Ever yet was blessed with seeing bird above his chamber door—
Bird or beast upon the sculptured bust above his chamber door,
With such name as "Nevermore."

Next, we shall perform another cryptanalysis on the following ciphertext. The ciphertext is known to be encrypted using the affine cipher and is originally written in English.

```
GHLFB OGNBY FBMZN RYZF
```

Using the alphabet size $p = 26$ and the English word bank, we decrypt the ciphertext using the implemented brute-force algorithm. The program generates 56 possible plaintext for that particular ciphertext. After closer examination, the most plausible solution is the following plaintext,

```
DOGSA NDCAT SAREC UTE
```

which can most likely be interpreted as "dogs and cats are cute."

Once again, we will conduct a cryptanalysis on the following ciphertext, using the same known information as in the previous cryptanalysis.

```
GHLFB OGNBY FBMZB OXVBC F
```

The following text is the accepted plaintext solution as an output of the program.

```
PSEMA NPKAR MAHUA NOIAD M
```

Unfortunately, it appears that the accepted solution above does not hold any meaning.

C. Brute-Force Algorithm Evaluation

The effectiveness of the brute-force algorithm implementation was evaluated in terms of successfully decrypting the ciphertext. Through cryptanalysis, it was observed that the implementation works better when the input ciphertext is large. This improved performance can be attributed to the scoring system employed by the decrypter. With a larger ciphertext, there is a higher likelihood of obtaining valid words during the decryption process, resulting in more accurate scoring and better identification of the correct plaintext. Thus, the decrypter's effectiveness might be positively correlated with the size of the ciphertext.

Furthermore, we found that preserving the spacing of the plaintext in the ciphertext can enhance the performance of the brute-force algorithm used in the affine cipher decrypter. When the spacing is preserved, the decrypter can leverage the word patterns and word lengths to its advantage. By maintaining the original spacing, the decrypter can identify and group together potential words, which aids in generating meaningful decrypted text. This insight highlights the importance of considering the spacing and formatting of the plaintext during the encryption process to maximize the effectiveness of the decrypter's brute-force algorithm.

IV. CONCLUSION

The implementation and evaluation of the affine cipher decrypter using a brute-force algorithm have provided valuable insights into its effectiveness in decrypting ciphertext. Through cryptanalysis, it was observed that the decrypter performs better when operating on larger ciphertext inputs. This can be attributed to the scoring system employed, which benefits from a larger pool of potential valid words, resulting in more accurate scoring and improved identification of the correct plaintext. Furthermore, preserving the spacing of the plaintext in the ciphertext proved to enhance the decrypter's performance, allowing for better utilization of word patterns and lengths during the decryption process.

V. ACKNOWLEDGMENT

This paper would not have been possible without the opportunity given by my lecturer, Dr. Nur Ulfa Maulidevi, S.T., M.Sc. Her enthusiasm for educating her students motivated me to understand better and explore the brute-force algorithm further. I would also like to thank Dr. Ir. Rinaldi Munir, M.T., the lecturer of the Cryptography course in my university. His extraordinary work has inspired me to pick the affine cipher as the main subject of this paper.

VI. APPENDIX

The Python implementation of the brute-force algorithm overviewed in this paper is available in the remote Git repository linked here. The code is open source under the MIT license.

REFERENCES

[1] R. Mortier, H. Haddadi, T. Henderson, D. McAuley, and J. Crowcroft, "Human-Data Interaction: The Human Face of the Data-Driven Society." arXiv, Jan. 06, 2015. Available: http://arxiv.org/abs/1412.6159

[2] N. C. Simbolon, "An Overview on Reed-Solomon Error-Correcting Code and Its Implementation for File Recovery," 2022.

[3] S. Singh, *The code book: how to make it, break it, hack it, crack it*. New York: Delacorte Press, 2002.

[4] J. Hoffstein, J. Pipher, and J. H. Silverman, *An Introduction to Mathematical Cryptography*. in Undergraduate Texts in Mathematics. New York, NY: Springer New York, 2014. doi: 10.1007/978-1-4939-1711-2.

[5] M. Kozdron, "Affine Ciphers, Decimation Ciphers, and Modular Arithmetic." 2006. [Online]. Available: https://pi.math.cornell.edu/~kozdron/Teaching/Cornell/135Summer06/Handouts/affine.pdf

[6] A. Levitin, *Introduction to the design & analysis of algorithms*, 3rd ed. Boston: Pearson, 2012.

STATEMENT OF ORIGINALITY

I hereby declare that this paper is my own writing, not an adaptation, or translation of someone else's paper, and not plagiarized.

Bandung, May 22nd, 2023

Noel Christoffel Simbolon