Lecture 13: Vigenère, Vernam, and DES

Example of Vigenère Cipher

Encode the plaintext: `thequickbrownfoxjumpsoverthelazydog` using the key TIGER.

Example of Vigenère Cipher

Encode the plaintext: `thequickbrownfoxjumpsoverthelazydog` using the key TIGER.

plaintext	`the`	`quick`	`brown`	`fox`	`jumps`	`over`	`the`	`lazy`	`dog`

Example of Vigenère Cipher

Encode the plaintext: `thequickbrownfoxjumpsoverthelazydog` using the key TIGER.

plaintext	`the`	`quick`	`brown`	`fox`	`jumps`	`over`	`the`	`lazy`	`dog`
+ key	`TIG`	`ERTIG`	`ERTIG`	`ERT`	`IGERT`	`IGER`	`TIG`	`ERTI`	`GER`

Example of Vigenère Cipher

Encode the plaintext: `thequickbrownfoxjumpsoverthelazydog` using the key TIGER.

plaintext	the	quick	brown	fox	jumps	over	the	lazy	dog
+ key	TIG	ERTIG	ERTIG	ERT	IGERT	IGER	TIG	ERTI	GER
ciphertext	MPK	ULBKQ	FIHET	JFQ	RAQGL	WBII	MPK	PRSG	JSX

Attacking the Vigenère Cipher

- Kasiski: Look at distances between identical pieces of ciphertext.

Attacking the Vigenère Cipher

- Kasiski: Look at distances between identical pieces of ciphertext.

- Idea: the key length probably divides this distance.

Example 10.2.3

A Vigenère cipher has been used on English source text which has removed punctuation and blanks. The following ciphertext was produced:

offset	ciphertext									
0	UPVZB	BVUPN	KKFOL	OGAKU	FBTKF	LFXUJ	VIPZV	KFZXO	FIDLO	ONLUP
50	KKFUZ	OMQFQ	MQXKU	AFIUP	VVVVK	KFDFL	DMFIU	PVVFI	ZVTMU	XDBZY
100	FVVYF	ZTHBA	ZQHEY	LTXVU	JVXFM	IDRSQ	EJNCI	PVZZQ	HQEYJ	BZQHB
150	YHTWL	OUWND	OLVUJ	VREZA	JHTWW	VPTZW	VLVDM	TROPV	XWIMN	KJBVE
200	FITKV	XRQEL	FZOBY	HSMND	TVFOJ	DZQHB	YLOOZ	QTQXK	UISLS	LNLUP
250	RESWB	HOEZQ	HERVC	MRWJV	XWIMR	LSISR	WMIHF	TZQHN	CXUBV	UJVXF
300	JZTOJ	VXGJA	REMMU	GPEEG	PEEWP	BYHXI	KHS			

Example 10.2.3

offset	ciphertext									
0	UPVZB	BVUPN	KKFOL	OGAKU	FBTKF	LFXUJ	VIPZV	KFZXO	FIDLO	ONLUP
50	KKFUZ	OMQFQ	MQXKU	AFIUP	VVVVK	KFDFL	DMFIU	PVVFI	ZVTMU	XDBZY
100	FVVYF	ZTHBA	ZQHEY	LTXVU	JVXFM	IDRSQ	EJNCI	PVZZQ	HQEYJ	BZQHB
150	YHTWL	OUWND	OLVUJ	VREZA	JHTWW	VPTZW	VLVDM	TROPV	XWIMN	KJBVE
200	FITKV	XRQEL	FZOBY	HSMND	TVFOJ	DZQHB	YLOOZ	QTQXK	UISLS	LNLUP
250	RESWB	HOEZQ	HERVC	MRWJV	XWIMR	LSISR	WMIHF	TZQHN	CXUBV	UJVXF
300	JZTOJ	VXGJA	REMMU	GPEEG	PEEWP	BYHXI	KHS			

ℓ	submessage offsets	frequencies of letters in submessage sorted																								
2		F	V	U	Z	J	T	X	R	S	P	H	N	E	K	M	O	Y	D	I	B	C	L	W	G	A	Q
	0, 2, 4,...	19	16	12	12	11	10	9	8	8	7	6	6	5	5	5	5	5	4	4	3	2	2	2	1	0	0
		Q	V	L	B	I	O	E	K	W	H	M	Z	P	U	A	D	X	G	T	Y	F	J	N	R	C	S
	1, 3, 5, ...	14	14	13	10	10	10	9	9	9	8	8	7	6	6	5	5	5	3	3	3	2	2	2	2	1	0
4	0, 4, 8, ...	12	10	10	7	6	6	6	5	5	4	2	2	2	1	1	1	1	1	1	1	0	0	0	0	0	0
	1, 5, 9, ...	10	9	8	7	5	5	5	5	5	5	4	3	3	2	2	2	1	1	1	0	0	0	0	0	0	0
	2, 6, 10, ...	12	11	8	8	7	5	5	5	4	4	2	2	2	2	2	1	1	1	1	0	0	0	0	0	0	0
	3, 7, 11, ...	9	9	8	8	7	5	5	5	4	3	3	3	3	3	2	2	2	1	1	0	0	0	0	0	0	0

