



哈爾濱工業大學(深圳)

HARBIN INSTITUTE OF TECHNOLOGY, SHENZHEN

# SHA1算法实验

主讲教师：蒋琳

实验教师：苏婷





- **掌握 Hash 函数的计算原理和算法特点**
- **掌握 SHA1 算法的原理**
- **实现 SHA1 算法的计算过程**



## 实验原理—哈希函数

### ◆ 压缩性

应用于任意大小的数据块，都会产生定长的输出。

### ◆ 有效性

对任意给定的消息 $m$ ，计算 $H(m)$ 是容易的。

### ◆ 安全性

抗原像：由 $H(m)$ 计算出 $m$ 不可行

抗第二原像：找到 $m'$ ，满足 $H(m) = H(m')$  计算上不可行

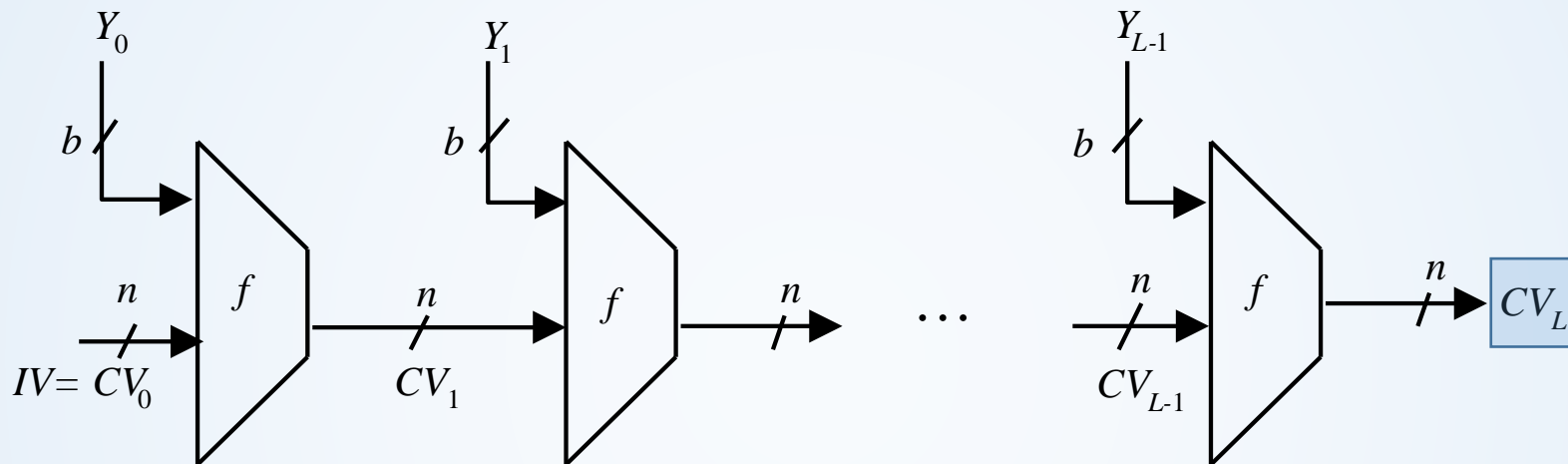
抗碰撞：给定 $H$ ，找到任意两个不同的消息 $m$ 和 $m'$ ，满足 $H(m) = H(m')$

计算上不可行



# 实验原理—哈希函数

## 迭代型哈希函数的一般结构-MD结构



$$CV_0 = IV = n$$

$$CV_i = f(CV_{i-1}, Y_{i-1}) \quad 1 \leq i \leq L$$

$$H(M) = CV_L$$

**IV** 初始向量

**CV** 链接值

**Y<sub>i</sub>** 第*i*个输入分组

**f** 压缩函数

**n** Hash值长度

**b** 输入分组长度



## SHA1算法计算过程:

## ➤ 预处理

- ◆对消息填充
- ◆初始化缓冲区

## ► 压缩过程

- ◆ 循环处理L个消息分组
- ◆ 压缩函数由4轮处理过程构成，每一轮又由20步迭代组成
- ◆ 第4轮的输出再与第1轮的输入 $CV_q$ 相加，产生 $CV_{q+1}$ ，其中加法是缓冲区5个字中的每一个字与 $CV_q$ 中相应的字模 $2^{32}$ 相加.

