Classical Cryptography

Encodings and digital ciphers

Karst Koymans

Informatics Institute
University of Amsterdam
(version 22.9, 2023/02/23 19:42:39 UTC)

Friday, February 24, 2023

Table of Contents

Codes and codebooks

- A **code** operates on **semantic** components
  - Words, paragraphs, ...
- A **codebook** is a **lookup table** for codes
- Codes can be very hard to cryptanalyse
- Possible analysis methods
  - Compare many different coded texts
  - Use side channels (other available sources)
  - Try to identify cribs
  - Build up knowledge over time

An example codebook

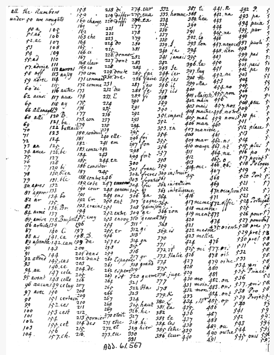

Figure 1: A code book

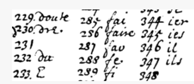

Figure 2: Enlarged part

Source: Slides Hans van der Meer

Encodings

- An **encoding** is a transformation of pieces of information into another representation for communication or storage
- An encoding is **keyless**
- An encoding can be **public** or **secret**
- The pieces of information need not have a semantic value like in a codebook and can be single letters or symbols

Ciphers and algorithms

- A **cipher** operates on meaningless components
  - Individual letters or small groups of letters
  - Bits or bytes
- Ciphers are **syntax** related
- Ciphers use **algorithms**
  - with secret (or public) keys as parameters
- Encryption is the process of applying a cipher
- Decryption is the process of reversing a cipher

Polygraphic versus polyliteral ciphers/encodings

- Polygraphic ciphers/encodings translate a block of letters into another block of letters, numbers or symbols
  - An example is Porta's digraph system
- Polyliteral ciphers/encodings translate a single letter into a (larger, full) block of letters, numbers or symbols
  - Polyliteral ciphers/encodings are in fact a simple substitution into another, often "bigger" but also "structured", alphabet which can henceforth be fractionated

Polybius Square

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	IJ	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

Figure 3: A simple polyliteral[1]encoding (**Polybius**)

[1]Because we use digits this is also called a dinome substitution

A rectangular variant

	0	1	2	3	4	5	6	7	8
0		A	B	C	D	E	F	G	H
1	I	J	K	L	M	N	O	P	Q
2	R	S	T	U	V	W	X	Y	Z

Figure 4: A 0-based rectangular encoding for the full alphabet

But note it still uses the legacy A=01 encoding

The standard legacy encoding

	0	1	2	3	4	5	6	7	8	9
0		A	B	C	D	E	F	G	H	I
1	J	K	L	M	N	O	P	Q	R	S
2	T	U	V	W	X	Y	Z			

Figure 5: A 0-based encoding for the full alphabet, with unused space

- This encoding is just the regular legacy encoding translating A, ..., Z to 01[2], ..., 26
- 00, 27, 28 and 29 are available for more symbols if needed

[2]One may or may not remove leading 0s, treating them as numbers or strings

The standard modern encoding

	0	1	2	3	4	5	6	7	8	9
0	A	B	C	D	E	F	G	H	I	J
1	K	L	M	N	O	P	Q	R	S	T
2	U	V	W	X	Y	Z				

Figure 6: A 0-based encoding for the full alphabet, with unused space

This encoding is just the regular modern encoding translating A, ..., Z to 00, ..., 25

A table for every base b numeral system (1)

Let us for instance look at base $b = 3$

	00	01	02	10	11	12	20	21	22
0	␣	A	B	C	D	E	F	G	H
1	I	J	K	L	M	N	O	P	Q
2	R	S	T	U	V	W	X	Y	Z

Figure 7: A ternary encoding for the full alphabet including a space

It would have been so nice[3] to build computers based on the (balanced) ternary system instead of the usual binary one...

[3]The Russians tried to do so: `https://en.wikipedia.org/wiki/Setun`

A table for every base b numeral system (2)

Let us now look at the common binary base $b = 2$

	000	001	010	011	100	101	110	111
00		A	B	C	D	E	F	G
01	H	I	J	K	L	M	N	O
10	P	Q	R	S	T	U	V	W
11	X	Y	Z					

Figure 8: A binary legacy encoding with room for $2^5 = 32$ symbols

Base32 is a modern variant with added symbols 2, 3, 4, 5, 6, 7

Bacon

Figure 9: Francis Bacon (1561 – 1626)

Source: https://en.wikipedia.org/wiki/Francis_Bacon

The Bacon code (steganography)

- Francis Bacon (1561–1626)
- First use a binary code with a=0 and b=1
  - In the original we had I=J and U=V, coding 24 letters with A=aaaaa, ..., Z=babbb
  - In modern variants the full alphabet is encoded[4] with A=aaaaa, ..., Z=bbaab
- SEconDlY HIDE The INDivIDuAL BitS by USiNg GLypH PROPeRTieS LiKE ColOR, ITaLIZatIon, SIze, ...