Example 10.2.3

offset	ciphertext									
0	UPVZB	BVUPN	KKFOL	OGAKU	FBTKF	LFXUJ	VIPZV	KFZXO	FIDLO	ONLUP
50	KKFUZ	OMQFQ	MQXKU	AFIUP	VVVVK	KFDFL	DMFIU	PVVFI	ZVTMU	XDBZY
100	FVVYF	ZTHBA	ZQHEY	LTXVU	JVXFM	IDRSQ	EJNCI	PVZZQ	HQEYJ	BZQHB
150	YHTWL	OUWND	OLVUJ	VREZA	JHTWW	VPTZW	VLVDM	TROPV	XWIMN	KJBVE
200	FITKV	XRQEL	FZOBY	HSMND	TVFOJ	DZQHB	YLOOZ	QTQXK	UISLS	LNLUP
250	RESWB	HOEZQ	HERVC	MRWJV	XWIMR	LSISR	WMIHF	TZQHN	CXUBV	UJVXF
300	JZTOJ	VXGJA	REMMU	GPEEG	PEEWP	BYHXI	KHS			

ℓ	submessage offsets	frequencies of letters in submessage sorted																								
2	0, 2, 4, ...	19	16	12	12	11	10	9	8	8	7	6	6	5	5	5	5	5	4	4	3	2	2	2	1	0	0
	1, 3, 5, ...	14	14	13	10	10	10	9	9	9	8	8	7	6	6	5	5	5	3	3	3	2	2	2	2	1	0
4		F	J	U	P	H	S	T	M	O	V	B	D	I	C	E	G	N	W	X	Z	A	K	L	Q	R	Y
	0, 4, 8, ...	12	10	10	7	6	6	6	5	5	4	2	2	2	1	1	1	1	1	1	1	0	0	0	0	0	0
		B	V	M	I	A	E	L	P	Q	Z	W	0	T	D	N	X	C	J	U	F	G	H	K	R	S	Y
	1, 5, 9, ...	10	9	8	7	5	5	5	5	5	5	4	3	3	2	2	2	1	1	1	0	0	0	0	0	0	0
		V	Z	R	X	F	K	N	Y	E	T	D	I	L	S	U	B	C	J	W	A	G	H	M	O	P	Q
	2, 6, 10, ...	12	11	8	8	7	5	5	5	4	4	2	2	2	2	2	1	1	1	1	0	0	0	0	0	0	0
		K	Q	H	L	O	U	V	W	E	D	G	I	X	Y	F	R	Z	J	P	A	B	C	M	N	S	T
	3, 7, 11, ...	9	9	8	8	7	5	5	5	4	3	3	3	3	3	2	2	2	1	1	0	0	0	0	0	0	0

Example 10.2.3

ℓ	submessage offsets	frequencies of letters in submessage sorted			
4		F	J	U	P
	0, 4, 8, ...	12	10	10	7
		B	V	M	I
	1, 5, 9, ...	10	9	8	7
		V	Z	R	X
	2, 6, 10, ...	12	11	8	8
		K	Q	H	L
	3, 7, 11, ...	9	9	8	8

common ciphertext letters for $\ell = 4$

FJUP
BVMI
VZRX
KQHL

$(c - \text{`}e\text{'}) \mod 26$ →

corresponding key letters

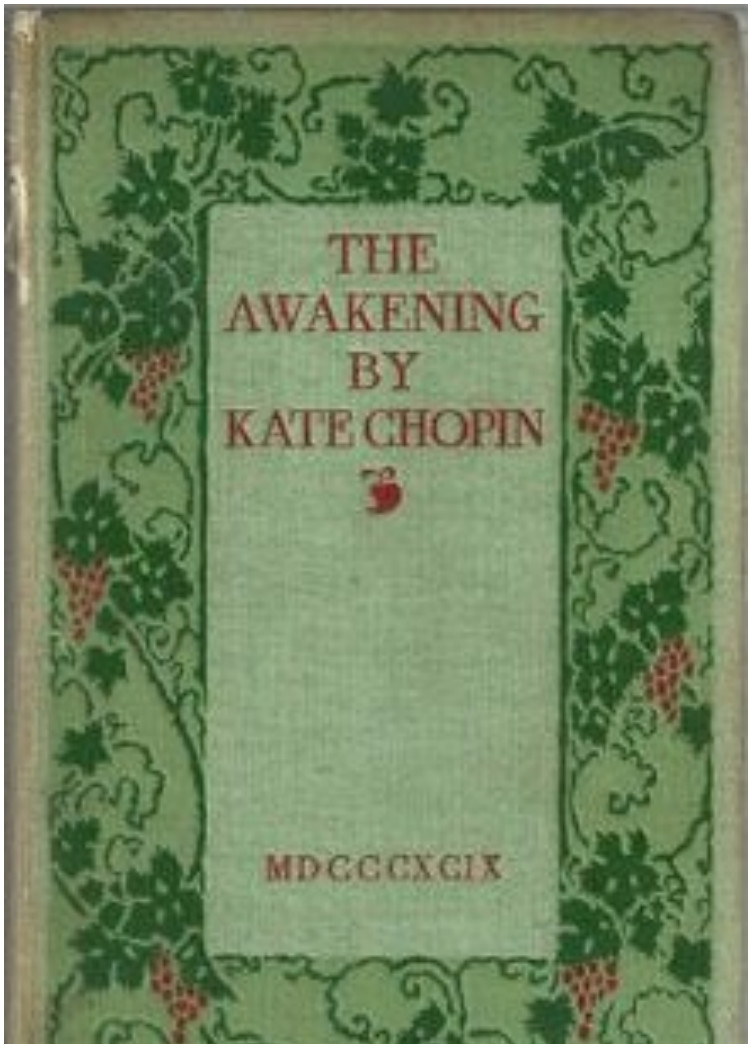
BFQL
XRIE
RVNT
GMDH


44=256 attempts later...

