4 Shannon's Theory

4.1 Introduction

- Claude Shannon wrote some of the pivotal papers on modern cryptology theory around 1949:
  1. C E Shannon, "Communication Theory of Secrecy Systems", Bell System Technical Journal, Vol 28, Oct 1949, pp 656-715
  2. C E Shannon, "Prediction and Entropy of printed English", Bell System Technical Journal, Vol 30, Jan 1951, pp 50-64
- in these he developed the concepts of:
  - **entropy** of a message,
  - **redundancy** in a language,
  - theories about how much information is needed to break a cipher
  - computationally secure vs unconditionally secure ciphers
- he showed that the Vernam cipher (discussed later) is the only currently known unconditionally secure cipher, provided the key is truly random

Recall the three measures of security defined earlier:

1. **unconditional security**: cryptosystem cannot be broken, even with infinite computing power
2. **provable security**: breaking cryptosystem is *equivalent* to solving a difficult computational problem
3. **computational security**: no known method of breaking cryptosystem can be done in a "reasonable" amount of computing time

We have seen that neither a monoalphabetic substitution cipher nor the Vigenere cipher are even computationally secure against a ciphertext-only attack, given sufficient plaintext.

In this section, we develop the theory of cryptosystems which are unconditionally secure against a ciphertext-only attack.

4.2 Reminder of some probability

We make use of probability theory to study unconditional security. In each case, X and Y are random variables.

Definition 4.1 *We denote the probability that X takes on the value x by $p(x)$, and that Y takes on the value y by $p(y)$.*

Definition 4.2 *The* **joint probability** *$p(x, y)$ is the probability that X takes on the value x* **and** *Y takes on the value y.*

Definition 4.3 *The* **conditional probability** *$p(x|y)$ denotes the probability that X takes on the value x* **given that** *Y takes on the value y.*

Definition 4.4 *The random variables X and Y are said to be* ***independent*** *if $p(x,y) = p(x)p(y)$ for all possible values of x of X and y of Y.*

Remark 4.5 If $p(y) \neq 0$ then we the conditional probability $p(x|y)$ is given by

$$p(x|y) = \frac{p(x,y)}{p(y)}.$$

We can rearrange the above equation to give $p(x,y) = p(x|y)p(y)$ and by switching variables $p(y,x) = p(y|x)p(x)$. Then since $p(x,y) = p(y,x)$, we have that

$$p(x|y)p(y) = p(y|x)p(x).$$

Theorem 4.6 (Bayes' Theorem) *If $p(y) \neq 0$ then*

$$p(x|y) = \frac{p(x)p(y|x)}{p(y)}.$$

Corollary 4.7 *X and Y are independent variables if and only if $p(x|y) = p(x)$ and $p(y|x) = p(y)$, for all $x \in X$ and $y \in Y$.*

We will use Bayes' Theorem to make inferences about the plaintext, given some ciphertext, by letting $p(x|y)$ be the probability that the plaintext character was x, given that the ciphertext character is y.

4.3 Distributions in cryptosystems

Definition 4.8 *We denote the probability that plaintext x occurs by $p_{\mathcal{P}}(x)$ and the probability that key K is chosen by $p_{\mathcal{K}}(K)$.*

Remark 4.9 We assume that the key K and the plaintext x are independent events. This is reasonable, as the key is chosen in advance without knowledge of what the plaintext will be.

The two probability distributions $p_{\mathcal{P}}(x)$ for $x \in \mathcal{P}$ and $p_{\mathcal{K}}(K)$ for $K \in \mathcal{K}$ induce a probability distribution on the ciphertextspace $p_{\mathcal{C}}(y)$ for $y \in \mathcal{C}$, which we now estimate. For a key $K \in \mathcal{K}$ define

$$C(K) = \{e_K(x) \ : \ x \in \mathcal{P}\},$$

so $C(K)$ is the set of possible ciphertexts which are the result of applying the key K to some plaintext.

The following calculations can be made by anyone who knows the probability distributions of the plaintext and keytext spaces.

To calculate the probability of obtaining the ciphertext y, we take the sum over every key K that can lead to ciphertext y, the product of the probability that K was used and the probability that the plaintext $d_K(y)$ was used, that is the plaintext that encrypts to y using the key K. So for each ciphertext $y \in \mathcal{C}$,

$$p_{\mathcal{C}}(y) = \sum_{\{K : y \in C(K)\}} p_{\mathcal{K}}(K) p_{\mathcal{P}}(d_K(y)).$$

Thus we can compute the conditional probability that $y \in \mathcal{C}$ is the ciphertext, given that $x \in \mathcal{P}$ is the plaintext:

$$p_{\mathcal{C}}(y|x) = \sum_{\{K : x = d_K(y)\}} p_{\mathcal{K}}(K).$$

Hence, by using Bayes' Theorem, we can also compute the conditional probability that x is the plaintext, given that y is the ciphertext:

$$p_{\mathcal{P}}(x|y) \;=\; \frac{p_{\mathcal{P}}(x)p_{\mathcal{C}}(y|x)}{p_{\mathcal{C}}(y)} = \frac{p_{\mathcal{P}}(x) \displaystyle\sum_{\{K:x=d_K(y)\}} p_{\mathcal{K}}(K)}{\displaystyle\sum_{\{K:y\in C(K)\}} p_{\mathcal{K}}(K)p_{\mathcal{P}}(d_K(y))}.$$

Example 4.10 *Let $\mathcal{P} = \{a, b\}$ with $p_{\mathcal{P}}(a) = 1/4$ and $p_{\mathcal{P}}(b) = 3/4$. Let $\mathcal{K} = \{K_1, K_2, K_3\}$ with $p_{\mathcal{K}}(K_1) = 1/2$ and $p_{\mathcal{K}}(K_2) = p_{\mathcal{K}}(K_3) = 1/4$. Let $\mathcal{C} = \{1, 2, 3, 4\}$. Finally let the encryption matrix E_K be*

	a	b
K_1	1	2
K_2	2	3
K_3	3	4

Then the probability distribution of $p_{\mathcal{C}}$ is given by:

$$\begin{aligned} p_{\mathcal{C}}(1) &= 1/8, \\ p_{\mathcal{C}}(2) &= 3/8 + 1/16 = 7/16, \\ p_{\mathcal{C}}(3) &= 3/16 + 1/16 = 1/4, \\ p_{\mathcal{C}}(4) &= 3/16. \end{aligned}$$

Finally, we have the following conditional probability distributions:

$$\begin{array}{lcllcl} p_{\mathcal{P}}(a|1) & = & 1 & \quad p_{\mathcal{P}}(b|1) & = & 0 \\ p_{\mathcal{P}}(a|2) & = & 1/7 & \quad p_{\mathcal{P}}(b|2) & = & 6/7 \\ p_{\mathcal{P}}(a|3) & = & 1/4 & \quad p_{\mathcal{P}}(b|3) & = & 3/4 \\ p_{\mathcal{P}}(a|4) & = & 0 & \quad p_{\mathcal{P}}(b|4) & = & 1 \end{array}$$

Exercise 4.11 Consider a general mixed alphabet and assume that all 26! keys are equally likely. Show that $p_{\mathcal{P}}(a|D) = p_{\mathcal{P}}(a)$.

4.4 Perfect secrecy

Definition 4.12 *A cryptosystem has* perfect secrecy *if $p_{\mathcal{P}}(x|y) = p_{\mathcal{P}}(x)$ for all $x \in \mathcal{P}$ and $y \in \mathcal{C}$. That is, the probability that the plaintext is x, given that the ciphertext y is observed, is identical to the probability that the plaintext is x.*

Theorem 4.13 *Suppose the 26 keys in the Shift Cipher are used with equal probability $1/26$. Then for any plaintext probability distribution, the Shift Cipher has perfect secrecy.*

Proof: Recall that $\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathbb{Z}_{26}$ and for $0 \leqslant K \leqslant 25$ the encryption rule is $e_K(x) = x + K \pmod{26}$. First we compute the distribution $p_{\mathcal{C}}$. Let $y \in \mathbb{Z}_{26}$ then

$$\begin{aligned}
p_{\mathcal{C}}(y) &= \sum_{K \in \mathbb{Z}_{26}} p_{\mathcal{K}}(K) p_{\mathcal{P}}(d_K(y)) \\
&= \sum_{K \in \mathbb{Z}_{26}} (1/26) p_{\mathcal{P}}(y - K) \\
&= (1/26) \sum_{K \in \mathbb{Z}_{26}} p_{\mathcal{P}}(y - K) \\
&= (1/26) \sum_{y \in \mathbb{Z}_{26}} p_{\mathcal{P}}(y) \\
&= 1/26, \quad \text{for any } y \in \mathbb{Z}_{26}.
\end{aligned}$$

Next we have that $p_{\mathcal{C}}(y|x) = p_{\mathcal{K}}(y - x \bmod 26) = 1/26$ for every x and y (take the unique key $K = y - x$). Then using Bayes' Theorem we have perfect secrecy since

$$\begin{aligned}
p_{\mathcal{P}}(x|y) &= \frac{p_{\mathcal{P}}(x) p_{\mathcal{C}}(y|x)}{p_{\mathcal{C}}(y)} \\
&= \frac{p_{\mathcal{P}}(x)(1/26)}{(1/26)} \\
&= p_{\mathcal{P}}(x).
\end{aligned}$$

□

Remark 4.14 Therefore the Shift Cipher is "unbreakable" provided that a new random key is used to encrypt every plaintext character.

Exercise 4.15 Define a modified shift cipher to be the same as the shift cipher except that the key 0 is not allowed. Show that the modified shift cipher does not achieve perfect secrecy.

Let us investigate perfect secrecy in general. Without loss of generality, we can assume that $p_{\mathcal{C}}(y) \neq 0$ for all $y \in \mathcal{C}$ (if not, then y is never used and can be omitted). Assume the cryptosystem has perfect secrecy.

By Bayes' theorem, the condition that $p_{\mathcal{P}}(x|y) = p_{\mathcal{P}}(x)$ for all $x \in \mathcal{P}$ and $y \in \mathcal{C}$ is equivalent to $p_{\mathcal{C}}(y|x) = p_{\mathcal{C}}(y)$ for all $x \in \mathcal{P}$ and $y \in \mathcal{C}$.

Fix any $x \in \mathcal{P}$. For each $y \in \mathcal{C}$, we have

$$p_{\mathcal{C}}(y|x) = \sum_{\{K : x = d_K(y)\}} p_{\mathcal{K}}(K) = p_{\mathcal{C}}(y) \neq 0.$$

Hence for each $y \in \mathcal{C}$, there must be at least one key K such that $e_K(x) = y$.

It follows that $|\mathcal{K}| \geqslant |\mathcal{C}|$. In any cryptosystem we must always have $|\mathcal{C}| \geqslant |\mathcal{P}|$. Hence

$$|\mathcal{K}| \geqslant |\mathcal{C}| \geqslant |\mathcal{P}|.$$

In the boundary case, in which $|\mathcal{K}| = |\mathcal{C}| = |\mathcal{P}|$, Shannon gives a nice characterization of when perfect secrecy can be obtained.

Theorem 4.16 Shannon's Theorem *Suppose $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ is a cryptosystem, where $|\mathcal{K}| = |\mathcal{C}| = |\mathcal{P}|$. Then the cryptosystem provides perfect secrecy if and only if*

1. *every key is used with equal probability $1/|\mathcal{K}|$, and*
2. *for every $x \in \mathcal{P}$ and every $y \in \mathcal{C}$, there is a unique key K such that $e_K(x) = y$.*

Proof: First we assume the given cryptosystem provides perfect secrecy. Therefore, for each $x \in \mathcal{P}$ and $y \in \mathcal{C}$ there must be at least one key K such that $e_K(x) = y$. So we have

$$\begin{aligned} |\mathcal{C}| &= |\{e_K(x) : K \in \mathcal{K}\}| \\ &\leqslant |\mathcal{K}|. \end{aligned}$$

Using $|\mathcal{C}| = |\mathcal{K}|$ (this is the boundary case) we find that $|\{e_K(x) : K \in \mathcal{K}\}| = |\mathcal{K}|$. Hence, for each $x \in \mathcal{P}$ and $y \in \mathcal{C}$, there must be a unique key K such that $e_K(x) = y$. This proves condition (2).

Next, prove condition (1). Denote $n = |\mathcal{K}|$. Let $\mathcal{P} = \{x_i : 1 \leqslant i \leqslant n\}$ and fix a $y \in \mathcal{C}$. We can name the keys $K_1, K_2, \ldots, K_n$ in such a way that $e_{K_i}(x_i) = y$, for $1 \leqslant i \leqslant n$. Using Bayes' Theorem and assumption of perfect secrecy, we have

$$\begin{aligned} p_{\mathcal{P}}(x_i|y) &= \frac{p_{\mathcal{P}}(x_i) p_{\mathcal{C}}(y|x_i)}{p_{\mathcal{C}}(y)} \\ &= \frac{p_{\mathcal{P}}(x_i) p_{\mathcal{K}}(K_i)}{p_{\mathcal{C}}(y)}. \end{aligned}$$

Using $p_{\mathcal{P}}(x_i|y) = p_{\mathcal{P}}(x_i)$ we obtain $p_{\mathcal{K}}(K_i) = p_{\mathcal{C}}(y)$, for $1 \leqslant i \leqslant n$. Therefore, we must have

$$p_{\mathcal{K}}(K_i) = 1/|\mathcal{K}|.$$

Conversely, suppose the two hypothesized conditions (1) and (2) are satisfied. Proof that these imply perfect secrecy mirror the proof of Theorem 4.13 (perfect secrecy of the Shift cipher).

