A stroll through the history of encryption

Johanan Ottensooser[a] and Matthew Stichinsky[b]
Cornell Tech, 111 8th Avenue, New York, NY 10011

This paper will discuss several ciphers in their historical context. Where it is within the authors' skill level, the cipher will be reduced to python code (available here: *https : //github.com/oatsandsugar/crypto_play/*); where it isn't, the cipher will be described mathematically; where even that escapes us, it will be described through narrative. We intend to describe the strength and weaknesses of each cipher, what led to their development and obsolescence, and any notable stories of their use. This paper will discuss:

1. **Atbash cipher, a simple monoalphabetic substitution cipher with a keyspace of 1;**
2. **Scytale cipher, a simple transposition cipher with a small practical keyspace;**
3. **Caesar cipher, a simple monoalphabetic substitution cipher with a keyspace the size of the alphabet used;**
4. **Vigeǹere cipher, a more complex polyalphabetic substitution cipher with complexity increasing with the size and randomness of the key;**
5. **one time pads, a specific subset of Vigeǹere ciphers still in use, that are not crackable;**
6. **Enigma machine cipher, a complex polyalphabetic substitution cipher; and**
7. **RSA cipher, a complex polyaplphabetic substitution cipher that employs public private key encryption.**

While this is not a complete list of cryptographic developments, it allows us to highlight several important developments in cryptography.

[a] Tech LLM Candidate '17: Cornell Tech; Bachelor of Law and Bachelor of Business (Economics): University of Technology, Sydney
[b] Tech LLM Candidate '17

I. Introduction

This paper intends to highlight several important developments in cryptography, culminating in a description of cryptographic rigor. In order to achieve this goal, this paper will encode certain ciphers in a uniform manner. This will allow the comparison of different ciphers by certain fields in their description, and a description of their strengths and weaknesses.

Where we have been able to, we reduce each cipher to python code[13].

II. Defining cryptography

There are two central mechanisms for hiding messages: *steganography* and *cryptography*.

Steganography is about literally concealing messages. It has been used since classical times (for example, messages written on the wood of a wax tablet holding a false message; tattoos on the head of a slave whose hair has been allowed to grow back) and is used to this day (for example, messages encoded into insignificant digits of .jpeg files, or almost invisible yellow ink dots automatically printed that allow a knowing observer to identify the printer was used). The defining characteristic of this is about obscuring an image—once the image is found, it is plain to read.

Cryptography developed later—earliest records are ancient Egyptian, Greek and Hebrew usages (from which the Atbash cipher and the Caesar cipher, purportedly originate), and continue to be developed and used to the present day.

This paper will focus on the second: *cryptography*.

A. Elements of a cipher

To understand a cryptographic cipher, it is essential to understand its elements. This will allow for the easiest comparison between ciphers. An encryption system comprises the features in Figure 1.

B. What makes a good cipher

A strong cipher is one that fulfills the requirements of its creation, be it to prove authorship of a message or to prevent comprehension of the message by an unintended viewer. A cipher that is not

m	$m \in M$	message m, within a message space M
$gen()$	$k \longleftarrow \text{Gen } s.t. k \in K$	key generating function: generates a key k within a keyspace K
k	$k \in K$	key generated by gen
$enc()$	$c \longleftarrow enc_k(m)$ for $k \in K, m \in M$	encryption function: generates cipher-text c with inputs k and m
c	$gen_k(m) = c$	cipher-text
$dec()$	$\forall m \in M, k \in K \longrightarrow dec_k(enc_k(m)) = m$	decryption function, returns m with inputs k and c

Fig. 1 The elements of a cipher

compromised is a good cipher, and if it can do so economically and with the least possible friction to the users, it is even more so. For the sake of shorthand, we will consider *Kerchoff's principle*: a cipher should be secure even if $enc()$ and $dec()$ are known, and only k remains hidden. This paper will analyse the relevant ciphers against this standard.

III. The Atbash cipher

The Atbash cipher is purported to have been used in the old testament, as well as by ancient Israelites. It is a monoalphabetic substitution cipher where the substitution alphabet is the reverse of the plaintext alphabet:

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

Fig. 2 The Atbash cipher

The cipher is named for its application in the Hebrew alphabet, where the initial letter "aleph" is replaced by "tav" and "bet" is replaced by "shin": giving "Atbash".