Example 10.2.3

ciphertext	UPVZB	BVUPN	KKFOL	OGAKU	FBTKF	LFXUJ	VIPZV	KFZXO	FIDLO	ONLUP
+ key	BIRDB	IRDBI	RDBIR	DBIRD	BIRDB	IRDBI	RDBIR	DBIRD	BIRDB	IRDBI
plaintext	thewa	terof	thegu	lfstr	etche	doutb	efore	hergl	eamin	gwith

Source:



Block Ciphers

A block cipher:

Block Ciphers

A block cipher:

- breaks plaintext into blocks length n

Block Ciphers

A block cipher:

- breaks plaintext into blocks length n
- applies a function to each of these smaller blocks to produce ciphertext

Block Ciphers

A block cipher:

- breaks plaintext into blocks length n
- applies a function to each of these smaller blocks to produce ciphertext
- Example 1: Vigenère cipher

Block Ciphers

A block cipher:

- breaks plaintext into blocks length n
- applies a function to each of these smaller blocks to produce ciphertext
- Example 1: Vigenère cipher
- Example 2: Data Encryption Standard (DES)

Block Ciphers vs. Stream Ciphers

Substitution and Vigenère ciphers are *block ciphers*:

Block Ciphers vs. Stream Ciphers

Substitution and Vigenère ciphers are *block ciphers*:

- apply a fixed mapping to blocks consisting of a fixed number of characters.

Block Ciphers vs. Stream Ciphers

Substitution and Vigenère ciphers are *block ciphers*:

- apply a fixed mapping to blocks consisting of a fixed number of characters.
- *memoryless*: Mapping of block doesn't depend on position in the message.

Block Ciphers vs. Stream Ciphers

Substitution and Vigenère ciphers are *block ciphers*:

- apply a fixed mapping to blocks consisting of a fixed number of characters.
- *memoryless*: Mapping of block doesn't depend on position in the message.

Stream ciphers:

Block Ciphers vs. Stream Ciphers

Substitution and Vigenère ciphers are *block ciphers*:

- apply a fixed mapping to blocks consisting of a fixed number of characters.
- *memoryless*: Mapping of block doesn't depend on position in the message.

Stream ciphers:

- allow the mapping to depend on the position within the message

Block Ciphers vs. Stream Ciphers

Substitution and Vigenère ciphers are *block ciphers*:

- apply a fixed mapping to blocks consisting of a fixed number of characters.
- *memoryless*: Mapping of block doesn't depend on position in the message.

Stream ciphers:

- allow the mapping to depend on the position within the message
- also called *state ciphers*

The English Alphabet as binary numbers

- Recall: $A \leftrightarrow 0$, $B \leftrightarrow 1$, $\ldots$, $Z \leftrightarrow 25$

The English Alphabet as binary numbers

- Recall: $A \leftrightarrow 0$, $B \leftrightarrow 1$, $\ldots$, $Z \leftrightarrow 25$
- Common to use binary: $\mathcal{A} = \{0, 1\}$. Write letters as binary words with 5 digits:

The English Alphabet as binary numbers

- Recall: $A \leftrightarrow 0$, $B \leftrightarrow 1$, $\ldots$, $Z \leftrightarrow 25$
- Common to use binary: $\mathcal{A} = \{0, 1\}$. Write letters as binary words with 5 digits:
  - $A \leftrightarrow 0 \leftrightarrow 00000$

The English Alphabet as binary numbers

- Recall: $A \leftrightarrow 0$, $B \leftrightarrow 1$, $\ldots$, $Z \leftrightarrow 25$
- Common to use binary: $\mathcal{A} = \{0, 1\}$. Write letters as binary words with 5 digits:
  - $A \leftrightarrow 0 \leftrightarrow 00000$
  - $B \leftrightarrow 1 \leftrightarrow 00001$

The English Alphabet as binary numbers

- Recall: $A \leftrightarrow 0$, $B \leftrightarrow 1$, $\ldots$, $Z \leftrightarrow 25$
- Common to use binary: $\mathcal{A} = \{0, 1\}$. Write letters as binary words with 5 digits:
  - $A \leftrightarrow 0 \leftrightarrow 00000$
  - $B \leftrightarrow 1 \leftrightarrow 00001$
  - $C \leftrightarrow 2 \leftrightarrow 00010$

The English Alphabet as binary numbers

- Recall: $A \leftrightarrow 0$, $B \leftrightarrow 1$, $\ldots$, $Z \leftrightarrow 25$
- Common to use binary: $\mathcal{A} = \{0, 1\}$. Write letters as binary words with 5 digits:
  - $A \leftrightarrow 0 \leftrightarrow 00000$
  - $B \leftrightarrow 1 \leftrightarrow 00001$
  - $C \leftrightarrow 2 \leftrightarrow 00010$

  etc...