□

Exercise 4.17 Let n be a positive integer. A *Latin square* of order n is an $n \times n$ array L of integers $1, 2, \ldots, n$ such that every one of the n integers occurs exactly once in each row and each column of L. We often denote the entry in row i and column j of the Latin square L by $L(i, j)$. An example of a Latin square of order 4 is:

1	2	3	4
2	3	4	1
3	4	1	2
4	1	2	3

Given any Latin square L of order n we can define a related cryptosystem. Take $\mathcal{P} = \mathcal{C} = \mathcal{K} = \{1, 2, \ldots, n\}$. For the key i and plaintext j where $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant n$, the encryption rule e_i is defined to be $e_i(j) = L(i, j)$. Each key is used with equal probability. Show that this Latin square cryptosystem achieves perfect secrecy.

4.5 One-time Pads

Shannon's theorem (Theorem 4.16) offers a rigorous proof of the perfect secrecy of Vernam's one-time pad cipher (Gilbert Vernam, 1917).

Algorithm 4.18 Vernam's one-time pad algorithm
Let $n \geqslant 1$ be an integer and take $\mathcal{P} = \mathcal{C} = \mathcal{K} = (\mathbb{Z}_2)^n$.

Let the key $K = (K_1, K_2, \ldots, K_n)$ be a random sequence of bits generated by some "good" random generator. Let the plaintext be $x = (x_1, x_2, \ldots, x_n)$ and ciphertext $y = (y_1, y_2, \ldots, y_n)$. Define

$$e_K(x) = (x_1 + K_1, x_2 + K_2, \ldots, x_n + K_n) \mod 2;$$

and

$$d_K(y) = (y_1 + K_1, y_2 + K_2, \ldots, y_n + K_n) \mod 2.$$

□

Using Theorem 4.16, we can see that the One-time Pad provides perfect secrecy.
Vernam hoped that his idea would have widespread commercial use. Unfortunately, there are major disadvantages to unconditionally secure cryptosystems such as the one-time pad. The fact that $|\mathcal{K}| \geqslant |\mathcal{P}|$ means that the amount of key that must be communicated securely is at least as big as the amount of plaintext.

However, the One-time Pad has seen application in military and diplomatic context, where unconditional security may be of great importance.

4.6 Random number generation

Random numbers find numerous uses in cryptography, especially in authentication schemes, session key generation, conventional and public-key cryptography.

Perfect random generation is impossible by a deterministic, finite-state device, like a computer. Usually we have to generate *pseudorandom* numbers with a deterministic source. Resulting numbers must be unpredictable, independent and uniformly distributed.
Two pseudorandom number generators are:

Linear congruence generator:

$$x \rightarrow ax + b \bmod m$$

Blum-Blum-Shub generator:

$$x \rightarrow x^2 \bmod pq$$

5 Modern Cryptography

- From now on we will concentrate on modern encryption systems.
- We will usually consider a message as a sequence of bits (eg as a series of ASCII characters concatenated).
- Good Cipher Design
  - **Avalanche effect:** small input changes $\Longrightarrow$ large output changes
  - **Completeness effect:** small parts of output depend on large parts of input
- Bad Cipher Design
  - based on what appears to be random, but isn't (eg LFSRs)

5.1 Substitution-Permutation Ciphers

- In a block cipher the message is broken into blocks, each of which is then encrypted. Most modern ciphers we will study are of this form.
- In 1949, Claude Shannon introduced the idea of substitution-permutation (S-P) networks, which are the modern forms of substitution-transposition product ciphers.

  1. **Substitution Operation**
     - A binary word of length n is replaced by some other binary word of length n.
     - The substitution function maps the 2^n binary words of length n to the 2^n binary words of length n and the whole substitution function forms the key.
     - If we use n-bit words, the total number of possible keys is $2^n!$. This grows very rapidly as n gets large.
     - We can think of this as a large lookup table, with n address lines (hence 2^n addresses), each line n bits wide being the output value.
     - We will call the substitution function an **S-box**.

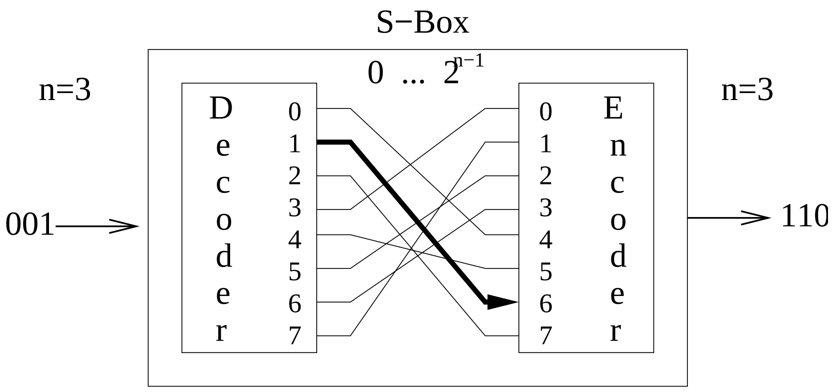

Figure 12: Substitution operation

2. **Permutation Operation**
   - A binary word of length n has its bits rearranged (permuted).
   - The permutation forms the key.
   - If we use n-bit words, the total number of possible keys is $n!$, which grows more slowly than $2^n!$, and hence is less secure than the substitution step.
   - The rearrangement is equivalent to a wire-crossing in practice (though is much harder to do in software).
   - We will call the permutation function a **P-box**.

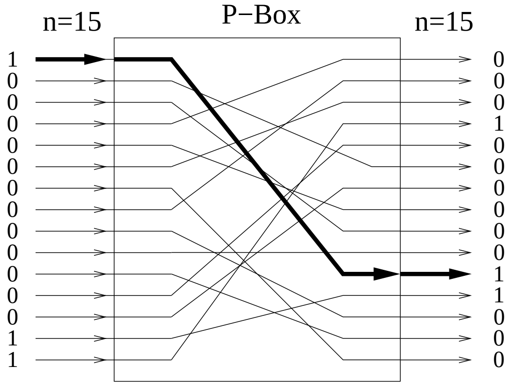

Figure 13: Permutation operation

5.2 Practical Substitution-Permutation Networks

- In practice, we need to decrypt messages, as well as to encrypt them. Hence either:
  - we must define inverses for each S and P-box, but this doubles the code/hardware needed, or
  - we must define a structure that is easy to reverse, so we can use basically the same code or hardware for both encryption and decryption.
- Horst Feistel (IBM) devised such a structure in the early 1970's, called a **Feistel** cipher.

5.2.1 Reminder of some binary mathematics

You are assumed to be familiar with the basic concepts of binary representation of numbers and simple binary operations such as *and* ($\wedge$), *or* ($\vee$), *not* ($\sim$) and *exclusive-or* ($\oplus$).

Remark 5.1 *Here are truth tables for these simple binary operations:*

p	q	$p \wedge q$	$p \vee q$	$\sim p$	$p \oplus q$
1	1	1	1	0	0
1	0	0	1	0	1
0	1	0	1	1	1
0	0	0	0	1	0

Remark 5.2 *Usually, computers and other electronic means of data transmission store data as* words, *which are bitstrings of a certain length. For example, many computers use words of length 32 bits.*

Remark 5.3 *When we speak of applying an operation* bit-wise *to a word or pair of words, we really mean that the operation is performed on each bit (or pair of corresponding bits), usually in parallel.*

Exercise 5.4 If a and b are bitstrings defined as follows, find $a \wedge b$ and $a \oplus b$.

$$\begin{aligned} a &= 0\,0\,1\,1\,0\,1\,0\,1\,1\,1\,1\,0\,1\,0\,1\,0\,1 \\ b &= 1\,1\,1\,0\,0\,1\,1\,1\,0\,0\,0\,1\,1\,1\,1\,0\,1 \\ a \wedge b &= \\ a \oplus b &= \end{aligned}$$

5.2.2 Feistel Ciphers

- In an S-P cipher the block size cannot be too small, as the resulting cipher might not be sufficiently complicated. However permuting the bits of a large block is not practical.
- Feistel's idea was to approximate an ideal block cipher system, by doing permutations on smaller components and then combining these smaller components to give a large overall block size.
- The input to a Feistel cipher is a plaintext of length $2n$ and a key K.
- The plaintext is divided into two parts: L_0 (left) and R_0 (right).
- The two halves are passed through r rounds of processing and then combined to produce the ciphertext block.
- The input to the i^{th} round is (L_{i-1}, R_{i-1}) (the output from the $(i-1)^{\text{th}}$ round) and a subkey K_i (usually generated from the initial key K).
- At the i^{th} round, the new data is obtained by:

$$\begin{aligned} L_i &= R_{i-1} \\ R_i &= L_{i-1} \oplus f(R_{i-1},\, K_i) \end{aligned}$$

  where $\oplus$ means exclusive-or and f is a function which incorporates one stage of the S-P network, controlled by the i^{th} subkey. The final ciphertext block is obtained by writing the two outputs of the r^{th} round in the order $R_r L_r$.

- In practice, we link a number of these stages together (typically 16 rounds) to form the full cipher.
- Decryption is the reverse of this process: work backwards through the hardware/software.

3-stages of a Feistal cipher are illustrated below. The plaintext is L_0R_0. The Feistal cipher equations become:

$$\begin{aligned} L_1 &= R_0 & L_2 &= R_1 & L_3 &= R_2 \\ R_1 &= L_0 \oplus f(R_0,\, K_1) & R_2 &= L_1 \oplus f(R_1,\, K_2) & R_3 &= L_2 \oplus f(R_2,\, K_3) \end{aligned}$$

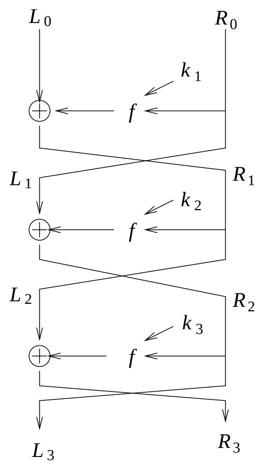

Figure 14: A 3-stage Feistel cipher

The final output of a 3-stage Feistal cipher would then be written as R_3L_3.

Exercise 5.5 Consider a 3-stage Feistal cipher as depicted above. Let the function f be defined by $f((x_1, x_2, x_3, x_4),\, \pi) = (x_{\pi(1)}, x_{\pi(2)}, x_{\pi(3)}, x_{\pi(4)})$. Let the initial key K be the permutation (1234) and let each subkey k_i be the permutation obtained by applying the initial key permutation i times. Determine the ciphertext for the plaintext 00011011.

5.2.3 Design parameters

- **Block and key size:** Larger block and key sizes give greater security but slower encryption and decryption. 64 bits is currently the accepted block size, and 128 bits is currently the accepted key size.
- **Number of rounds:** Typical size is 16 rounds.
- **Subkey generation and round function:** The greater the complexity of the algorithm, the greater the difficulty of cryptanalysis.
- **Encryption and decryption speeds:** Of major concern in practical applications!
- **Ease of analysis:** Algorithm should be easy to analyse (not break!) in order to understand strengths and weaknesses, and increase user confidence.

Some practical implementations include:

System	Block size	Key size	num rounds
DES	64	56	16
Double-DES	64	112	32
Triple-DES	64	168	48
Blowfish	64	32...448	16
RC5	32,64,128	0...2040	variable
CASR-128	64	40...128	16
RC2	64	8...1024	16

6 The Data Encryption Standard

6.1 Introduction

- In May 1973 (again in Aug 1974) the National Bureau of Standards (now the National Institute of Standards and Technology, NIST) called for possible encryption algorithms for use in unclassified government applications.
- NBS suggested the following guidelines:
  1. High level of security
  2. Complete specification and easy to understand
  3. Security must be based on the key, not on the secrecy of the algorithm
  4. System available to all users
  5. Easily adaptable for diverse applications
  6. Economical implementation in electronic devices
  7. Algorithm efficient to use
  8. Algorithm must be easy to validate
  9. Algorithm must be exportable
- The guidelines were meant to enhance public confidence and encourage widespread use of the cryptosystem.
- The response was mostly disappointing, but IBM submitted their Lucifer design, and following a period of redesign and comment it became the Data Encryption Standard (DES).
- DES was adopted as a (US) federal standard in Nov 1976, published by NBS as a hardware only scheme in Jan 1977 and by ANSI for both hardware and software standards in ANSI X3.92-1981 (also X3.106-1983 modes of use).
- Subsequently it has been widely adopted and is now published in many standards around the world (eg Australian Standard AS2805.5-1985).
- One of the largest users of the DES was the banking industry, particularly with EFT, and EFTPOS.
- Although the standard is public, the design criteria used are classified and have yet to be released.
- There has been considerable controversy over the design, particularly in the choice of a 56-bit key:
  1. W Diffie, M Hellman "Exhaustive Cryptanalysis of the NBS Data Encryption Standard", IEEE Computer 10(6), June 1977, pp74-84
  2. M Hellman "DES will be totally insecure within ten years", IEEE Spectrum 16(7), Jul 1979, pp 31-41
- Recent analysis has shown that despite this, the choice was appropriate, and that DES is well designed.