[4]Holden's book uses A=aaaab, ..., Z=bbaba

Baudot and Vernam

Figure 10: Émile Baudot (1845 – 1903)

Figure 11: Gilbert Vernam (1890 – 1960)

Source: https://en.wikipedia.org/wiki/Émile_Baudot

Source: https://en.wikipedia.org/wiki/Gilbert_Vernam

The Teletypewriter

- Émile Baudot (1845–1903)
  - Baudot code
  - Paper tape with punched holes
  - 5 positions or bits
- Gilbert Vernam (1890–1960)
  - Secures Baudot code transmission
  - Uses a second (key)tape to be XORed with the plaintext tape
  - Essentially creating a one-time pad

The wonderfully versatile XOR

- XOR is a binary (bitwise) operation
  - Its nice properties derive from addition modulo 2
  - Modulo 2 subtraction is the same as addition
  - Encryption works by $c = p \oplus k$
    - and since $k \oplus k = 0$
  - Decryption works by $p = c \oplus k$
- XOR also has a ternary, quaternary, ... variant
  - Multiple inputs and one output
  - Can be combined in arbitrary trees
    - And with some care even in graphs with loops

Length tricks

- Nulls
  - Using encoding symbols with no corresponding plaintext
- Straddling (“with a leg on each side”)
  - Use different length encoding strings for different plaintext letters
  - Usually the frequently occurring letters use a smaller length
  - This will result in compression properties
  - Enable fractionation
  - Also called a monome-binome or monome-dinome cipher

The straddling checkerboard (1)

	0	1	2	3	4	5	6	7	8	9
				A	B	C	D	E	F	G
1	H	I	J	K	L	M	N	O	P	Q
2	R	S	T	U	V	W	X	Y	Z	

Figure 12: Why are the first three positions blank?

The straddling checkerboard (2)

	0	1	8	3	4	5	2	9	7	6
				T	R	E	A	S	O	N
0	B	C	D	F	G	H	I	J	K	L
1	M	P	Q	U	V	W	X	Y	Z	.
8	0	1	2	3	4	5	6	7	8	9

Figure 13: A variant that compresses (most occurring letters monome)

Source: slides Hans van der Meer

The straddling checkerboard (3)

	0	1	8	3	4
	A	E	I	O	U
5	B	C	D	F	G
2	H	K	L	M	N
9	P	Q	R	S	T
7	V	W	X	Y	Z

Figure 14: A variant where the 6 can be used as a null

Source: slides Hans van der Meer

The straddling checkerboard (4)

	0	1	8	3	4	5
2	A	B	C	D	E	F
9	G	H	I	J	K	L
7	M	N	O	P	Q	R
62	S	T	U	V	W	X
67	Y	Z	0	1	2	3
69	4	5	6	7	8	9

Figure 15: A dinome-trinome variant

Source: slides Hans van der Meer

The straddling checkerboard (5)

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

Figure 16: Lots of homophones

Source: slides Hans van der Meer

Cryptanalysis of straddling checkerboards

- Identify dinome coordinates
  - They occur more frequently
  - They have lots of different contacts
  - Look at repetition of four or more identical digits
  - Look at patterns like $\boxed{\text{ABAB}}$
- Solve the resulting monoalphabetic substitution
- And possibly identify the key used

Fractionation after polyliteral encoding

- After having encoded letters one may consider subunits of polyliterals
  - In the binary case those subunits could be bits
- More substitutions and especially transpositions can be executed
  - That is what classic and modern block ciphers like DES and AES do
- The resulting new subunits might be assembled again into polyliterals
  - Which can then possibly be translated back to the original alphabet

Fractionating system example: ADFGVX (1)

Letter	Morse code
A	· –
D	– · ·
F	· · – ·
G	– – ·
V	· · · –
X	– · · –

Also see `https://www.johndcook.com/blog/2020/02/22/adfgvx/`

Fractionating system example: ADFGVX (2)

	A	D	F	G	V	X
A	b	5	x	q	j	c
D	6	y	r	k	d	7
F	z	s	l	e	8	1
G	t	m	f	9	2	u
V	n	g	0	3	v	o
X	h	a	4	w	p	i