A. The Atbash cipher in Python

1. *Variables*

```
m = "" # this can be any string of ASCII characters, of any length

k = # unused in this cipher

m_ord = [] # this is m transformed into a list of integers

c_ord = []
```

```
c = ""
```

Note, the variables are identical in most ciphers discussed, and will only be discussed for subsequent ciphers if interesting, or if such discussion forms part of the criticism of that cipher.

2. The encryption function

```
def enc(z):
    for x in z:
        c_ord.append(27 - x + 64)
enc(m_ord)
```

3. The decryption function

```
def dec(z):
  for x in z:
    m_ord.append(27 - x + 64)
dec(c_ord)
```

B. Criticism of the Atbash cipher

The Atbash cipher is an extremely simple cipher that, without question, fails if *Kerchoff's Principle* is considered. Since there is a keyspace of 1, if the encryption and decryption algorithms are known, then it is trivial to decode the ciphertext.

IV. The Caesar cipher

The Caesar cipher is purported to have been used extensively in Roman times. Similar to the Atbash cipher, the Caesar cipher is a monoalphabetic substitution cipher. The Caesar cipher shifts each letter of the alphabet by a given number of places:

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

Fig. 3 The Caesar cipher for $k = 4$

The Caesar cipher was named after its use (as explained by Suetonius in "Life of Julius Caesar") by Julius Caesar: *"If he had anything confidential to say, he wrote it in cipher, that is, by so changing the order of the letters of the alphabet, that not a word could be made out. If anyone wishes to decipher these, and get at their meaning, he must substitute the fourth letter of the alphabet, namely D, for A, and so with the others."*

V. Scytale cipher

Unlike the above ciphers, the Scytale cipher is a transposition cipher (c.f. a substitution cipher). Substitution ciphers replace single letters with other letters. Transposition ciphers obscure meaning by reordering the message text to create the ciphertext. These were one of the earliest forms of ciphers used, and were reportedly used by the Ancient Greeks, including, notably, the Spartan army.

The key length is the period of the cipher, with the message reordered into ciphertext by taking the first, and then the $n * k$th letter of the message, circling back to the start when necessary and continuing the same function:

message	m	a	k	e	y	o	u	r	m	e	s	s	a	g	e	s	e	c	u	r	e
key line 1	m	→	→	→	y	→	→	→	m	→	→	→	a	→	→	→	e	→	→	→	e
key line 2	↪	a	→	→	→	o	→	→	→	e	→	→	→	g	→	→	→	c	→	→	→
key line 3	↪	→	k	→	→	→	u	→	→	→	s	→	→	→	e	→	→	→	u	→	→
key line 4	↪	→	→	e	→	→	→	r	→	→	→	s	→	→	→	s	→	→	→	r	●
ciphertext	M	Y	M	A	E	E	A	O	E	G	C	K	U	S	E	U	E	R	S	S	R

Fig. 4 The Scytale cipher for $k = 4$

The word "Scytale" comes from the Greek word for "baton", which was one of the earliest tools used to encrypt and decrypt Scytale ciphers—a strip of leather or parchment was wrapped around a segmented baton, and the message enscribed row by row across the faces of the baton: when unwrapped, the message was not clear unless the recipient had a baton with the same number and size of faces.

Such a Scytale was referred to by Archilochus (700BC), and by Plutarch (in *Lives (Lysander 19)*) as an encryption device: *"... they make two round pieces of wood exactly alike ... Whenever, then, they wish to send some secret and important message, they make a scroll of parchment ... and* *wind it round their scytale ... they write what they wish on the parchment, just as it lies wrapped about the scytale; and when they have written their message, they take the parchment off and send it ... He, when he has received it, cannot otherwise get any meaning out of it ... unless he takes his own scytale and winds the strip of parchment about it...".*

A. The Scytale cipher in python

1. Variables

```
...
order = [] # this holds  the order that the message will be arranged into to make ciphertext
...
```

2. The encryption function

```
repetitions = int((len(m) / k)) + 1
for i in range(0,k):
    for j in range(0,repetitions):
        index = (i + (j * k))
        if index > len(m) - 1:
            continue
        else:
            order.append(index)
c = [m[i] for i in order]
```