- The DES was regularly reviewed and renewed. The last renewal (of regular DES) was in Jan 1994, as soon thereafter rapid advances in computing speed rendered the 56 bit key susceptible to exhaustive key search, as predicted by Diffie and Hellman. In 1999 the DES was renewed as a standard that specified the use of Triple DES, and this standard was withdrawn in 2005.
- There have been demonstrated breaks:
  - 1997 on a large network of computers in a few months
  - 1998 on dedicated hardware in a few days
  - 1999 above combined in 22 hrs!
- The DES has also been theoretically broken using a method called Differential or Linear Cryptanalysis, however in practise this is unlikely to be a problem.
- When it was no longer considered "secure" for transactions, it was replaced by the AES.

6.2 Overview of DES

The DES algorithm occurs in three steps:

1. Given a plaintext $x = (x_1, x_2, \ldots, x_{64})$, a bitstring x_0 is constructed by permuting the bits of x according to a fixed initial permutation IP (see Figure 15). So

$$IP(x_1, x_2, \ldots, x_{64}) = (x_{58}, x_{50}, x_{42}, \ldots, x_7).$$

We write $x_0 = IP(x) = L_0R_0$, where L_0 comprises the first 32 bits of x_0 and R_0 the last 32 bits.

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Figure 15: Initial Permutation IP

2. **16** iterations of a certain Feistel algorithm are then computed. We compute L_iR_i, $1 \leqslant i \leqslant 16$, according to the following rule:

$$\begin{aligned} L_i &= R_{i-1} \\ R_i &= L_{i-1} \oplus f(R_{i-1}, K_i), \end{aligned}$$

where $\oplus$ denotes the exclusive-or of two bitstrings and $K_1, K_2, \ldots, K_{16}$ are each bitstrings of length 48 computed as a function of the key K. This is shown in Figure 16. The function f is discussed below.

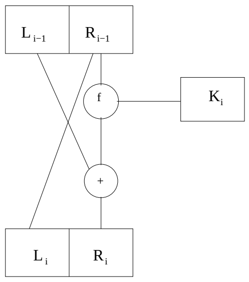

Figure 16: One round of DES encryption

3. Apply the inverse permutation IP^{-1} to the bitstring $R_{16}L_{16}$, obtaining the ciphertext y. Thus $y = IP^{-1}(R_{16}L_{16})$ (see Figure 17).

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Figure 17: The Inverse Permutation IP^{-1}

The initial permutation and the final permutation have little impact on the security of DES. The purpose of these permutations was to ease the input/output transfer of data blocks between preexisting hardware and newly created DES implementations.

6.3 The function f

Recall that in Step 2 of DES described above, we had

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i).$$

The function f takes as input a first argument A, which is a bitstring of length 32, and a second argument J which is a bitstring of length 48, and produces as output a bitstring of length 32. Thus

$$f: \ \{0,1\}^{32} \times \{0,1\}^{48} \to \{0,1\}^{32}.$$

1. The first argument A is "expanded" to a bitstring of length 48 according to a fixed *expansion function* $E: \ \{0,1\}^{32} \to \{0,1\}^{48}$. $E(A)$ is a permutation of A with 16 bits occurring twice; see Figure 18.

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Figure 18: E **bit-selection table**

2. Compute $E(A) \oplus J$ and write the result as the concatenation of eight 6-bit strings $B = B_1B_2B_3B_4B_5B_6B_7B_8$.

3. This step uses eight *S-boxes* $S_1 \dots S_8$ (see Figure 20 for the S-boxes). Each S_i is a fixed 4×16 array whose entries come from the integers $0, 1, \dots 15$. Think of S_i as a function

$$S_i : \ \{0,1\}^2 \times \{0,1\}^4 \to \{0,1\}^4.$$

Given a bitstring of length six, say $B_j = b_1b_2b_3b_4b_5b_6$, we compute $S_j(B_j)$ as follows:

The two bits b_1b_6 determine the binary representation of a row r of S_j and the four bits $b_2b_3b_4b_5$ determine the binary representation of a column c of S_j. Then $S_j(B_j) = S_j(r, c)$, written in binary as a bitstring of length four. In this fashion, we compute $C_j = S_j(B_j)$ for $1 \leqslant j \leqslant 8$.

4. The string $C = C_1C_2C_3C_4C_5C_6C_7C_8$ of length 32 is permuted according to a fixed permutation P (see Figure 19). The resulting bitstring $P(C)$ is defined to be $f(A, J)$.

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

Figure 19: P

6.4 Computation of key schedule

- Finally, we need to describe the computation of the subkeys (called the key schedule) from the initial key K.

- K is a bitstring of length 64.
- 56 bits of K are used in the subkeys and 8 bits are parity-check bits (for *error-detection*).
  - The bits in positions $8, 16, \ldots, 64$ are defined so each 8-bit byte has an *odd* number of 1's.
  - Hence, a single error can be detected within each byte.
  - The parity-check bits are ignored in the computation of the subkeys.
- The complete key schedule is obtained via the following procedure. The full schedule is given at the end of this section.

Steps in computation of key schedule:

1. Given a 64-bit key K, ignore the parity-check bits and permute the bits of K according to the following (fixed) permutation PC_1. We will write $PC_1(K) = C_0D_0$, where C_0 comprises the first 28 bits of $PC_1(K)$ and D_0 the last 28 bits.

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

PC_1

2. For i ranging from 1 to 16, compute

$$\begin{aligned} C_i &= LS_i(C_{i-1}) \\ D_i &= LS_i(D_{i-1}), \end{aligned}$$

and $K_i = PC_2(C_iD_i)$. LS_i represent a cyclic shift (to the left) of either one or two positions, depending on the value of i: shift one position if $i = 1, 2, 9$ or 16, and shift two positions otherwise. PC_2 is the following (fixed) permutation.

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

PC_2

6.5 Summary of DES

1. Compute subkeys from initial 64-bit key K.
   (i) Apply PC_1 to 64-bit string K to get 56-bit string C_0D_0 (each 28 bits).
   (ii) Left shift C_0 and D_0 to get $C_1, D_1, C_2, D_2, \ldots, C_{16}, D_{16}$ and apply PC_2 to each C_iD_i to get K_i (a 48-bit string) for $i = 1, 2 \ldots, 16$.
2. Apply IP to (plaintext) bitstring x of length 64 to get L_0R_0.
3. Apply 16 rounds of Feistel algorithm to get $R_{16}L_{16}$. In each round, apply the following steps to compute $f(R_{i-1}, K_i)$:
   (a) Expand 32-bit string R_{i-1} to 48-bit string.
   (b) Add 48-bit string from step (a) to 48-bit subkey K_i.
   (c) Split resulting 48-bit string into 8 groups of 6 and use S-boxes to replace each 6-bit string with a 4-bit string, thereby obtaining a 32-bit string.
   (d) Apply permuation P to this 32-bit string to get the output $f(R_{i-1}, K_i)$.
4. Apply IP^{-1} to bitstring $R_{16}L_{16}$ to get (ciphertext) bitstring y of length 64.

6.6 Hexadecimal notation

Remark 6.1 In computer science (and related mathematics) it is often convenient to work in *hexadecimal* notation (or *hex*): this is a system in which numbers are represented base 16.

Remark 6.2 In usual base 10 operations there are 10 digits (`0 - 9`); in base 2 operations there are 2 digits (`0, 1`). In base 16 there are 16 digits (`0 - 9, a, b, c, d, e, f`). The digit `a` in base 16 equals 10 in base 10, `b` equals 11, `c` equals 12, `d` equals 13, `e` equals 14 and `f` equals 15.

Remark 6.3 Be a bit careful; when working base 16, the number 123 means $1 \times 16^2 + 2 \times 16 + 3 = 256 + 32 + 3 = 291$ base 10.

Remark 6.4 Base 16 is often convenient because there are exactly 16 distinct bitstrings of length 4, so there is a convenient correspondence between bitstrings of length 4 and hexadecimal digits. Clearly, 0000 equals 0 (base 10) which equals `0` (hex), 1000 equals 8 (base 10) which equals `8` (in hex), and 1110 equals $8 + 4 + 2 = 14$ (base 10) which equals `e` (hex).

Remark 6.5 We will often write lengthy strings of bits using hex notation; to convert from hex to bits, convert each hex character to its 4-bit equivalent. For example, `2f`= 00101111.

6.7 Using DES

Definition 6.6 A cyclic shift to the left (or *left rotate*) by one position involves moving each bit in a bitstring one place to the left, with the original leftmost bit becoming the new rightmost bit.

Exercise 6.7 Given the bitstring 11100110, cyclic shift it left by one place, then cyclic shift the answer left by 4 places.

Example 6.8 Apply the above key computation algorithm to obtain K_1 from the initial (hexadecimal) key 133457799BBCDFF1.

hex	1	3	3	4	5	7	7	9
binary	0001	0011	0011	0100	0101	0111	0111	1001

hex	9	B	B	C	D	F	F	1
binary	1001	1011	1011	1100	1101	1111	1111	0001

So $K =$ 0001 0011 0011 0100 0101 0111 0111 1001 1001 1011 1011 1100 1101 1111 1111 0001

$PC_1(K) = C_0D_0$ where
$C_0 =$ 1111 0000 1100 1100 1010 1010 1111 and $D_0 =$ 0101 0101 0110 0110 0111 1000 1111.

Left shift by one to get C_1 and D_1, so
$C_1 =$ 111 0000 1100 1100 1010 1010 1111 1 and $D_1 =$ 101 0101 0110 0110 0111 1000 1111 0.

$PC_2(C_1D_1) = K_1 =$ 0001 1011 0000 0010 1110 1111 1111 1100 0111 0000 0111 0010.

Example 6.9 Perform the first round of DES encrpytion on the (hexadecimal) plaintext 0123456789ABCDEF using the (hexadecimal) key 133457799BBCDFF1.

First convert the plaintext to binary, and note that we have calculated K_1 in Example 6.8.

hex	0	1	2	3	4	5	6	7
binary	0000	0001	0010	0011	0100	0101	0110	0111

hex	8	9	A	B	C	D	E	F
binary	1000	1001	1010	1011	1100	1101	1110	1111

Permute the 64-bit string to get $IP(x) = L_0R_0$ where
$L_0 =$ 1100 1100 0000 0000 1100 1100 1111 1111 and $R_0 =$ 1111 0000 1010 1010 1111 0000 1010 1010.

To calculate $f(R_0, K_1)$:

(a) Expand R_0 to get
$$E(R_0) = 0111\ 1010\ 0001\ 0101\ 0101\ 0101\ 0111\ 1010\ 0001\ 0101\ 0101\ 0101.$$

(b) Use the key K_1 to calculate
$$E(R_0) \oplus K_1 = 011000\ 010001\ 011110\ 111010\ 100001\ 100110\ 010100\ 100111.$$

(c) Use the S-boxes to replace each 6-bit string with a 4-bit string. For the first six bits $b_1b_2b_3b_4b_5b_6 =$ 011000 we look up row $b_1b_6 = 00$ and column $b_2b_3b_4b_5 = 1100$ in the S-box S_1. This is 5 which gives 0101 in binary. Repeat the procedure for the other 6-bit strings and the resulting 32-bit string is
$$C = 0101\,1100\,1000\,0010\,1011\,0101\,1001\,0111.$$

(d) This is permuted to get
$$f(R_0, K_1) = P(C) = 0010\,0011\,0100\,1010\,1010\,1001\,1011\,1011.$$

Thus $L_1 = R_0$ and $R_1 = L_0 \oplus f(R_0, K_1) =$ 1110 1111 0100 1010 0110 0101 0100 0100.

6.8 Decryption

Decryption is done using the same algorithm as encryption, starting with y as the input, but using the key schedule $K_{16}, K_{15}, \ldots, K_1$ in reverse order. The output will be the plaintext x.

6.9 The DES Controversy

When DES was proposed as a standard, there was considerable criticism.

One objection to DES concerned the S-boxes. All computations in DES, with the exception of the S-boxes, are *linear.* The S-boxes, being the non-linear component of the cryptosystem, are vital to its security.

Do the S-boxes contain hidden trapdoors, allowing the National Security Agency (NSA) to easily decrypt data?

In 1976, the NSA asserted that the following properties of the S-boxes are design criteria:

1. Each row of each S-box is a permutation of the integers $0, 1, 2, \ldots, 15$.
2. No S-box is a linear or affine function of its inputs.
3. Changing one input bit to an S-box causes at least two output bits to change.
4. For any S-box and any input x, $S(x)$ and $S(x \oplus 001100)$ differ in at least two bits.
5. For any S-box, for any input x, and for any e, $f \in \{0, 1\}$, $S(x) \neq S(x \oplus 11ef00)$.
6. For any S-box, if one input bit is fixed, and we look at the value of one fixed output bit, the number of inputs for which this output bit equals 0 will be "close to" the number of inputs for which the output bit equal 1.