The English Alphabet as binary numbers

- Recall: $A \leftrightarrow 0$, $B \leftrightarrow 1$, $\ldots$, $Z \leftrightarrow 25$
- Common to use binary: $\mathcal{A} = \{0, 1\}$. Write letters as binary words with 5 digits:
  - $A \leftrightarrow 0 \leftrightarrow 00000$
  - $B \leftrightarrow 1 \leftrightarrow 00001$
  - $C \leftrightarrow 2 \leftrightarrow 00010$

    etc...
  - $Y \leftrightarrow 24 \leftrightarrow 11000$

The English Alphabet as binary numbers

- Recall: $A \leftrightarrow 0$, $B \leftrightarrow 1$, $\ldots$, $Z \leftrightarrow 25$
- Common to use binary: $\mathcal{A} = \{0, 1\}$. Write letters as binary words with 5 digits:
  - $A \leftrightarrow 0 \leftrightarrow 00000$
  - $B \leftrightarrow 1 \leftrightarrow 00001$
  - $C \leftrightarrow 2 \leftrightarrow 00010$

    etc...
  - $Y \leftrightarrow 24 \leftrightarrow 11000$
  - $Z \leftrightarrow 25 \leftrightarrow 11001$

The English Alphabet as binary numbers

- Recall: $A \leftrightarrow 0$, $B \leftrightarrow 1$, $\ldots$, $Z \leftrightarrow 25$
- Common to use binary: $\mathcal{A} = \{0, 1\}$. Write letters as binary words with 5 digits:
  - $A \leftrightarrow 0 \leftrightarrow 00000$
  - $B \leftrightarrow 1 \leftrightarrow 00001$
  - $C \leftrightarrow 2 \leftrightarrow 00010$

    etc...
  - $Y \leftrightarrow 24 \leftrightarrow 11000$
  - $Z \leftrightarrow 25 \leftrightarrow 11001$

If we use the alphabet $\mathcal{A} = \{0, 1\}$ like this, then the encryption and decryption functions operate on blocks of length 5.

Addition of binary words

- This is just addition in K^n like we've been doing.

Addition of binary words

- This is just addition in K^n like we've been doing.

- Also denoted by $\oplus$

Addition of binary words

- This is just addition in K^n like we've been doing.

- Also denoted by $\oplus$
- Example: 1011+0100=1011$\oplus$0100 = 1111.

Vernam Cipher

- Stream cipher on $\{0, 1\}$.

Vernam Cipher

- Stream cipher on $\{0, 1\}$.
- Keys are also messages over $\{0, 1\}$.

Vernam Cipher

- Stream cipher on $\{0, 1\}$.
- Keys are also messages over $\{0, 1\}$.
- Encryption: $E_k(m) = c = m \oplus k$.

Vernam Cipher

- Stream cipher on $\{0, 1\}$.
- Keys are also messages over $\{0, 1\}$.
- Encryption: $E_k(m) = c = m \oplus k$.
- As $x + x = 0$ in K^n, decryption is just: $D_k(c) = m = c \oplus k$.

Vernam Cipher

- Stream cipher on $\{0, 1\}$.
- Keys are also messages over $\{0, 1\}$.
- Encryption: $E_k(m) = c = m \oplus k$.
- As $x + x = 0$ in K^n, decryption is just: $D_k(c) = m = c \oplus k$.
- If each digit of k is chosen randomly with probability $1/2$, Vernam cipher is known as the *one-time pad*.

Vernam Cipher

- Stream cipher on $\{0, 1\}$.
- Keys are also messages over $\{0, 1\}$.
- Encryption: $E_k(m) = c = m \oplus k$.
- As $x + x = 0$ in K^n, decryption is just: $D_k(c) = m = c \oplus k$.
- If each digit of k is chosen randomly with probability $1/2$, Vernam cipher is known as the *one-time pad*.
- Completely secure, but also completely impractical (how would you send secure messages to all your contacts?).

Feistel Networks

Goal:

Feistel Networks

Goal:

1. have keyspace large enough so that exhaustive search is infeasible (trying to beat frequency analysis, say).

Feistel Networks

Goal:

1. have keyspace large enough so that exhaustive search is infeasible (trying to beat frequency analysis, say).
2. have blocksize large enough to make it difficult to build a database from observed ciphertext

Feistel Networks

Goal:

1. have keyspace large enough so that exhaustive search is infeasible (trying to beat frequency analysis, say).
2. have blocksize large enough to make it difficult to build a database from observed ciphertext
3. get a function that is guaranteed to be invertible and to be its own inverse.

Feistel Networks

Goal:

1. have keyspace large enough so that exhaustive search is infeasible (trying to beat frequency analysis, say).
2. have blocksize large enough to make it difficult to build a database from observed ciphertext
3. get a function that is guaranteed to be invertible and to be its own inverse.

Confusion: complicate the relationship between key and ciphertext.

Feistel Networks

Goal:

1. have keyspace large enough so that exhaustive search is infeasible (trying to beat frequency analysis, say).
2. have blocksize large enough to make it difficult to build a database from observed ciphertext
3. get a function that is guaranteed to be invertible and to be its own inverse.