Figure 17: ADFGVX 6-by-6 square

Fractionating system example: ADFGVX (3)

- First use the polyliteral ADFGVX square
- Then use a keyed columnar transposition
- Example encryption with keyword GANDHI and square filled as in previous slide
  - AGGAV AXGDA DFGGA FXFFV
    VXXFG XXVGF VAAXX ADAXG
    FFFFV D
- Exercise: decode this message

Shannon

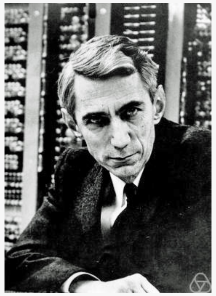

Figure 18: Claude Shannon (1916 – 2001)

Source: `https://en.wikipedia.org/wiki/Claude_Shannon`

Shannon's theory (1)

- **Confusion**
  - Each ciphertext bit has complex (nonlinear) relations with the plaintext and key bits
  - Mostly achieved by substitutions
- **Diffusion**
  - Each plaintext or key bit affects many bits of the ciphertext
  - Mostly achieved by transpositions

Shannon's theory (2)

- **Mixing** transformation (function) F
  - Non-secret, confusing and diffusing transformation
  - A transposition (T), followed by an alternation of linear Hill (H) maps and substitutions (S)
  - $F = H \circ S \circ H \circ S \circ H \circ T$
  - Both T and H operate on full blocks of letters
  - S operates componentwise, on each individual letter

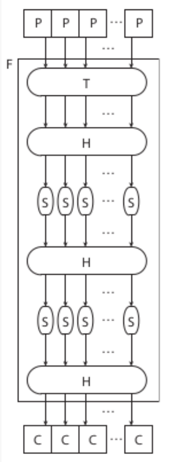

Figure 19: Shannon's mixing function F

Source: The Mathematics of Secrets by Joshua Holden

Shannon's theory (3)

- Shannon's cipher construction
  - Uses one, two or even more mixing transformations
    - For two mixings this is $C = W_{k_3} \circ F_2 \circ V_{k_2} \circ F_1 \circ U_{k_1}$
    - k_1, k_2, k_3 is keying material for simple ciphers U, V, W
    - Here secret keys enter the scene by adding more confusion, typically through the simple substitutions U, V and W

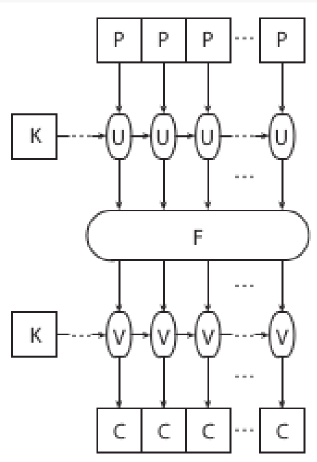

Figure 20: Shannon's cipher

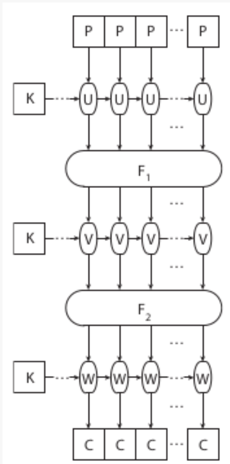

Figure 21: Shannon's more secure cipher

Source: The Mathematics of Secrets by Joshua Holden

SP-networks

- **SP-networks** resemble Shannon's construction
  - Works with bits instead of larger alphabets
- Uses **large diffusing transpositions** of bits
- Uses **smaller confusing polygraphic substitutions**
  - Works on sequences of bits (bytes, nibbles, ...)
- Alternates these in a number of rounds
- Mixes in (parts of) the key at the start of each round
  - Mixing uses simple XORs
  - Also at the end the key is once more mixed in

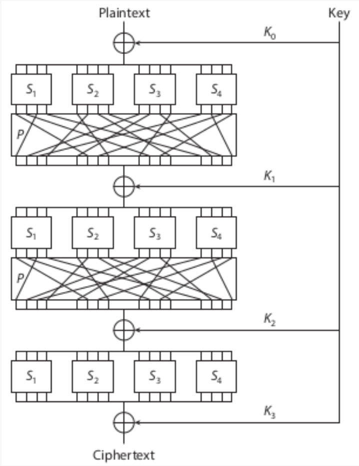

Figure 22: SP network

Source: The Mathematics of Secrets by Joshua Holden

Feistel

Figure 23: Horst Feistel (1915 – 1990)