It is not publicly known if further design criteria were used in the construction of the S-boxes.

The most pertinent criticism of DES is that the size of the keyspace, 2^{56}, is too small to be really secure. Various special-purpose machines have been proposed for a known plaintext attack, which would essentially perform an exhaustive search for the key.

1977	Diffie and Hellman	\$20,000,000 computer	finds the key in a day.
1993	Michael Wiener	\$1,000,000 computer	finds the key in 3.5 hours.
2006	University of Bochum and Kiel	\$10,000 computer	breaks DES in around 7 days

6.10 Applications of DES

- Even though the description of DES is quite lengthy, it can be implemented very efficiently, either in hardware or in software.

- One very important application of DES was in banking transactions, using standards developed by the American Bankers Association. DES was used to encrypt personal identification numbers (PINs) and account transactions carried out by automated teller machines (ATMs). Until quite recently, DES was also used by the Clearing House Interbank Payments System (CHIP) to authenticate transactions involving over \$1.5 $\times 10^{12}$ per week.
- DES was also widely used in government organisations, such as the Department of Energy, the Justice Department, and the Federal Reserve System.
- The federal standard FIPS 46-3 was withdrawn by the National Institute of Standards and Technology in 2005.

6.11 Variants of DES

- A number of DES variants are still used (eg on the internet).
- **Double DES**:
  - requires two keys K_1 and K_2, applied to a plaintext
  - $x \to e_{K_1}(x) \to e_{K_2}(e_{K_1}(x))$
- **Triple DES**:
  - requires three keys K_1, K_2 and K_3, applied to a plaintext
  - $x \to e_{K_1}(x) \to e_{K_2}(e_{K_1}(x)) \to e_{K_3}(e_{K_2}(e_{K_1}(x)))$
  - no known practical methods of attack: brute force is impossible
- **Triple DES with two keys**:
  - requires two keys K_1 and K_2, applied to a plaintext
  - $x \to e_{K_1}(x) \to E_{K_2}(e_{K_1}(x)) \to e_{K_1}(E_{K_2}(e_{K_1}(x)))$

  It can be shown that double DES is not reducible to (single) DES.

  That is, given keys K_1 and K_2, there does not exist a single key K such that $e_K(x) = e_{K_2}(e_{K_1}(x))$.

6.12 Weak keys

- In many block ciphers, not all keys are equally good.
- Some keys result in reduced cipher complexity, by exhibiting certain regularities in encryption or a poor level of encryption.
- Such keys result in the same subkey being generated in more than one round.
- **Weak keys**
  - same subkey is generated for **every** round
  - DES has 4 weak keys (eg string of all 0's, string of all 1's)
- **Semi-Weak keys**

  - only two distinct subkeys are generated on alternate rounds
  - DES has 12 of these (in 6 pairs)

- **Demi-Semi-Weak keys**
  - only four distinct subkeys generated
- Must avoid usage of any weak keys!
- It is not a significant problem for DES, as they form such a small fraction of all available keys.

6.13 Key schedule and S-boxes

S_1	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_5	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Figure 20: The eight S-boxes for DES

The key schedule for the 16 rounds

K_1	10	51	34	60	49	17	33	57	2	9	19	42
	3	35	26	25	44	58	59	1	36	27	18	41
	22	28	39	54	37	4	47	30	5	53	23	29
	61	21	38	63	15	20	45	14	13	62	55	31
K_2	2	43	26	52	41	9	25	49	59	1	11	34
	60	27	18	17	36	50	51	58	57	19	10	33
	14	20	31	46	29	63	39	22	28	45	15	21
	53	13	30	55	7	12	37	6	5	54	47	23
K_3	51	27	10	36	25	58	9	33	43	50	60	18
	44	11	2	1	49	34	35	42	41	3	59	17
	61	4	15	30	13	47	23	6	12	29	62	5
	37	28	14	39	54	63	21	53	20	38	31	7
K_4	35	11	59	49	9	42	58	17	27	34	44	2
	57	60	51	50	33	18	19	26	25	52	43	1
	45	55	62	14	28	31	7	53	63	13	46	20
	21	12	61	23	38	47	5	37	4	22	15	54
K_5	19	60	43	33	58	26	42	1	11	18	57	51
	41	44	35	34	17	2	3	10	9	36	27	50
	29	39	46	61	12	15	54	37	47	28	30	4
	5	63	45	7	22	31	20	21	55	6	62	38
K_6	3	44	27	17	42	10	26	50	60	2	41	35
	25	57	19	18	1	51	52	59	58	49	11	34
	13	23	30	45	63	62	38	21	31	12	14	55
	20	47	29	54	6	15	4	5	39	53	46	22
K_7	52	57	11	1	26	59	10	34	44	51	25	19
	9	41	3	2	50	35	36	43	42	33	60	18
	28	7	14	29	47	46	22	5	15	63	61	39
	4	31	13	38	53	62	55	20	23	37	30	6
K_8	36	41	60	50	10	43	59	18	57	35	9	3
	58	25	52	51	34	19	49	27	26	17	44	2
	12	54	61	13	31	30	6	20	62	47	45	23
	55	15	28	22	37	46	39	4	7	21	14	53
K_9	57	33	52	42	2	35	51	10	49	27	1	60
	50	17	44	43	26	11	41	19	18	9	36	59
	4	46	53	5	23	22	61	12	54	39	37	15
	47	7	20	14	29	38	31	63	62	13	6	45
K_{10}	41	17	36	26	51	19	35	59	33	11	50	44
	34	1	57	27	10	60	25	3	2	58	49	43
	55	30	37	20	7	6	45	63	38	23	21	62
	31	54	4	61	13	22	15	47	46	28	53	29

K_{11}	25	1	49	10	35	3	19	43	17	60	34	57
	18	50	41	11	59	44	9	52	51	42	33	27
	39	14	21	4	54	53	29	47	22	7	5	46
	15	38	55	45	28	6	62	31	30	12	37	13
K_{12}	9	50	33	59	19	52	3	27	1	44	18	41
	2	34	25	60	43	57	58	36	35	26	17	11
	23	61	5	55	38	37	13	31	6	54	20	30
	62	22	39	29	12	53	46	15	14	63	21	28
K_{13}	58	34	17	43	3	36	52	11	50	57	2	25
	51	18	9	44	27	41	42	49	19	10	1	60
	7	45	20	39	22	21	28	15	53	38	4	14
	46	6	23	13	63	37	30	62	61	47	5	12
K_{14}	42	18	1	27	52	49	36	60	34	41	51	9
	35	2	58	57	11	25	26	33	3	59	50	44
	54	29	4	23	6	5	12	62	37	22	55	61
	30	53	7	28	47	21	14	46	45	31	20	63
K_{15}	26	2	50	11	36	33	49	44	18	25	35	58
	19	51	42	41	60	9	10	17	52	43	34	57
	38	13	55	7	53	20	63	46	21	6	39	45
	14	37	54	12	31	5	61	30	29	15	4	47
K_{16}	18	59	42	3	57	25	41	36	10	17	27	50
	11	43	34	33	52	1	2	9	44	35	26	49
	30	5	47	62	45	12	55	38	13	61	31	37
	6	29	46	4	23	28	53	22	21	7	63	39

7 International Data Encryption Algorithm

7.1 Reminder of some mathematics

Definition 7.1 A *binary operation* $\circ$ on a non-empty set S is a mapping which associates with each ordered pair (a, b) of elements of S a uniquely defined element $a \circ b$ of S. Thus $\forall a, b \in S$, $\exists$ a unique $c \in S$ such that $c = a \circ b$.

Remark 7.2 We say that the set S is *closed* under the operation $\circ$.

Example 7.3 Addition and multiplication are both examples of binary operations on $\mathbb{Z}$.

Example 7.4 We can represent a binary operation $\circ$ on a set $A = \{a, b, c, d, e\}$ using a table as follows.

$\circ$	a	b	c	d	e
a	a	b	c	d	e
b	b	c	d	e	a
c	c	d	e	a	b
d	d	e	a	b	c
e	e	a	b	c	d

For example, $a \circ c = c$, $d \circ b = e$.

Exercise 7.5 Which of the following represent binary operations?

1. addition on $\mathbb{Z}_{26}$?
2. addition on $S = \{0, 1, 2, 3, 4\}$?
3. addition on the set of even integers?
4. addition on the set of odd integers?
5. Example 7.4 above?

Remark 7.6 A binary operation $\circ$ on a set S is said to be *commutative* if $a \circ b = b \circ a$ for all $a, b \in S$.

Exercise 7.7 Which of the following binary operations are commutative?

1. addition on $\mathbb{R}$?
2. division on $\mathbb{R}$?
3. Example 7.4 above?

Remark 7.8 A binary operation $\circ$ on a set S is said to be *associative* if $(a \circ b) \circ c = a \circ (b \circ c)$ for all $a, b, c \in S$.

Exercise 7.9 Which of the following binary operations are associative?

1. addition on $\mathbb{R}$?
2. Example 7.4 above?
3. $\circ$ is the binary operation on $\mathbb{R}$ defined by $a \circ b = a + 2b$ for all $a, b \in \mathbb{R}$.

Definition 7.10 A set S is said to have an *identity* element with respect to the binary operation $\circ$ on S if there exists an element $u \in S$ such that $u \circ x = x \circ u = x$ for all $x \in S$. We call such an element u the identity element.

Exercise 7.11 What is the identity element for following binary operations.

1. addition on $\mathbb{R}$?
2. multiplication on $\mathbb{R}$?
3. Example 7.4 above?

Theorem 7.12 *If an identity u exists on a set S with binary operation $\circ$, then u is unique.*

Definition 7.13 Let u be the identity element for the binary operation $\circ$ defined on the set S. An element $y \in S$ is called an *inverse* of $x \in S$ if $x \circ y = y \circ x = u$.

Exercise 7.14 If S is the set on which each of the following binary operations is defined, find the inverse for each $x \in S$.

1. addition on $\mathbb{R}$?
2. multiplication on $\mathbb{R}$?
3. Example 7.4 above?

Remark 7.15 We sometimes write a^{-1} to mean the inverse of a. Be careful with this notation: if the operation is addition, then a^{-1} must be interpreted as the additive inverse, $-a$.

Definition 7.16 A non-empty set $\mathcal{G}$ on which a binary operation $\circ$ is defined is said to form a *group* with respect to $\circ$ provided that for any $a, b \in \mathcal{G}$, we have:

1. Associative law: $(a \circ b) \circ c = a \circ (b \circ c)$
2. Identity: there exists $u \in \mathcal{G}$ such that $a \circ u = u \circ a = a$
3. Inverses: For each $a \in \mathcal{G}$ there exists $a^{-1} \in \mathcal{G}$ such that $a \circ a^{-1} = a^{-1} \circ a = u$

(By definition, the set is closed under the binary operation $\circ$.)

Exercise 7.17 Which of the following form a group?

1. addition on $\mathbb{R}$?
2. multiplication on $\mathbb{R}$?
3. Example 7.4 above?

7.2 Introduction to IDEA

- **IDEA**: International Data Encryption Algorithm
- It is considered to be one of the most secure block ciphers available.
- IDEA was designed by Lai and Massey in 1991.
- The plaintext blocks are 64 bits long and the keys are 128 bits long.
- IDEA is **not** a Feistel algorithm.
- The same algorithm is used for decryption and encryption (with decryption subkeys calculated from encryption subkeys).
- It is easy to implement in both software and hardware.
- Encryption/decryption speed is similar to DES.
- IDEA is an integral part of the well-known internet security suite PGP (*Pretty Good Privacy*).

7.3 Mathematics of IDEA

- IDEA contains eight rounds of encryption (each with 14 steps), and one output round.
- Each round of encryption is based on three group operations which are carried out on n-bit blocks:
  1. $\oplus$: bitwise exclusive or;
  2. $\boxplus$: addition modulo 2^n, where the blocks are considered as their integer values;
  3. $\odot$: multiplication modulo $2^n + 1$, where all blocks are considered as their integer values except that the block containing all zeroes is considered as the integer 2^n (we require that $2^n + 1$ be prime).
- Plaintext blocks (of size 64 bits) are divided into four blocks, each of length $n = 16$ bits, and the above operations are carried out on the 16-bit blocks.

Exercise 7.18 For this example, let $n = 4$. Evaluate each of

1. $0110 \oplus 1011$
2. $0110 \boxplus 1011$
3. $0110 \odot 1011$
4. $0000 \odot 1011$

We describe the IDEA algorithm in two steps: first key generation, then the encryption algorithm.