3. The decryption function

```
m_no = (len(c) / k) + 1
repetitions = int((len(c) / k)) + 1
for i in range(0,repetitions):
    for j in range(0,k+1):
        index = (i + (j * repetitions))

        if index > len(m) - 1:

            continue

        else:

            re_order.append(index)

m_proof = [c[i] for i in re_order[0:len(m)]] # this works where k=x*len(m): x=int

m_proof = ''.join(m_proof)
```

B. Criticism of the Scytale cipher

There are two central criticisms of the Scytale cipher. The first is poetic and is derived from its historic literal application: the ciphertext ribbon displayed kinks in it corresponding to the Scytale that encoded the message, making it exceptionally easy to decode. It was thought that this was cured by enscribing the ciphertext onto another parchment and transmitting it that way. The second issue is the small size of the practical keyspace for this cipher—again, making it so easy to decipher without the key that it can be brute-forced manually.

The weakness of this cipher has led some historians to note that this may have actually been a method of guaranteeing authorship (like certification and signature systems subsequently developed)—unless the folds on the cyphertext ribbon matched exactly with the Scytale of the purported sender, the recipient could not be sure that the message was sent by that sender.

C. The Caesar cipher in Python

1. *Variables*

```
...

k = 0 # this can be any integer that falls within the message space

...
```

Unlike the Atbash Cipher, the Caesar cipher has a key. Here, it is limited to the alphabet used, e.g.:

- in the single case English alphabet $\longrightarrow k \in [0, 25]$; or,

- for the extended ASCII alphabet $\longrightarrow k \in [0, 255]$.

2. The encryption function

```
def enc(z):
  for x in z:
    c_ord.append(x + k)
enc(m_ord)
```

3. The decryption function

```
def dec(z):
  for x in z:
    m_ord_proof.append(x - k)
dec(c_ord)
```

D. Criticism of the Caesar cipher

Looking at the similarity of the encryption and decryption algorithms for the Atbash and Caesar ciphers, it is patently clear that both ciphers suffer the same weakness—the keyspace is too small. While the Caesar cipher grows the keyspace from 1 to either 26 or 256 (depending on the alphabet used), this is still within the realm of being decrypted by brute-force without a computer.

Further, with a simple statistical analysis a user would be able to determine which letters replace common letters (e.g., the letter that appears most commonly is likely to substitute for "e"). This is not a criticism unique to the Caesar cipher, it applies to any monoalphabetic substitution cipher (including non-linear substitutions), as well as many more sophisticated ciphers.

VI. The Vigênere cipher

The Vigènere cipher was developed at least a thousand years after the aforementioned ciphers. It is, however, comparable, as it is also a substitution cipher. Unlike the Atbash and Caesar ciphers, however, it is a polyaphabetic substitution cipher: in a polyalphabetic substitution cipher, the same input may return a different output (c.f. a monoalphabetic substitution cipher, where for each time a single letter is used in a message, the ciphertext will be the same).

It is essentially a cipher of addition: with the sum of the the numerical values of the message and the key making up the ciphertext:

message	m	a	k	e	y	o	u	r	m	e	s	s	a	g	e	s	e	c	u	r	e
key	s	w	e	e	t	s	w	e	e	t	s	w	e	e	t	s	w	e	e	t	s
ord(message)	77	65	75	69	89	79	85	82	77	69	83	83	65	71	69	83	69	67	85	82	69
ord(key)	83	87	69	69	84	83	87	69	69	84	83	87	69	69	84	83	87	69	69	84	83
ord(ciphertext)	160	152	144	138	173	162	172	151	146	153	166	170	134	140	153	166	156	136	154	166	152

Fig. 5 The Vigênere cipher for $k = “sweet''$

The Vigênere cipher was originally described by Giovan Battista Bellaso in his 1553 book *La cifra del. Sig. Giovan Battista Bellaso.* Bellaso worked as a secretary in the Italian courts during a golden period in the history of cryptography: *“The seven appended messages have been accurately compiled according to the concepts taught. They contain some beautiful things that are interesting to know. This will give the skilled and ingenious cryptographers the opportunity to strive to solve them, especially those who assert being capable to solve all kinds of ciphers. If this is true, as many believe, it will not be difficult for them to solve these cryptograms knowing all the rules by which they have been compiled, considering that the different ciphering methods are practically numberless.”* In the 19th Century, Bellaso's cipher was misattributed to the author Blaise de Vigènere.[14]