Confusion: complicate the relationship between key and ciphertext.

Diffusion: spread the structure present in the plaintext so that a ciphertext depends on a lot of plaintext bits.

Feistel Networks

Maps messages with $2n$ bits to ciphertext of the same length where $n \in \mathbb{N}$.

Feistel Networks

Maps messages with $2n$ bits to ciphertext of the same length where $n \in \mathbb{N}$. The input to the scheme is the message itself and a key k.

Feistel Networks

Maps messages with $2n$ bits to ciphertext of the same length where $n \in \mathbb{N}$. The input to the scheme is the message itself and a key k. Encryption proceeds through r rounds.

Feistel Networks

Maps messages with $2n$ bits to ciphertext of the same length where $n \in \mathbb{N}$. The input to the scheme is the message itself and a key k. Encryption proceeds through r rounds.

1. A *key-scheduling algorithm* determines the subkeys $k_1, \ldots, k_r$. Each subkey determines a function $f_{k_i} : \{0,1\}^n \to \{0,1\}^n$.

Feistel Networks

Maps messages with $2n$ bits to ciphertext of the same length where $n \in \mathbb{N}$. The input to the scheme is the message itself and a key k. Encryption proceeds through r rounds.

1. A *key-scheduling algorithm* determines the subkeys $k_1, \ldots, k_r$. Each subkey determines a function $f_{k_i} : \{0,1\}^n \to \{0,1\}^n$.
2. A $2n$-bit message m is split into an n-bit left and and n-bit right half, written $m = (m_0, m_1)$ may write as (L, R).

Feistel Networks

Maps messages with $2n$ bits to ciphertext of the same length where $n \in \mathbb{N}$. The input to the scheme is the message itself and a key k. Encryption proceeds through r rounds.

1. A *key-scheduling algorithm* determines the subkeys $k_1, \ldots, k_r$. Each subkey determines a function $f_{k_i} : \{0,1\}^n \to \{0,1\}^n$.
2. A $2n$-bit message m is split into an n-bit left and and n-bit right half, written $m = (m_0, m_1)$ may write as (L, R).Rounds:

$$\begin{aligned}
1: &\quad (m_0, m_1) \mapsto (m_1, m_2 = m_0 \oplus f_{k_1}(m_1)) \\
2: &\quad (m_1, m_2) \mapsto (m_2, m_3 = m_1 \oplus f_{k_2}(m_2)) \\
&\vdots \qquad\qquad\qquad\qquad \vdots \\
r: &\quad (m_{r-1}, m_r) \mapsto (m_r, m_{r+1} = m_{r-1} \oplus f_{k_r}(m_r))
\end{aligned}$$

Feistel Networks

Maps messages with $2n$ bits to ciphertext of the same length where $n \in \mathbb{N}$. The input to the scheme is the message itself and a key k. Encryption proceeds through r rounds.

1. A *key-scheduling algorithm* determines the subkeys $k_1, \ldots, k_r$. Each subkey determines a function $f_{k_i} : \{0,1\}^n \to \{0,1\}^n$.
2. A $2n$-bit message m is split into an n-bit left and and n-bit right half, written $m = (m_0, m_1)$ may write as (L, R).Rounds:

$$\begin{aligned} 1: &\quad (m_0, m_1) \mapsto (m_1, m_2 = m_0 \oplus f_{k_1}(m_1)) \\ 2: &\quad (m_1, m_2) \mapsto (m_2, m_3 = m_1 \oplus f_{k_2}(m_2)) \\ &\vdots \qquad\qquad\qquad\qquad \vdots \\ r: &\quad (m_{r-1}, m_r) \mapsto (m_r, m_{r+1} = m_{r-1} \oplus f_{k_r}(m_r)) \end{aligned}$$

The output interchanges the left and right halves

Feistel Networks

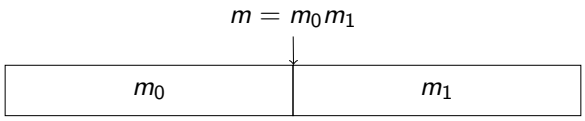
Maps messages with $2n$ bits to ciphertext of the same length where $n \in \mathbb{N}$. The input to the scheme is the message itself and a key k. Encryption proceeds through r rounds.

1. A *key-scheduling algorithm* determines the subkeys $k_1, \ldots, k_r$. Each subkey determines a function $f_{k_i} : \{0,1\}^n \to \{0,1\}^n$.
2. A $2n$-bit message m is split into an n-bit left and and n-bit right half, written $m = (m_0, m_1)$ may write as (L, R).Rounds:

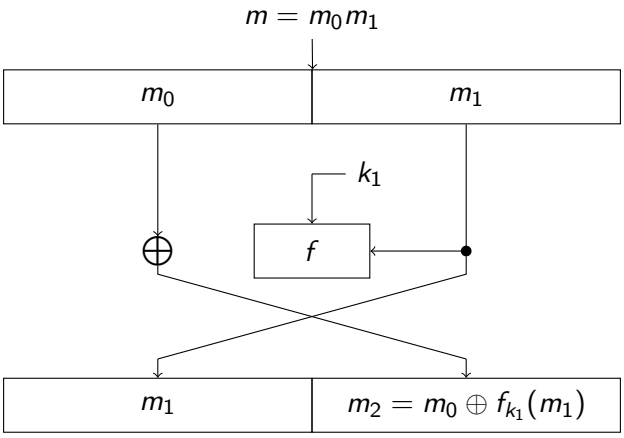
$$\begin{aligned} 1: &\quad (m_0, m_1) \mapsto (m_1, m_2 = m_0 \oplus f_{k_1}(m_1)) \\ 2: &\quad (m_1, m_2) \mapsto (m_2, m_3 = m_1 \oplus f_{k_2}(m_2)) \\ &\vdots \qquad\qquad\qquad\qquad \vdots \\ r: &\quad (m_{r-1}, m_r) \mapsto (m_r, m_{r+1} = m_{r-1} \oplus f_{k_r}(m_r)) \end{aligned}$$

   The output interchanges the left and right halves
3. The interchange allows decryption to use the same form of the r rounds but with the subkeys in reverse order.