Algorithm 7.19 Key generation for IDEA
Let the IDEA key of length 128 bits be K. Each encryption round uses 6 subkeys, and the output round uses 4 subkeys. The subkeys, denoted $K_i^{(r)}$, $1 \le r \le 8, 1 \le i \le 6$ and $K_i^{(9)}$, $1 \le i \le 4$ are obtained from K by continually dividing K into 8 16-bit subkeys, rotating K 25 bits to the left and then repeating. The resulting subkeys are shown in Figure 21.

r	K_1	K_2	K_3	K_4	K_5	K_6
1	$0-15$	$16-31$	$32-47$	$48-63$	$64-79$	$80-95$
2	$96-111$	$112-127$	$25-40$	$41-56$	$57-72$	$73-88$
3	$89-104$	$105-120$	$121-8$	$9-24$	$50-65$	$66-81$
4	$82-97$	$98-113$	$114-1$	$2-17$	$18-33$	$34-49$
5	$75-90$	$91-106$	$107-122$	$123-10$	$11-26$	$27-42$
6	$43-58$	$59-74$	$100-115$	$116-3$	$4-19$	$20-35$
7	$36-51$	$52-67$	$68-83$	$84-99$	$125-12$	$13-28$
8	$29-44$	$45-60$	$61-76$	$77-92$	$93-108$	$109-124$
9	$22-37$	$38-53$	$54-69$	$70-85$	$-$	$-$

Figure 21: Bit content of subkeys for each round of IDEA. Key bits are indexed from 0 in the original 128-bit key K

Algorithm 7.20 IDEA encryption algorithm
Let the plaintext P (of length 64 bits) be written in four blocks $P_1P_2P_3P_4$, each of length 16 bits. Denote the ciphertext at the start of each round r, $1 \le r \le 9$, by $C^{(r)} = C_1^{(r)}C_2^{(r)}C_3^{(r)}C_4^{(r)}$, with $C^{(1)} = P$. Each round r, $1 \le r \le 8$, of IDEA comprises the following steps (see Figure 22):

1. $Y_1 = C_1^{(r)} \odot K_1^{(r)} = (C_1^{(r)} \cdot K_1^{(r)}) \bmod (2^{16}+1)$
2. $Y_2 = C_2^{(r)} \boxplus K_2^{(r)} = (C_2^{(r)} + K_2^{(r)}) \bmod (2^{16})$
3. $Y_3 = C_3^{(r)} \boxplus K_3^{(r)} = (C_3^{(r)} + K_3^{(r)}) \bmod (2^{16})$
4. $Y_4 = C_4^{(r)} \odot K_4^{(r)} = (C_4^{(r)} \cdot K_4^{(r)}) \bmod (2^{16}+1)$
5. $Y_5 = Y_1 \oplus Y_3$
6. $Y_6 = Y_2 \oplus Y_4$
7. $Y_7 = Y_5 \odot K_5^{(r)} = (Y_5 \cdot K_5^{(r)}) \bmod (2^{16}+1)$
8. $Y_8 = Y_6 \boxplus Y_7 = (Y_6 + Y_7) \bmod (2^{16})$
9. $Y_9 = Y_8 \odot K_6^{(r)} = (Y_8 \cdot K_6^{(r)}) \bmod (2^{16}+1)$
10. $Y_{10} = Y_7 \boxplus Y_9 = (Y_7 + Y_9) \bmod (2^{16})$
11. $C_1^{(r+1)} = Y_1 \oplus Y_9$
12. $C_2^{(r+1)} = Y_3 \oplus Y_9$
13. $C_3^{(r+1)} = Y_2 \oplus Y_{10}$
14. $C_4^{(r+1)} = Y_4 \oplus Y_{10}$

The values Y_i, $1 \leq i \leq 10$ are called the *intermediate values*. The outputs $C_i^{(r)}$, $1 \leq i \leq 4$, are called the *intermediate ciphertexts*. To obtain the final ciphertext, follow the eight encryption rounds by the following *output* round:

1. $C_1 = C_1^{(9)} \odot K_1^{(9)} = (C_1^{(9)} \cdot K_1^{(9)}) \bmod (2^{16}+1)$
2. $C_2 = C_3^{(9)} \boxplus K_2^{(9)} = (C_3^{(9)} + K_2^{(9)}) \bmod (2^{16})$
3. $C_3 = C_2^{(9)} \boxplus K_3^{(9)} = (C_2^{(9)} + K_3^{(9)}) \bmod (2^{16})$
4. $C_4 = C_4^{(9)} \odot K_4^{(9)} = (C_4^{(9)} \cdot K_4^{(9)}) \bmod (2^{16}+1)$

Finally, the 64-bit ciphertext block is given by the four 16-bit blocks $C_1C_2C_3C_4$.

Summary of IDEA

1. Write the 64-bit plaintext in four blocks of 16 bits as
$$P = P_1P_2P_3P_4 = C_1^{(1)}C_2^{(1)}C_3^{(1)}C_4^{(1)} = C^{(1)}.$$
2. Carry out the 14 steps to obtain $C_1^{(2)}C_2^{(2)}C_3^{(2)}C_4^{(2)} = C^{(2)}$. Repeat this until you get $C_1^{(9)}C_2^{(9)}C_3^{(9)}C_4^{(9)} = C^{(9)}$.
3. Carry out the 4 steps of the output round to get the 64-bit ciphertext $C = C_1C_2C_3C_4$.

Example 7.21 Apply the first round of IDEA for the following key and plaintext (in hex).
Key: 01010303030301010123cdef00110011   Plaintext: 000f11111111000f

KEY:

hex	0	1	0	1	0	3	0	3
binary	0000	0001	0000	0001	0000	0011	0000	0011
hex	0	3	0	3	0	1	0	1
binary	0000	0011	0000	0011	0000	0001	0000	0001
hex	0	1	2	3	c	d	e	f
binary	0000	0001	0010	0011	1100	1101	1110	1111
hex	0	0	1	1	0	0	1	1
binary	0000	0000	0001	0001	0000	0000	0001	0001

Keys for round 1

$K_1^{(1)} = 0000\,0001\,0000\,0001 \quad = 257$
$K_2^{(1)} = 0000\,0011\,0000\,0011 \quad = 771$
$K_3^{(1)} = 0000\,0011\,0000\,0011 \quad = 771$
$K_4^{(1)} = 0000\,0001\,0000\,0001 \quad = 257$
$K_5^{(1)} = 0000\,0001\,0010\,0011 \quad = 291$
$K_6^{(1)} = 1100\,1101\,1110\,1111 \quad = 52719$

PLAINTEXT:

hex	0	0	0	f	1	1	1	1
binary	0000	0000	0000	1111	0001	0001	0001	0001

hex	1	1	1	1	0	0	0	f
binary	0001	0001	0001	0001	0000	0000	0000	1111

ENCRYPTION ROUND 1:

$P_1 = C_1^{(1)} = 0000\,0000\,0000\,1111 \; = 15$
$P_2 = C_2^{(1)} = 0001\,0001\,0001\,0001 \; = 4369$
$P_3 = C_3^{(1)} = 0001\,0001\,0001\,0001 \; = 4369$
$P_4 = C_4^{(1)} = 0000\,0000\,0000\,1111 \; = 15$

$Y_1 = C_1^{(1)} \odot K_1^{(1)} = (15 \cdot 257) \bmod 65537 = 3855$ so $Y_1 = 0000\,1111\,0000\,1111$

$Y_2 = C_2^{(1)} \boxplus K_2^{(1)} = (4369 + 771) \bmod 65536 = 5140$ so $Y_2 = 0001\,0100\,0001\,0100$

$Y_3 = C_3^{(1)} \boxplus K_3^{(1)} = (4369 + 771) \bmod 65536 = 5140$ so $Y_3 = 0001\,0100\,0001\,0100$

$Y_4 = C_4^{(1)} \odot K_4^{(1)} = (15 \cdot 257) \bmod 65537 = 3855$ so $Y_4 = 0000\,1111\,0000\,1111$

$Y_5 = Y_1 \oplus Y_3 = 0001\,1011\,0001\,1011 \; = 6939$

$Y_6 = Y_2 \oplus Y_4 = 0001\,1011\,0001\,1011 \; = 6939$

$Y_7 = Y_5 \odot K_5^{(1)} = (6939 \cdot 291) \bmod 65537 = 53139$ so $Y_7 = 1100\,1111\,1001\,0011$

$Y_8 = Y_6 \boxplus Y_7 = (6939 + 53139) \bmod 65536 = 60078$ so $Y_8 = 1110\,1010\,1010\,1110$

$Y_9 = Y_8 \odot K_6^{(1)} = (60078 \cdot 52719) \bmod 65537 = 45483$ so $Y_9 = 1011\,0001\,1010\,1011$

$Y_{10} = Y_7 \boxplus Y_9 = (53139 + 45483) \bmod 65536 = 33086$ so $Y_{10} = 1000\,0001\,0011\,1101$

The output of round 1 is:

$C_1^{(2)} = Y_1 \oplus Y_9 = 1011\,1110\,1010\,0100$

$C_2^{(2)} = Y_3 \oplus Y_9 = 1010\,0101\,1011\,1111$

$C_3^{(2)} = Y_2 \oplus Y_{10} = 1001\,0101\,0010\,1001$

$C_4^{(2)} = Y_4 \oplus Y_{10} = 1000\,1110\,0011\,0010$

The following figure gives a schematic diagram of IDEA. Note that for clarity, in the diagram the operation $\odot$ is shown as a multiplication sign inside a circle.

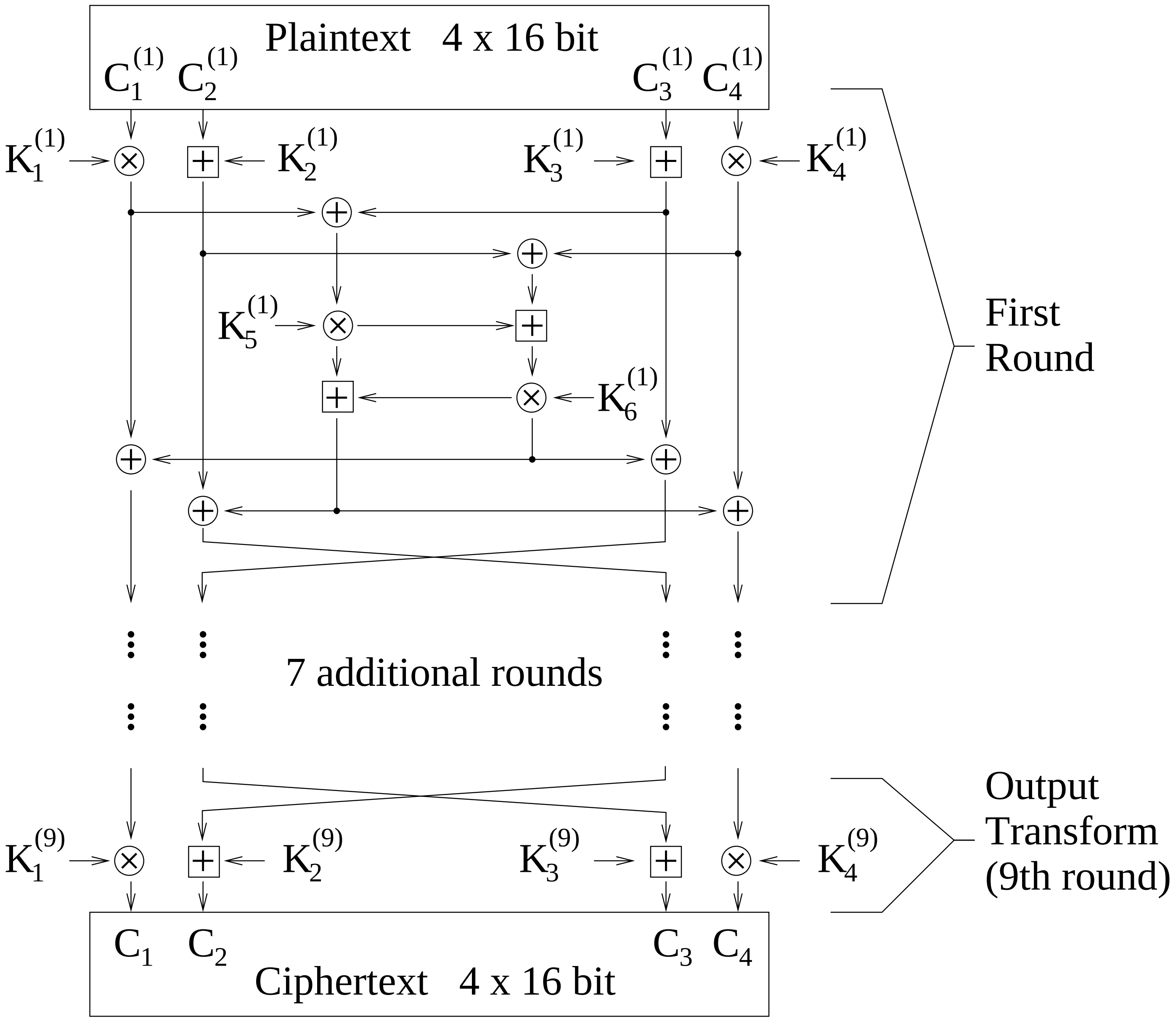
Plaintext 4 x 16 bit
$C_1^{(1)}$ $C_2^{(1)}$ $C_3^{(1)}$ $C_4^{(1)}$
$K_1^{(1)}$ $K_2^{(1)}$ $K_3^{(1)}$ $K_4^{(1)}$
$K_5^{(1)}$
$K_6^{(1)}$
First Round
7 additional rounds
Output Transform (9th round)
$K_1^{(9)}$ $K_2^{(9)}$ $K_3^{(9)}$ $K_4^{(9)}$
C_1 C_2 C_3 C_4
Ciphertext 4 x 16 bit

Figure 22: IDEA encryption

7.4 Decryption under IDEA

The decryption process is identical to encryption except that different subkeys are used. Let $DK_i^{(r)}$ denote the i^{th} decryption subkey for round r of the decryption process, $1 \leq r \leq 9$ and $1 \leq i \leq 6$ ($i \neq 5, 6$ for round 9). Then

- for $r = 1, 2, \ldots 9$, $(DK_1^{(r)},\ DK_4^{(r)}) = ((K_1^{(10-r)})^{-1},\ (K_4^{(10-r)})^{-1})$,
- for $r = 2, \ldots 8$, $(DK_2^{(r)},\ DK_3^{(r)}) = (-(K_3^{(10-r)}),\ -(K_2^{(10-r)}))$,
- for $r = 1, 9$, $(DK_2^{(r)},\ DK_3^{(r)}) = (-(K_2^{(10-r)}),\ -(K_3^{(10-r)}))$,
- for $r = 1, 2, \ldots 8$, $(DK_5^{(r)},\ DK_6^{(r)}) = (K_5^{(9-r)},\ K_6^{(9-r)})$,

where Z^{-1} means the multiplicative inverse modulo $2^{16} + 1$ of Z and $-Z$ means the additive inverse modulo 2^{16} of Z and in carrying out these two operations, the block is considered as its integer value.