A. The Vigênere cipher in Python

1. Variables

```
...
k_ord_long = [] # this holds a transformation of k S.T. len(m) = len(k_ord_long)
...
```

2. The encryption function

```
def enc(z):
    for i,x in enumerate(m_ord):
        c_ord.append(m_ord[i] + k_ord_long[i])
enc(m_ord)
```

3. The decryption function

```
def dec(z):
    for i,x in enumerate(c_ord):
        m_ord_proof.append(c_ord[i] - k_ord_long[i])
dec(c_ord_proof)
```

B. Criticism of the Vigênere cipher

Where a Vigènere cipher has a small k, the cipher is susceptible to statistical distribution attacks. This becomes more difficult with the length of k. The strength of this cipher is not dependent on the strength of the encryption algorithm, but, rather the strength of the key—as such, it is not considered a "strong" cipher.

VII. One time pads

The extreme example of the above comes in the form of a one time pad cipher. This is a strict subset of the Vigènere cipher: it is a Vigènere cipher where $len(k) = len(m)$ and there is no discernible pattern in k (e.g. k is truly random).

If the length of the key is shorter than the length of the message, then the same weaknesses exist as in the Vigènere cipher. If there is some discernible pattern in the key then the key, and therefore the message, could be statistically derived (e.g., if the bible or some other book is used as the key, or the key is a badly generated random number with some pattern). If the same pad is used more than once, then inter-message analyses can reveal the key (which was how the original Vigènere cipher was purportedly cracked).

Where these requirements are satisfied and k is not compromised, the cipher is not compromisable. As such, these are still used today.

VIII. The Enigma cipher

Much like the Vigènere cipher (which continued to be used during the same period as the Enigma cipher), the Enigma cipher is a polyalphabetic substitution function. However, the encryption algorithm of the Enigma cipher is much more complex, with an enormous number of substitution alphabets cycled through (almost definitely enough that the same substitution alphabet was unlikely to be used in the same message).

This required inputting the key (the rotors, ring settings and plugboard connections chosen) as well as the "state" of the encryption algorithm (the position of the rotors on the machine). With each key input, roughly the following processes would occur:

1. if the plugboard connected the pressed letter with another, they would be swapped (e.g., if "q" and "b" were connected, $q \longrightarrow b$);

2. the button press would go through the various circuits embedded in each rotor (as selected, ordered and positioned), which alongside the inbuilt circuits would lead a light to shine through a letter that would be the ciphertext transformation; and

3. mechanical processes would, according to the ring settings, rotate each rotor by a certain number of positions, such that after each keypress the circuits between the input keys and output bulbs change (e.g., if "q" is typed, the same circuit to the output would not repeat until the period of the collection of rotors is completed).

To encode a simulacra of an Enigma machine in python is beyond the scope of this paper.[15] However, no matter how much more complex the Enigma cipher was in comparison to its predecessors, it was essentially still a polyalphabetic substitution cipher—albeit with a greatly expanded keyspace and a more complex encryption algorithm.

A. Criticism of the Enigma cipher

There were some historical weaknesses that rendered the Enigma cipher more breakable than it otherwise should have been. Many of these were procedural, including, for example, the losses of machines (that essentially revealed the encryption function despite the user's intention[16]), repetition of salutations, etc. There were some critical algorithmic flaws too, however:

1. for any letter input, the encryption algorithm would never return that letter as ciphertext;
2. for any letter input, given the same key and encryption algorithm "position", the ciphertext would always be identical.

The cryptographic and procedural flaws in the Enigma system meant that it was often broken within hours of a new key being chosen.

IX. The RSA cipher

The next revolution in cryptography came with the introduction of public private key encryption. This is distinguishable from the aforementioned ciphers in that there is no need for the transmitter of the information to securely pass to the receiver a secret key for the ciphertext to be secure. Rather, the receiver transmits to the transmitter K_p (the public key). This allows the transmitter to encode a message that only the receiver can decode by using k_s (the private, or "secret" key).

This requires the use of one-directional algorithms, which, in the case of RSA, was that when given three large integers, a, b, and c: $(m^a)^b \equiv a \pmod{c}$. Here, (a, c) is distributed as the public key, and (b) is the private key used in decryption.