Source: `https://www.ithistory.org/honor-roll/mr-horst-feistel`

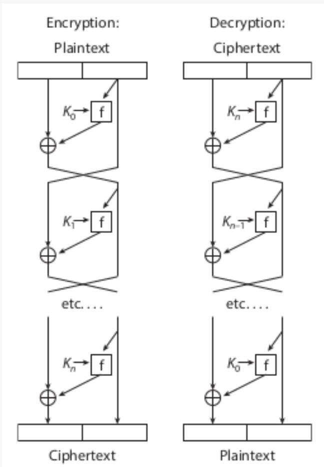

Figure 24: Feistel cipher construction

Source: The Mathematics of Secrets by Joshua Holden

Feistel networks (building block)

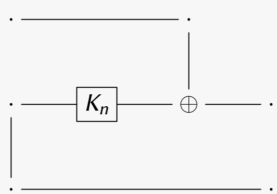

Figure 25: Building block (also used upside down)

Feistel networks (first few steps)

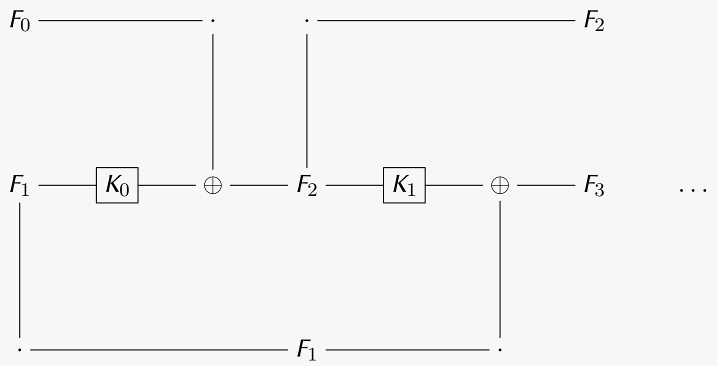

Figure 26: $F_2 = \mathcal{F}(K_0, F_1) \oplus F_0$; $F_3 = \mathcal{F}(K_1, F_2) \oplus F_1$

Feistel network encryption sequence

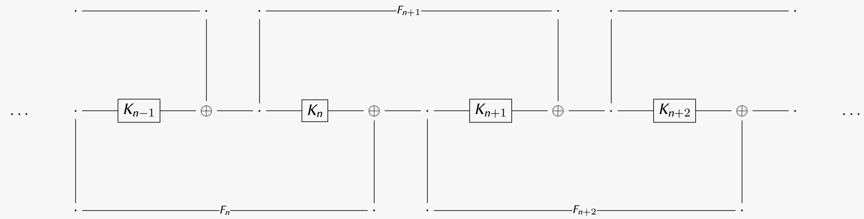

Figure 27: $F_{n+2} = \mathcal{F}(K_n, F_{n+1}) \oplus F_n$

Simpler Feistel network building block

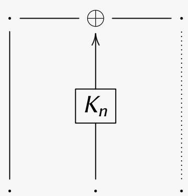

Figure 28: Building block (also used upside down)

Simpler Feistel network first steps

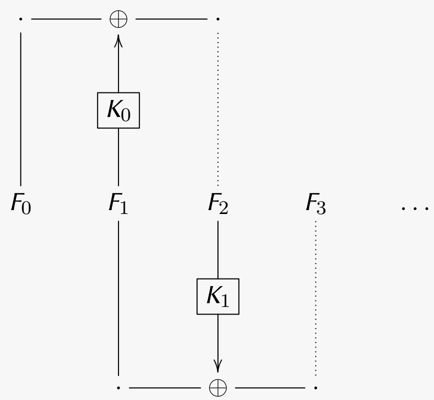

Figure 29: $F_2 = \mathcal{F}(K_0, F_1) \oplus F_0$; $F_3 = \mathcal{F}(K_1, F_2) \oplus F_1$

Simpler Feistel network encryption sequence

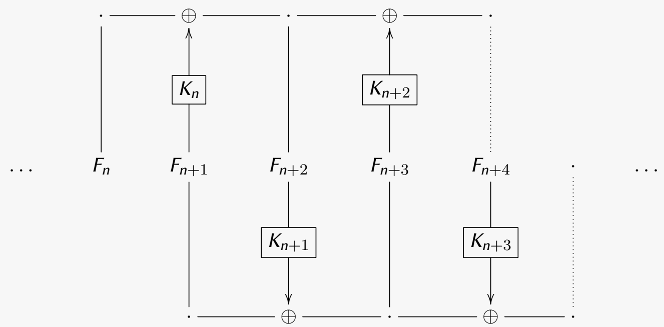

Figure 30: $F_{n+2} = \mathcal{F}(K_n, F_{n+1}) \oplus F_n$

Hence Feistel is “Fibonacci”-like