7.5 Cryptanalysis of IDEA

- Brute force is not possible as the search space has size 2^{128}.
- One of the design principles of IDEA was to facilitate analysis of its strength against *differential cryptanalysis*: IDEA is considered to be immune to such attacks.
- In addition, no linear cryptanalytic attacks on IDEA have been reported and there is no known algebraic weakness in IDEA.
- The most significant cryptanalytic result is due to Daemen (1994), who discovered a large class of 251 weak keys for which the use of such a key during encryption could be detected and the key recovered. However, since there are 2^{128} possible keys, this result has no impact on the practical security of the cipher for encryption.
- IDEA is generally considered secure and both the cipher development and its theoretical basis have been openly and widely discussed.

7.6 IDEA and AES

Despite being a very secure cryptosystem, IDEA could not be considered for the Advanced Encryption System (AES) discussed in the next section, since the AES specification called for 128-bit data blocks. IDEA has data blocks of length 64 bits and the algorithm does not generalise to allow 128 bit data blocks. To see why, recall that IDEA involves evaluating multiplication modulo $2^m + 1$, where m is one quarter of the block length. For this to work, we need $2^m + 1$ to be prime. This is true for $m = 16$, but not for $m = 32$.

8 AES: The Selection Process

DES was recently replaced as the industry standard cryptosystem. In this section we discuss some information about the selection process for the replacement for DES: **The Advanced Encryption Standard**.

We'll discuss technical detail in a separate section.

8.1 Press release

Commerce Department Announces Winner
of Global Information Security Competition

FOR IMMEDIATE RELEASE: Contact: Philip Bulman Oct. 2, 2000 (301) 975-5661
G 2000-176
A worldwide competition to develop a new encryption technique that can be used to protect computerised information ended today when Secretary of Commerce Norman Y. Mineta announced the nation's proposed new Advanced Encryption Standard. Mineta named the Rijndael (pronounced Rhine-doll) data encryption formula as the winner of a three-year competition involving some of the world's leading cryptographers.

"Once final, this standard will serve as a critical computer security tool supporting the rapid growth of electronic commerce," Mineta said. "This is a very significant step toward creating a more secure digital economy. It will allow e-commerce and e-government to flourish safely, creating new opportunities for all Americans," he said.

Computer scientists at the National Institute of Standards and Technology, an agency of the Commerce Department's Technology Administration, organised the international competition in a drive to develop a strong information encryption formula to protect sensitive information in federal computer systems. Many businesses are expected to use the AES as well....

Researchers from 12 different countries worked on developing advanced encoding methods during the global competition. NIST invited the worldwide cryptographic community to "attack" the encryption formulas in an effort to break the codes. After narrowing the field down from 15 formulas to five, NIST invited cryptographers to intensify their attacks on the finalists. The agency and the world cryptographic community also evaluated the encoding formulas for factors such as security, speed and versatility.

The Rijndael developers are Belgian cryptographers Joan Daemen (pronounced Yo'-ahn Dah'-mun) of Proton World International and Vincent Rijmen (pronounced Rye'-mun) of Katholieke Universiteit Leuven. Both are highly regarded experts within the international cryptographic community.

NIST organised and managed the competition with considerable private-sector cooperation. The competing AES candidates were sophisticated mathematical formulas called algorithms. Algorithms are at the heart of computerised encryption systems, which encode everything from electronic mail to the secret personal identification numbers, or PINs, that people use with bank teller machines.

When approved, the AES will be a public algorithm designed to protect sensitive government information well into the 21st century. It will replace the aging Data Encryption Standard, which NIST adopted in 1977 as a Federal Information Processing Standard used by federal agencies to protect sensitive, unclassified information. DES and a variant called Triple DES are used widely in the private sector as well, especially in the financial services industry.

The effort to establish the AES reflects the dramatic transformation that cryptography has undergone in recent years. Just a few decades ago the science of cryptography was an esoteric endeavour employed primarily by governments to protect state and military secrets. Today, millions of Americans use cryptography, often without knowing it. Most people who use automated teller machines have used cryptography because the secret PINs required by the machines are encrypted before being sent to a computer that makes sure the number matches the card. Others use information encryption when they make a purchase over the Internet. Their credit card numbers are encrypted when they place an order. Hundreds of encryption products currently employ DES or Triple DES, and such systems have become almost ubiquitous in the financial services industry. Consequently, the selection of the AES may affect millions of consumers and businesses.

NIST requested proposals for the AES on Sept. 12, 1997, and a variety of organisations around the world responded with enthusiasm. Each of the candidate algorithms was required to support key sizes of 128, 192 and 256 bits. For a 128-bit key size, there are approximately 340×10^{36} (340 followed by 36 zeros) possible keys.

NIST evaluated the candidate algorithms and received invaluable assistance from cryptographers at computer security companies and universities around the world. Good security was the primary quality required of the winning formula, but factors such as speed and versatility across a variety of computer platforms also were considered. In other words, the algorithms must be able to run securely and efficiently on large computers, desktop computers and even small devices such as smart cards.

NIST and leading cryptographers from around the world found that all five finalist algorithms had a very high degree of security. Rijndael was selected because it had the best combination of security, performance, efficiency, implementability and flexibility.

The AES competition was organised by computer scientists in NIST's Information Technology Laboratory. A lengthy technical analysis of the AES candidates is being posted on NIST's web site today at www.nist.gov/aes.

8.2 Remarks for the press

REMARKS BY RAY KAMMER
Director, National Institute of Standards and Technology
Advanced Encryption Standard announcement
Monday, October 2, 2000

NIST started thinking about the need to replace the Data Encryption Standard, or DES, in the early 1990s. We publicly indicated that a replacement might be developed as early as the 1993 reaffirmation of DES. Serious planning at NIST began in 1996, culminating in a public call for comments on draft AES requirements in January, 1997. We held a public workshop just a few months later, which was significant because it helped to solidify the key sizes for AES at 128, 192 and 256 bits. (DES has a key size of 56 bits.) Also, almost all of the workshop participants agreed that the AES should be available on a royalty-free basis.

We published a call for candidate algorithms three years ago this month, and the response from the global cryptography community has been truly gratifying. Leading cryptographers from around the world attended our conferences in California, Rome and New York. They contributed invaluably to the selection process. We have received a great deal of help from individuals, academics, industry and government. It has not been a short process. But this was necessary in order to build trust in the encryption algorithm, because there is no simple way to determine if an algorithm is secure.

Of the original 15 candidate algorithms, five were primarily of American origin and the rest were from overseas. We received submissions from places as diverse as Costa Rica, France, Japan, Korea and Norway. NIST received comments from people in more than 40 countries during the AES public analysis period.

Each of the submitters provided a detailed description of their algorithm, and implementations in both the ANSI C and Java computer programming languages. NIST made these available to reviewers worldwide, consistent with prevailing export regulations. Each submitter also agreed in writing to make their algorithm available on a royalty free basis if it were selected for inclusion in the AES. Many decided to make their inventions free regardless. We ended up with five excellent candidates, any one of which could have provided the security we require for AES. More than 800 pages of public analysis of these candidates is posted on our Web site at www.nist.gov/aes.

Our Information Technology Laboratory formed a cross-disciplinary team to review the comments. The team has drafted a lengthy technical paper describing the selection process and the reasons for our selection. This paper will go up on our Web site later today. The performance of the candidates varied considerably, depending on whether it was implemented in hardware, software or on platforms with limited processing and memory capabilities, such as smart cards.

We have remained carefully objective. This process has been an amazing, truly global competition, reflecting the worldwide nature of information security needs. And it is a reflection of our long tradition of work in the computer security arena. In the next month or so, we will formally publish a draft of the AES standard in the Federal Register for public review and comment. We expect analysis of the encryption algorithm to continue, a process which should help to build even more public confidence in the standard.

8.3 AES: Questions and Answers

Background Information

1. **What is the Advanced Encryption Standard (AES)?**

   The Advanced Encryption Standard (AES) will be a new Federal Information Processing Standard (FIPS) Publication that will specify a cryptographic algorithm for use by U.S. Government organisations to protect sensitive (unclassified) information. NIST also anticipates that the AES will be widely used on a voluntary basis by organisations, institutions, and individuals outside of the U.S. Government - and outside of the United States - in some cases.

2. **What algorithm has been selected by NIST, and how do you pronounce it?**

   NIST has selected Rijndael as the proposed AES algorithm. The algorithm's developers have suggested the following pronunciation alternatives: "Reign Dahl" "Rain Doll" and "Rhine Dahl"

3. **Who submitted the algorithm, and where are they from?**

   The two researchers who developed and submitted Rijndael for the AES are both cryptographers from Belgium: Dr. Joan Daemen (Yo'-ahn Dah'-mun) of Proton World International and

Dr. Vincent Rijmen (Rye'-mun), a postdoctoral researcher in the Electrical Engineering Department (ESAT) of Katholieke Universiteit Leuven. Both gentlemen have been very active in the cryptographic community.

4. **Is there a document that provides details on NIST's selection for the AES?**

   NIST's ad hoc AES selection "team" has written 'Report on the Development of the Advanced Encryption Standard (AES)'. It is a comprehensive report that discusses various issues related to the AES, presents analysis and comments received during the public comment period, summarises characteristics of the five finalist AES algorithms, compares and contrasts the finalists, and presents NIST's selection of Rijndael.

   Complete AES-related information is available on the AES home page. The site includes NIST's Report on the Development of the Advanced Encryption Standard (AES); Rijndael specifications, test values, and code; all public comments, including analysis papers from the various AES conferences; and other "historical" AES information.

5. **Why is this announcement of the AES significant?**

   This announcement marks the culmination of a four-year effort involving the cooperation between the U.S. Government, and private industry and academia from around the world to develop an encryption technique that has the potential to be used by millions of people in the years to come. NIST anticipates that this algorithm will be used widely - both domestically and internationally.

6. **Is the AES now an official U.S. Government standard?**

   No. NIST has simply announced the algorithm that will be formally proposed for incorporation in a new Draft Federal Information Processing Standard (FIPS) for public review and comment. Thereafter, the standard - revised, if appropriate - will be proposed to the Secretary of Commerce for adoption as an official Government standard.

7. **When will a draft of the AES standard become available? Will the public be able to comment on the draft standard?**

   NIST intends to publish a Draft FIPS for the AES approximately one to two months after the AES announcement. At that time, a Federal Register notice will solicit public comments on the Draft FIPS for the AES for a period of 90 days.

   When the Federal Register publishes that notice, NIST will post the Draft FIPS for the AES home page, along with information on how and where to submit public comments.

8. **When will the AES become an official standard?**

   The AES will become official after the 90-day public comment period concludes, NIST makes appropriate changes to the Draft FIPS, and the Secretary of Commerce approves the FIPS. Current estimates place this sometime in the spring of 2001 (i.e., April-June).

9. **In summary, what is the projected AES development timeline?**

   A tentative timeline for the remainder of the AES development effort is as follows:

   October 2, 2000 Announcement of NIST's selection for the AES.

November 2000 Draft FIPS for the AES published for public comments. February 2001 Comment period closes.

April-June 2001 (?)AES FIPS becomes official; conformance testing available.

This timeline is subject to change, depending on the publication date of the Draft FIPS and other factors.

NIST's Selection to Propose for the AES

1. **Why did NIST select Rijndael to propose for the AES?**

   When considered together, Rijndael's combination of security, performance, efficiency, ease of implementation and flexibility make it an appropriate selection for the AES.