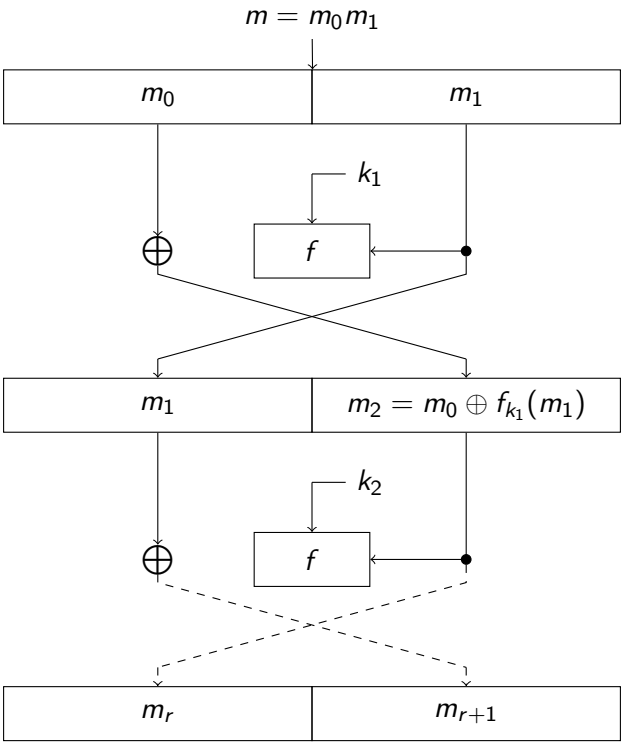
Feistel Network Diagram



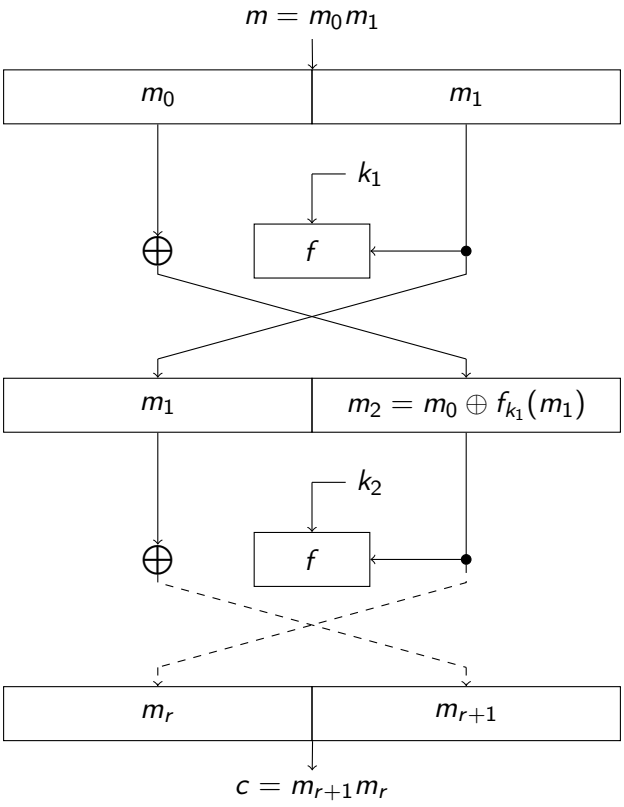
Feistel Network Diagram



Feistel Network Diagram



Feistel Network Diagram



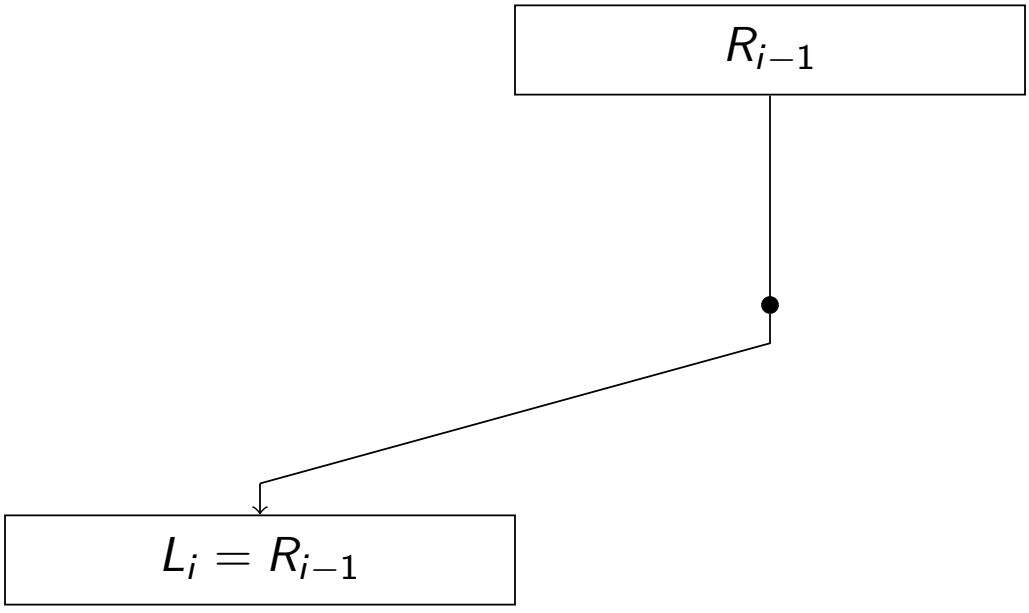
Feistel Networks

Building blocks: for round i, given input $x = (L_{i-1}, R_{i-1})$ and round key k:

Feistel Networks

Building blocks: for round i, given input $x = (L_{i-1}, R_{i-1})$ and round key k:

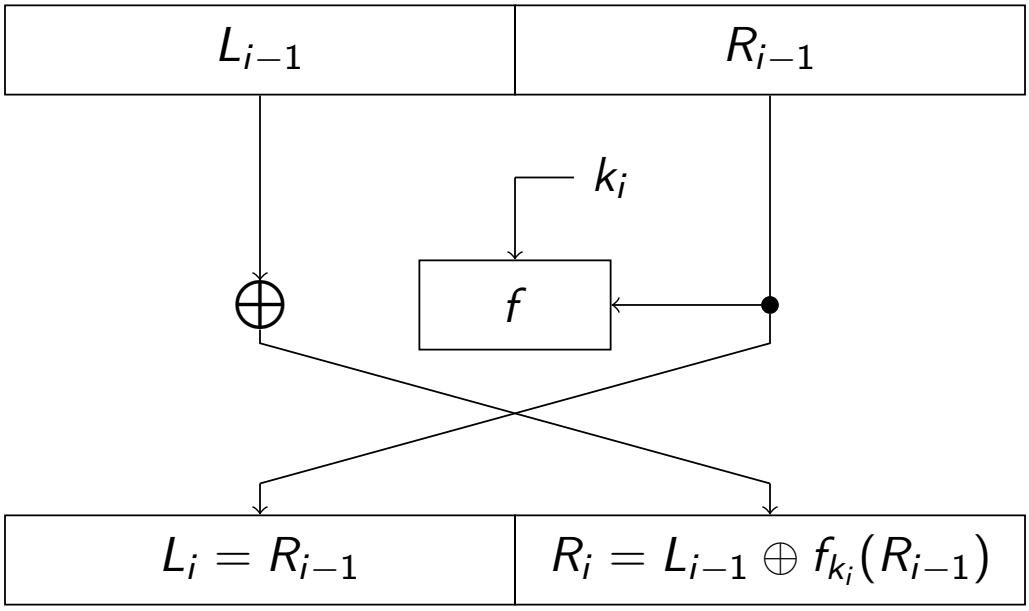
1. $L_i = R_{i-1}$



Feistel Networks

Building blocks: for round i, given input $x = (L_{i-1}, R_{i-1})$ and round key k:

1. $L_i = R_{i-1}$
2. $R_i = L_{i-1} \oplus f_i(R_{i-1})$



Feistel Networks (Simplified from book)

The basic algorithm for a Feistel Network with j rounds, cipher input $x = (L_0, R_0)$ and key k is:

1. Use k to calculate the **key schedule**: $k_1, k_2, \ldots k_j$. (In other words, calculate the keys that will be used in each round.)
2. For i $= 1$ to j:
   a. $L_i = R_{i-1}$
   b. $R_i = L_{i-1} \oplus f_i(R_{i-1})$
3. Return $x := (L_j, R_j)$

Feistel Networks (Simplified from book)

The basic algorithm for a Feistel Network with j rounds, cipher input $x = (L_0, R_0)$ and key k is:

1. Use k to calculate the **key schedule**: $k_1, k_2, \ldots k_j$. (In other words, calculate the keys that will be used in each round.)
2. For i = 1 to j:
   a. $L_i = R_{i-1}$
   b. $R_i = L_{i-1} \oplus f_i(R_{i-1})$
3. Return $x := (L_j, R_j)$

Since $R_{i-1} = L_i$ and $L_{i-1} = R_i \oplus f_i(R_{i-1})$, the Feistel network is always invertible.

DES

- DES was the standard for block ciphers for a few decades.

DES

- DES was the standard for block ciphers for a few decades.
- Developed in the 1970's by IBM and the NSA.

DES

- DES was the standard for block ciphers for a few decades.
- Developed in the 1970's by IBM and the NSA.
- Accepted as a Federal Information Processing Standard (FIPS) for the United States in 1977.