A. The RSA cipher mathematically

1. *Key generation*

Key generation in RSA requires the multiplication of two large (100+ digit) prime numbers (p, q) in order to be secure. The public and private keys are functions of the multiples of these primes (e.g. $k_p = pq$ and $k_s = (p, q)$).

2. *The encryption function*

The transmitter breaks apart the message into a series of integers m which are then transformed with the following function: $c \equiv m^a \pmod{c}$.

3. *The decryption function*

The reciver recreates m from c using the following function: $c^b \equiv (m^a)^b \pmod{c}$.

B. Criticisms of the RSA cipher

While this is an incredibly difficult cipher to decipher, the ciphertext generated need not be deciphered for information to leak. If, for example, a plaintext “no” is, in ciphertext “1827301983091”, and that is the entirety of the response, its repetition may give a cryptologists sufficient information. The RSA cipher has since been modified to add certain randomness to the encryption algorithm such that two encryptions of the same text will have different results.

This algorithm is compliant with the *Kerchoff Priciple*, being secure despite *enc*() and *dec*() being publicly known. However, it is still critical to consider the use of this cryptographic system, human error, bad procedure and bad architecture have led to information being leaked despite the use of a strong cryptographic system.

X. Conclusion

Cryptographic ciphers have increased in complexity and strength over time. With their complexity, we have learned that:

- the larger the keyspace, the more resilient the cipher is to brute-force attacks;
- that polyalphabetic ciphers are more resilient to attack than monoalphabetic ciphers, since they are resilient to statistical frequency attacks;
- that public private key encryption allow the transmission of encrypted information without needing a private channel to send a key; and
- that without randomness in the encryption function, no matter how resilient the cipher is, it will be susceptible to information leakage.

A user cannot, however, allow the use of a resilient cryptographic cipher to lull the user into a false sense of security, as that is only one part of a strong cryptographic system. Good human practices, true randonmness in key generation and intelligent system architecture are all necessary for the cipher to help maintain secrecy. Finally, despite certain systems seeming secure, they must be subject to audit with the latest technology, as modern systems (such as nascent quantum computing systems) may compromise the security of these systems in the near future.

Footnotes and References

[1] Vatistas, G. H., Lin, S., and Kwok, C. K., "Reverse Flow Radius in Vortex Chambers," *AIAA Journal*, Vol. 24, No. 11, 1986, pp. 1872, 1873. doi: 10.2514/3.13046

[2] Reeds, Jim (1998). "Solved: The ciphers in book III of Trithemius's Steganographia". Cryptologia.

[3] U.S. Patent 4,405,829 for a "Cryptographic communications system and method", granted on September 20, 1983.

[4] Shannon, Claude (4 October 1949). "Communication Theory of Secrecy Systems". Bell System Technical Journal. 28: 662.

[5] Russel, Frank (1999). Information Gathering in Classical Greece. U. Michigan Press.

[6] Plutarch. Plutarch's Lives. with an English Translation by. Bernadotte Perrin. Cambridge, MA. Harvard University Press. London. William Heinemann Ltd. 1916.

[7] Paul Y. Hoskisson. "Jeremiah's Game". Insights.

[8] Suetonis. "Life of Julius Caesar".

[9] Giovan Battista Bellaso (1553). "La cifra del. Sig. Giovan Battista Bellaso".

[10] David, Kahn (1999). "The Codebreakers: The Story of Secret Writing".

[11] Green, Matthew. "The Ideal Cipher Model". https://blog.cryptographyengineering.com/2013/04/11/wonkery-mailbag-ideal-ciphers/

[12] Red Hat. "A Brief History of Cryptography". https://access.redhat.com/blogs/766093/posts/1976023

[13] Excuse any clumsiness in the coding, neither of the authors have any formal computer science training. For the sake of brevity, only snippets of the code are extracted, for the entire program, please see $https://github.com/oatsandsugar/crypto_play$.

[14] David, Kahn (1999). *The Codebreakers: The Story of Secret Writing*.

[15] If the readers wish to interact with such a model, we commend to the reader Brian Neal's "Py-Enigma", available at https://py-enigma.readthedocs.io/en/latest/.

[16] Today, we would see that as good cryptographic practice, but considering the relative simplicity of the Enigma cipher, the secrecy of that algorithm was a central mechanism for protecting the cipher's rigor.