   Specifically, Rijndael appears to be consistently a very good performer in both hardware and software across a wide range of computing environments regardless of its use in feedback or non-feedback modes. Its key setup time is excellent, and its key agility is good. Rijndael's very low memory requirements make it very well suited for restricted-space environments, in which it also demonstrates excellent performance. Rijndael's operations are among the easiest to defend against power and timing attacks.

   Additionally, it appears that some defense can be provided against such attacks without significantly impacting Rijndael's performance. Rijndael is designed with some flexibility in terms of block and key sizes, and the algorithm can accommodate alterations in the number of rounds, although these features would require further study and are not being considered at this time. Finally, Rijndael's internal round structure appears to have good potential to benefit from instruction-level parallelism.

2. **What about the other four algorithms that were not selected?**

   In terms of security, NIST states in its report that "all five algorithms appear to have adequate security for the AES." NIST is not saying that there is anything "wrong" with any of the other four algorithms. However, when all of the analysis and comments were taken into consideration, the NIST team felt that Rijndael was the best selection for the AES.

3. **Why did NIST select only one algorithm to propose for the AES?**

   NIST considered the possibility of selecting multiple algorithms for the AES, since that was a topic of discussion during the public evaluation periods. Many arguments were made both for and against the inclusion of multiple algorithms in the standard, and NIST's AES selection team considered those comments prior to evaluating the algorithms.

   Briefly, NIST's AES selection team decided to select only one algorithm for several reasons. First, other FIPS-approved algorithms (e.g., Triple DES) offer a degree of systemic resiliency, should a problem arise with the AES. Second, multiple AES key sizes provide for increased levels of security. Third, a single algorithm AES will promote interoperability and decrease the complexity of implementations that will be built to comply with the AES specifications, hopefully promoting lower implementation costs than a multiple algorithm AES. Fourth, a single AES algorithm addresses vendors' concerns regarding potential intellectual property costs.

   NIST's report offers more details.

4. **Is NIST designating a backup algorithm?**

   No. Based on public comments and discussion, NIST's AES selection team decided not to designate a "backup" algorithm from among the other four finalists. Some vendors expressed concerns that a backup algorithm would become a de facto requirement in products (for immediate availability in the future), leading to higher implementation costs and a potential decrease in the interoperability of AES products. Also, given the uncertainty of the potential applicability of future breakthroughs in cryptanalysis, it would be unclear whether a designated backup would actually provide any resiliency for the main algorithm selection. Finally, NIST's AES selection team anticipated that other algorithms (whether specified in Government standards or not) will continue to be available in commercial products.

   NIST's report offers more details.

5. **How has the public been involved in the development of the AES?**

   From the beginning of the AES development effort, NIST has relied on the public's participation, including:

   (a) assisting NIST in the design of submission requirements and evaluation criteria (including minimum key and block size requirements and intellectual property requirements);
   (b) developing and submitting candidate algorithms;
   (c) analyzing the candidates and sharing those results with the public and NIST; and
   (d) actively participating in several international conferences.

   NIST also anticipates that the public will have very useful input on the Draft FIPS for the AES, and in the on-going analysis of Rijndael. It is expected that such analysis will be presented and published through various conferences such as CRYPTO, EUROCRYPT, ASIACRYPT, and the Fast Software Encryption Workshop (FSE).

Security and Maintenance of the AES

1. **Did NIST consider adding more rounds to Rijndael?**

   Prior to its evaluation of the five finalists, NIST's AES selection team discussed the issue of whether it should change the number of rounds for one or more of the algorithms, since that issue had been raised by the public during the recent comment period. Some of those public comments offered specific reasons for changing the number of rounds, although many did not, and there seemed to be no agreement regarding which algorithm(s) should be altered (and if so, exactly how that should be done).

   NIST's selection team recognised that changing the number of rounds would decrease the utility of the large amount of analysis that has taken place during the last two years. For some algorithms, it is not clear how the algorithm would be fully defined (e.g., the key schedule) with a different number of rounds, or how such a change would impact the security analysis. Another consideration was that none of the algorithms' submitters proposed to change the number of rounds in their algorithms, when they were given an opportunity to propose minor modifications in the summer of 1999. For these reasons, NIST's AES selection team decided it would be most appropriate to base its evaluation and selection on the five algorithms as they were originally submitted (i.e., without changing the number of rounds).

   NIST's report offers more details.

2. **Will the AES replace Triple DES and DES?**

   The AES is being developed to replace DES, but NIST anticipates that Triple DES will remain an approved algorithm (for U.S. Government use) for the foreseeable future. Single DES is being phased out of use, and is currently permitted in legacy systems, only.

   Triple DES and DES are specified in FIPS 46-3, while the AES will be specified in a completely separate FIPS. The status of the algorithms in each FIPS is handled separately by NIST.

3. **Is NIST concerned that the algorithm is of foreign origin?**

   No. The complete algorithm specification and design rationale have been available for review by NIST, NSA, and the general public for more than two years. From the beginning of the AES development effort, NIST has indicated that the involvement of the international crypto community has been necessary for the development of a high-quality standard.

4. **Approximately how big are the AES key sizes?**

   The AES will specify three key sizes: 128, 192 and 256 bits. In decimal terms, this means that there are approximately:

   - 3.4×10^{38} possible 128-bit keys;
   - 6.2×10^{57} possible 192-bit keys; and
   - 1.1×10^{77} possible 256-bit keys.

   In comparison, DES keys are 56 bits long, which means there are approximately 7.2×10^{16} possible DES keys. Thus, there are on the order of 1021 times more AES 128-bit keys than DES 56-bit keys.

5. **What is the chance that someone could use the "DES Cracker"-like hardware to crack an AES key?**

   In the late 1990s, specialised "DES Cracker" machines were built that could recover a DES key after a few hours. In other words, by trying possible key values, the hardware could determine which key was used to encrypt a message.

   Assuming that one could build a machine that could recover a DES key in a second, then it would take that machine approximately 149 thousand-billion (149 trillion) years to crack a 128-bit AES key. To put that into perspective, the universe is believed to be less than 20 billion years old.

6. **Will NIST continue to monitor the algorithm's security, and how will it handle security issues that may arise in the future?**

   Yes. As is the case with its other cryptographic algorithm standards, NIST will continue to follow developments in the cryptanalysis of Rijndael. Once the AES becomes an official standard, that standard will be formally reevaluated every five years. Maintenance activities for the standard will be developed at the appropriate time, in full consideration of the situation's particular circumstances. Should an issue arise that requires more immediate attention, NIST will act expeditiously and consider all available alternatives at that time.

7. **How long will the AES last?**

   No one can be sure how long the AES - or any other cryptographic algorithm - will remain secure. However, NIST's Data Encryption Standard (DES) was a U.S. Government standard for approximately twenty years before it was known to be "cracked" by massive parallel network computer attacks and special-purpose "DES-cracking" hardware. The AES supports significantly larger key sizes than what DES supports. Barring any attacks against AES that are faster than key exhaustion, then even with future advances in technology, AES has the potential to remain secure well beyond twenty years.

Implementation, Testing, and Use of the AES

1. **Who will be required to implement and use the AES?**

   When the AES is published as a FIPS, the algorithm will officially be identified as an approved encryption algorithm that can be used by U.S. Government organisations to protect sensitive (unclassified) information. As is currently the case, those Government organisations will be able to use other FIPS-approved algorithms in addition to, or in lieu of, the AES.

   Commercial and other non-U.S. Government organisations are invited - but not required - to adopt and implement the AES and NIST's other cryptographic standards.

2. **When will products implementing the AES be available, and will NIST test for their conformance?**

   It is anticipated that commercial products implementing Rijndael will be available shortly after the announcement. However, as indicated above, the AES itself will not become an official standard until sometime in 2001. When the AES becomes official, then NIST will have conformance testing available for products that implement Rijndael.

   That algorithm testing will be conducted under the Cryptographic Module Validation Program (CMVP), run jointly by NIST and the Communications Security Establishment (CSE) of the Government of Canada. Commercial, accredited laboratories test cryptographic implementations for conformance to NIST's standards, and if the implementations conform, then NIST and CSE issue jointly-signed validation certificates for those implementations.

3. **Are test values and reference code available for the AES?**

   Yes, test values and code provided by the Rijndael submitters are available, in order to assist implementers.

4. **Will implementations of the AES be exportable?**

   Yes, AES implementations will be exportable, and AES implementations in proprietary systems will just need a one-time review prior to export. For full details, please contact the Department of Commerce's Bureau of Export Administration (BXA), which maintains the export regulations related to cryptographic technology. More information on current export regulations is available from BXA's Information Technology Controls Division; TEL: (202) 482-0707, or FAX: (202) 501-0784.

9 AES: Technical Detail

In this section we'll consider Rijndael, the Advanced Encryption Standard. First we present some mathematical background, then we'll work through a specification of the algorithm, in a separate document.

Definition 9.1 Let F be a non-empty set with two binary operations, addition (denoted $+$) and multiplication (denoted $\times$, or by juxtaposition). Then $(F, +, \times)$ is called a *field*, if and only if

(1) $(F, +)$ is an abelian group; so:

   - closure
   - associativity
   - identity: there exists a *zero element* $0 \in F$ such that for each $a \in F$, $a + 0 = 0 + a = a$.
   - additive inverse: for each $a \in F$, there exists $-a \in F$ such that $a + (-a) = 0$.
   - commutativity

(2) Multiplication is associative: for each $a, b, c \in F$, $(ab)c = a(bc)$.

(3) Multiplication is commutative: for each $a, b \in F$, $ab = ba$.

(4) There is a unit element: there exists a non-zero element $1 \in F$ such that $a \times 1 = 1 \times a = a$, for each $a \in F$.

(5) Every non-zero element of F has a multiplicative inverse: for each $a \in F$, $a \neq 0$, there exists $a^{-1} \in F$ such that $a \times a^{-1} = a^{-1} \times a = 1$.

(6) Addition distributes over multiplication: for each $a, b, c \in F$:

   - $a(b + c) = ab + ac$
   - $(b + c)a = ba + ca$

Remark 9.2 Examples of fields are rational, real and complex numbers, with the usual definitions of addition and multiplication.

Definition 9.3 Let $K = \{0, 1\}$. For $a, b \in K$, define $a + b = 0$ if $a = b$ and $a + b = 1$ otherwise. Define $a \cdot b = 1$ if $a = b = 1$ and $a \cdot b = 0$ otherwise. Call an element of K a *scalar*.

Remark 9.4 It is easy to see that $(K, +, \cdot)$ forms a finite field, called the *Galois field of order 2*, or simply GF[2].

Remark 9.5 There is a Galois Field of order q, written GF[q], whenever q is a prime power. For AES, we are interested in GF[2^1] and GF[2^8] = GF[256].

Definition 9.6 Let $K^n = \{a_1a_2 \ldots a_n \mid a_i \in K, 1 \leq i \leq n\}$, and call an element $v \in K^n$ a *vector*. For $u, v \in K^n$, define $u + v$ componentwise, using addition defined above on K for each component. Define scalar multiplication of $v \in K^n$ by $a \in K$ in the obvious way. If v contains all 0s, then v is called the *zero word*.

Remark 9.7 It is easy to see that K^n is closed under addition and scalar multiplication.

Remark 9.8 Using addition and scalar multiplication defined above, K^n is a vector space (much more on this later).

Remark 9.9 With an appropriate definition of multiplication, we'll show that when $n = 8$, K^8 forms the field GF[256].

9.1 Polynomials over K

Definition 9.10 A *polynomial of degree n over K* is a polynomial

$$\alpha_n x^n + \alpha_{n-1}x^{n-1} + \ldots + \alpha_1 x + \alpha_0,$$

where the coefficients $\alpha_i \in K$ and $\alpha_n \neq 0$. The set of all polynomials over K is denoted $K[x]$. Polynomials over K are added and multiplied in the usual way, with a bit of care. For example, $x^k + x^k = 0$, so the degree of $f(x) + g(x)$ is not necessarily equal to the maximum of the degrees of $f(x)$ and $g(x)$.

Exercise 9.11 Let $f(x) = x^4 + x^3 + x + 1$, $g(x) = x^3 + x^2 + x$ and $h(x) = x^4 + x^2 + 1$. Find

- $f(x) + g(x)$
- $f(x) + h(x)$
- $f(x)g(x)$

Exercise 9.12 Let $f(x) = x + 1$. Find

- $(f(x))^2$
- $(f(x))^3$
- $(f(x))^4$

Remark 9.13 As just seen, $(f(x) + g(x))^2 = (f(x))^2 + (g(x))^2$.

Definition 9.14 Division Algorithm The usual long division process for polynomials over $\mathbb{R}$ also works for polynomials over K. That is, if $f(x), h(x) \in K[x]$ with $h(x) \neq 0$ then there exist unique polynomials

$$q(x), r(x) \in K[x] \text{ such that } f(x) = q(x)h(x) + r(x),$$

with $r(x) = 0$ or degree$(r(x))$ < degree$(h(x))$. As usual, $q(x)$ is called the *quotient* and $r(x)$ is called the *remainder*.