DES

- DES was the standard for block ciphers for a few decades.
- Developed in the 1970's by IBM and the NSA.
- Accepted as a Federal Information Processing Standard (FIPS) for the United States in 1977.
- Information can be found online here.

DES

- DES was the standard for block ciphers for a few decades.
- Developed in the 1970's by IBM and the NSA.
- Accepted as a Federal Information Processing Standard (FIPS) for the United States in 1977.
- Information can be found online here.
- FIPS approval for DES was withdrawn in 2005 primarily because of short key length.

DES

- DES was the standard for block ciphers for a few decades.
- Developed in the 1970's by IBM and the NSA.
- Accepted as a Federal Information Processing Standard (FIPS) for the United States in 1977.
- Information can be found online here.
- FIPS approval for DES was withdrawn in 2005 primarily because of short key length.
- Principles employed by DES are still used in the current encryption standards (e.g. AES and Triple-DES)

DES (cont.)

- The DES algorithm is public. The security is in the secret key.

DES (cont.)

- The DES algorithm is public. The security is in the secret key.
- DES uses a 56-bit key. (The key is actually 64 bits long, but 8 of these are parity-check bits)

DES (cont.)

- The DES algorithm is public. The security is in the secret key.
- DES uses a 56-bit key. (The key is actually 64 bits long, but 8 of these are parity-check bits)
- DES is a 16-round Feistel network.

DES (cont.)

- The DES algorithm is public. The security is in the secret key.
- DES uses a 56-bit key. (The key is actually 64 bits long, but 8 of these are parity-check bits)
- DES is a 16-round Feistel network.
- The round functions of the DES Feistel network are created using graphic in Figure 10.4 b.

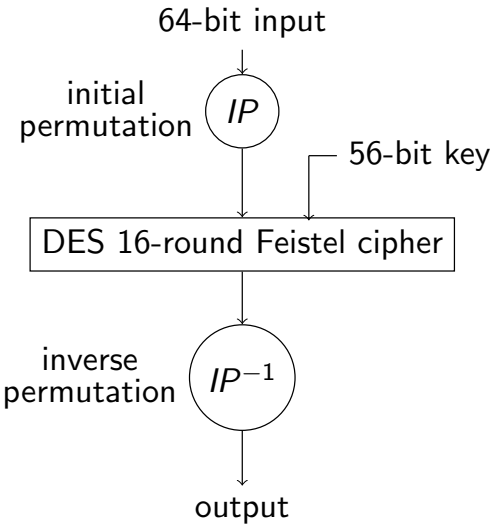
DES (cont.)

- The DES algorithm is public. The security is in the secret key.
- DES uses a 56-bit key. (The key is actually 64 bits long, but 8 of these are parity-check bits)
- DES is a 16-round Feistel network.
- The round functions of the DES Feistel network are created using graphic in Figure 10.4 b.
  - 32-bit input is expanded to 48 bits

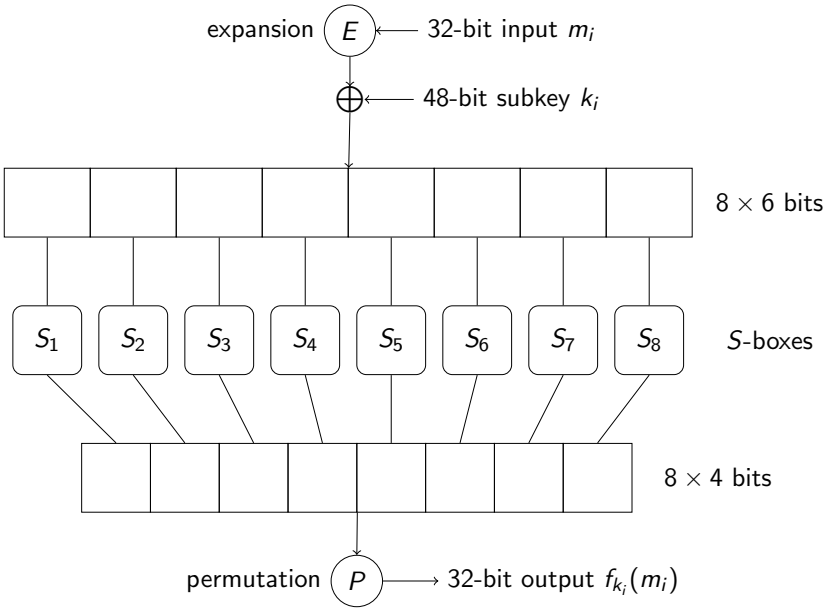
DES (cont.)

- The DES algorithm is public. The security is in the secret key.
- DES uses a 56-bit key. (The key is actually 64 bits long, but 8 of these are parity-check bits)
- DES is a 16-round Feistel network.
- The round functions of the DES Feistel network are created using graphic in Figure 10.4 b.
  - 32-bit input is expanded to 48 bits
  - "confusion" step results in 32-bit string

DES Schematics



(a) DES Structure



(b) Feistel inner function

Schematics for the DES

Next time

- We build some number theory tools, starting in Ch. 11.

Next time

- We build some number theory tools, starting in Ch. 11.

- Number theory is the basis of modern cryptogrpahy.