Remark 9.15 This is similar to what we already know about the quotient/remainder theorem and $\mathbb{Z}$.

Exercise 9.16 Let $f(x) = x^8 + x^7 + x^6 + x^2 + x$ and $h(x) = x^4 + x^2 + x + 1$. Find $q(x)$ and $r(x)$.

9.2 Polynomials and words

Definition 9.17 The polynomial $f(x) = \alpha_{n-1}x^{n-1} + \alpha_{n-2}x^{n-2} + \ldots + \alpha_1 x + \alpha_0$ of degree $n-1$ over K may be regarded as the word $v = \alpha_{n-1}\alpha_{n-2}\ldots\alpha_1\alpha_0$ of length n in K^n.

Remark 9.18 If we fix the length n of words we wish to consider, then every word of length n corresponds to exactly one polynomial over K of degree less than or equal to $n-1$. The word with all digits 0 is represented by the zero polynomial.

Exercise 9.19 Let $C = \{00000, 11100, 00111, 11011\}$. Represent the words of C with polynomials.

From the Division Algorithm, we have

Theorem 9.20 *Given a (fixed) polynomial $h(x)$ of degree n, by associating $f(x)$ with the remainder $r(x)$ of*

$$f(x) = q(x)h(x) + r(x),$$

each polynomial $f(x)$ corresponds to a unique polynomial of degree at most $n-1$, and hence to a unique word of length n.

Remark 9.21 Thus, given any polynomial $f(x) \in K[x]$, we can represent $f(x)$ by a unique representative of degree $< n$ from the set of residue (equivalence) classes of $K[x]$ modulo $h(x)$.

Remark 9.22 We say that $f(x)$ equals $r(x)$ modulo $h(x)$, or that $f(x) = r(x) \pmod{h(x)}$.

Exercise 9.23 Let $h(x) = x^5 + 1$. Compute the word v of length $n = 5$ corresponding to $f(x)$ modulo $h(x)$ where $f(x) = x^{11} + x^9 + x^4 + 1$.

Definition 9.24 We can avoid polynomial division when finding $r(x)$ in the following way. Noting that we are working mod $h(x)$, then if $h(x) = h_0 + h_1 x + \ldots + h_{n-1}x^{n-1} + x^n$, we can substitute for x^n in $f(x)$ using

$$x^n = h_0 + h_1 x + \ldots + h_{n-1}x^{n-1} \pmod{h(x)}.$$

Exercise 9.25 Repeat Exercise 9.23.

Definition 9.26 A polynomial $p(x) \in K[x]$ is *irreducible* if $p(x) = q(x)s(x)$ implies either $q(x)$ or $s(x)$ has degree 0 (that is, the only factors of $p(x)$ are 1 and itself).

Remark 9.27 This is similar to the definition of a prime number over $\mathbb{Z}$.

Remark 9.28 We'll see that for AES, we use the following order 8 irreducible polynomial:

$$m(x) = x^8 + x^4 + x^3 + x + 1.$$

9.3 AES Specification

Now, we'll work through the document 'A specification for Rijndael, the AES algorithm', by Brian Gladman. Note that in the rest of this section of the notes, the (sub)section numbers refer to the appropriate (sub)section of the AES specification paper.

1. Notation and Conventions

1.1 Rijndael was designed to allow an input (plaintext) block size of 128, 160, 192, 224 or 256 bits. AES is Rijndael with an input (plaintext) block size of 128 bits. In our examples, we will assume that the input (plaintext) is 128 bits, the key is 128 bits and the output (ciphertext) is 128 bits.

1.2 A byte is 8 bits. We denote a byte as $\{b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0\}$. A byte represents an element of the finite field GF(256). We will use three representations of a byte:

- bitstring, for example {01100011};
- polynomial of degree at most 7, for example $x^6 + x^5 + x + 1$;
- 2-digit hex number, for example {63}.

To distinguish between decimal representations and 2-digit hex numbers, the hex numbers will be in curly brackets, for example $\{xy\}$.

Exercise 9.29 Determine the bitstring and polynomial representations of the field elements {57} and {83}.

1.3 Rijndael operates on a state array. For AES it is a 4×4 array of bytes (usually written in hex form). (Look at the start of the example on page 21 of the paper to see how the input is read into the state array.)

1.4 The four bytes in each column of the state are sometimes thought of as a single 32-bit word.

2. Finite Field Operations

2.1 Addition of bytes (field elements) is as defined in the lecture notes as addition of polynomials over K:

- To add two bitstrings, perform an XOR operation bitwise, for example $\{01010111\} \oplus \{10000011\} = \{11010100\}$.
- To add two polynomials, add the corresponding coefficients modulo 2, for example $(x^6 + x^4 + x^2 + x + 1) \oplus (x^7 + x + 1) = x^7 + x^6 + x^4 + x^2$.
- To add two 2-digit hex numbers, convert to bitstrings and perform XOR operation, for example $\{57\} \oplus \{83\} = \{d4\}$.

2.2 Multiplication of bytes (field elements) is as defined in our previous lecture, working modulo the irreducible polynomial $m(x) = x^8 + x^4 + x^3 + x + 1$. Note that since this polynomial has degree 8 it cannot be written as a single byte, so in the paper it is sometimes written as the bitstring $1\{00011011\}$ or as the hex number $1\{1b\}$.

Exercise 9.30 Determine the product $\{57\} \cdot \{83\}$.

2.3 Multiplication can also be performed by repeated shifts of bitstrings. Note that if you start with a polynomial that does not have an x^7 term, then mutiplication by x corresponds to left-shifting the bitstring by one position.

Example $x^6 + x^4 + x^2 + x + 1$ corresponds to the bitstring $\{01010111\}$. Multiplying it by x gives $x^7 + x^5 + x^3 + x^2 + x$ which corresponds to the bitstring $\{10101110\}$.

Note that if your polynomial does have an x^7 term, then you need to reduce mod $m(x)$ after multiplying by x.

Example $\{57\} \cdot \{83\} = (x^6 + x^4 + x^2 + x + 1) \cdot (x^7 + x + 1)$ so we need to find (by shifting) $x^7\{57\}$, $x\{57\}$ and $1\{57\}$ and then XOR these three bitstrings together. See Table 1 on page 4 of the paper.

2.4 Multiplication can also be performed by adding powers of a generator. A generator of a finite field having n elements is an element g of the field for which the powers $\{g, g^2, g^3, g^4, \ldots, g^{n-1}\}$ generate all $n-1$ non-zero elements of the field. The field GF(256) can be generated by $\{03\}$. Thus every non-zero element of the field can be written in polynomial form as $(x+1)^{\text{power}}$. To multiply two field elements $a = g^\alpha$ and $b = g^\beta$, we get $a \cdot b = g^{\alpha+\beta}$, so we can use tables of powers to do multiplication. Note that here $\alpha + \beta$ is regular addition, not the $\oplus$ operation defined for field elements. Since the integer 255 is $\{ff\}$ and $g^{\{ff\}} = \{01\}$ for any generator, if $\alpha + \beta$ results in an integer larger than 255, then 255 can be subtracted from the power before referring to Table 3, that is $g^{\alpha+\beta} = g^{\{ff\}+\{xy\}} = g^{\{xy\}}$.

Exercise 9.31 Determine the product $\{57\} \cdot \{83\}$ using Table 2 and Table 3.

We can also use the tables to find multiplicative inverses of field elements. Recall that for any generator g, we have $g^{\{ff\}} = \{01\}$, or in decimal form $g^{255} = 1$. Thus we have that $g^{\{x\}} \cdot g^{\{ff\}-\{x\}} = g^{\{ff\}} = \{01\}$ so the inverse of $g^{\{x\}}$ is $g^{\{ff\}-\{x\}}$ (where $\{x\}$ is a 2-digit hex number).

Exercise 9.32 Find the inverse of $\{19\}$.

You can check your answer to Exercise 4 by multiplying the polynomials $x^4 + x^3 + 1$ and $x^5 + x^4 + x^3 + x^2 + x + 1$.

2.5 We can define polynomials whose coefficients are elements of GF(256) (rather than just elements of $\{0, 1\}$). Some parts of the cipher use these but we won't worry about them.

3. The Cipher

AES has the following general description:

Copy input into 4×4 state array.
Add initial round key to get the starting state.
Perform 9 rounds of the following:
    Substitute Bytes
    Shift Rows
    Mix Columns
    Add Round Key
Perform a final round of:
    Substitute Bytes
    Shift Rows
    Add Round Key
Output final 4×4 state array.

3.1 The Substitute Bytes routine uses an S-box. Each byte $\{xy\}$ in the current state array is replaced by the value in row x, column y of Table 5 on page 8 of the paper. The values in the S-box are computed by finding the multiplicative inverse of $\{xy\}$ and then transforming that byte $\{b_7b_6b_5b_4b_3b_2b_1b_0\}$ according to the transformation given in matrix form as equation (3.1.2) on page 8 of the paper.

Example Determine the S-box replacement for $\{19\}$. The inverse of $\{19\}$ is $\{3f\}$ (see previous example). The bitstring representation of $\{3f\}$ is $\{00111111\} = \{b_7b_6b_5b_4b_3b_2b_1b_0\}$. Thus the new bit seven is calculated as

$$b_7' = 1b_7 \oplus 1b_6 \oplus 1b_5 \oplus 1b_4 \oplus 1b_3 \oplus 0b_2 \oplus 0b_1 \oplus 0b_0 \oplus 0 = 0 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 0 \oplus 0 \oplus 0 \oplus 0 = 1.$$

Continuing in this way we see that $\{19\}$ is replaced by $\{11010100\} = \{d4\}$.
See the example on page 21 for a full Substitute Bytes routine in round one of AES.

3.2 The Shift Rows routine for AES takes the current state array, shifts the second row to the left by 1 position (wrapping around), shifts the third row to the left by 2 positions (wrapping around), and shifts the fourth row to the left by 3 positions (wrapping around).
See the example on page 21 for a full Shift Rows routine in round one of AES.

3.3 The Mix Columns routine takes each column of the current state array and transforms it according to the transformation given in matrix form as equation (3.3.1) on page 9 of the paper.
Example Determine the effect of Mix Columns on the column $\begin{bmatrix} \{d4\} & \{bf\} & \{5d\} & \{30\} \end{bmatrix}^{\mathrm{T}}$. According to equation (3.3.1) we must compute

$$\begin{bmatrix} \{02\} & \{01\} & \{01\} & \{03\} \\ \{03\} & \{02\} & \{01\} & \{01\} \\ \{01\} & \{03\} & \{02\} & \{01\} \\ \{01\} & \{01\} & \{03\} & \{02\} \end{bmatrix} \begin{bmatrix} \{30\} \\ \{5d\} \\ \{bf\} \\ \{d4\} \end{bmatrix}$$

$$= \begin{bmatrix} x(x^5+x^4)+1(x^6+x^4+x^3+x^2+1)+1(x^7+x^5+x^4+x^3+x^2+x+1)+(x+1)(x^7+x^6+x^4+x^2) \\ (x+1)(x^5+x^4)+x(x^6+x^4+x^3+x^2+1)+1(x^7+x^5+x^4+x^3+x^2+x+1)+1(x^7+x^6+x^4+x^2) \\ 1(x^5+x^4)+(x+1)(x^6+x^4+x^3+x^2+1)+x(x^7+x^5+x^4+x^3+x^2+x+1)+1(x^7+x^6+x^4+x^2) \\ 1(x^5+x^4)+1(x^6+x^4+x^3+x^2+1)+(x+1)(x^7+x^5+x^4+x^3+x^2+x+1)+x(x^7+x^6+x^4+x^2) \end{bmatrix}$$

$$= \begin{bmatrix} x^7+x^6+x^5+x^2+1 \\ x^7+1 \\ x^6+x^5+x^2+x \\ x^2 \end{bmatrix}$$

$$= \begin{bmatrix} \{e5\} \\ \{81\} \\ \{66\} \\ \{04\} \end{bmatrix}$$

Thus the new column is $\begin{bmatrix} \{04\} & \{66\} & \{81\} & \{e5\} \end{bmatrix}^{\mathrm{T}}$.

3.4 The Add Round Key takes the current state array and XORs each cell with the corresponding cell in the round key array.

Section 4 of the paper explains how to calculate the key schedule. We won't worry about it.

Here is a summary of the first round for the Example given on page 21 of the paper.

Input = 3243f6a8885a308d313198a2e0370734

Key = 2b7e151628aed2a6abf7158809cf4f3c

32	88	31	e0
43	5a	31	37
f6	30	98	07
a8	8d	a2	34

$\oplus$

2b	28	ab	09
7e	ae	f7	cf
15	d2	15	4f
16	a6	88	3c

$=$

19	a0	9a	e9
3d	f4	c6	f8
e3	e2	8d	48
be	2b	2a	08

After SubBytes:

d4	e0	b8	1e
27	bf	b4	41
11	98	5d	52
ae	f1	e5	30

After ShiftRows:

d4	e0	b8	1e
bf	b4	41	27
5d	52	11	98
30	ae	f1	e5

After MixColumns:

04	e0	48	28
66	cb	f8	06
81	19	d3	26
e5	9a	7a	4c