## ► 输出结果

- ◆ L个分组都被处理完后，最后一个 $H_{SHA1}$ 的输出即为产生的消息摘要



## 实验原理

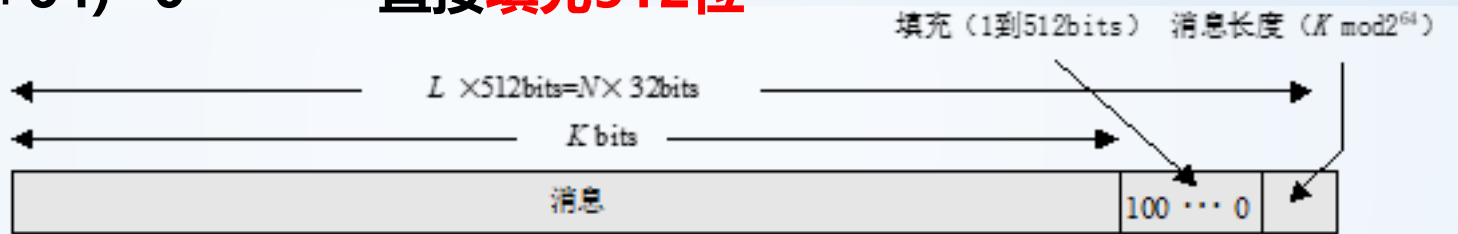
## ➤ 预处理—消息填充

**因为SHA1是512bit一个分组，所以填充后消息长度必须为512的整数倍**

- ◆ 首先在末尾处附上**64比特消息长度**
- ◆ 然后在消息原文后面填充，**第一位为1，其余为0，至少需要填充1位**，如果原始消息长度加上64bit消息长度刚好是512的整数倍，那么需要填充512位。
- ◆ 填充后的消息必须恰好为512的整数倍

### ➤ 举例说明，计算需要填充的长度

- ◆ 700位 ->  $1024-(700+64)=260$  填充260位
- ◆ 960位 ->  $1024-(960+64)=0$  直接填充512位



**Tips: 消息填充是必须的, 在任何情况下都需要消息填充**



## 实验原理

### ➤ 预处理—初始化缓冲区

算法中使用160位的缓冲区存储中间结果和最终的杂凑值。

缓冲区由5个32比特以**big-endian**方式存储数据的寄存器 A,B,C,D,E组成。

这五个缓冲区初始值为

*$A=0x67452301$*

*$B=0xEFCDAB89$*

*$C=0x98BADCFE$*

*$D=0x10325476$*

*$E= 0xC3D2E1F0$*



- 
- Diagram illustrating the iterative processing flow of SHA-256:
- Message Padding:** The message (消息) is padded to a total length of  $L \times 512 \text{ bit} = N \times 32 \text{ bit}$ . The padding includes a '1' followed by zeros (100...0). The total length is  $K \text{ bit}$ .
  - Iterative Processing:** The message is divided into blocks of 512 bits each, labeled  $Y_0, Y_1, \dots, Y_q, \dots, Y_{L-1}$ .
  - SHA Function:** Each block  $Y_i$  is processed by a SHA function (SHA) along with a 160-bit chain value (CV). The output of the SHA function is a 160-bit chain value (CV\_{i+1}).
  - Final Output:** The final output is a 160-bit summary (160bit摘要).





## 实验原理-- SHA1算法描述

$$CV_0 = IV;$$

$$CV_{q+1} = \text{SUM32}(CV_q, ABCDE_q);$$

$$MD = CV_L$$

IV：缓冲区ABCDE的初值。

ABCDE<sub>q</sub>：第q个消息分组经最后一轮处理过程处理后的输出

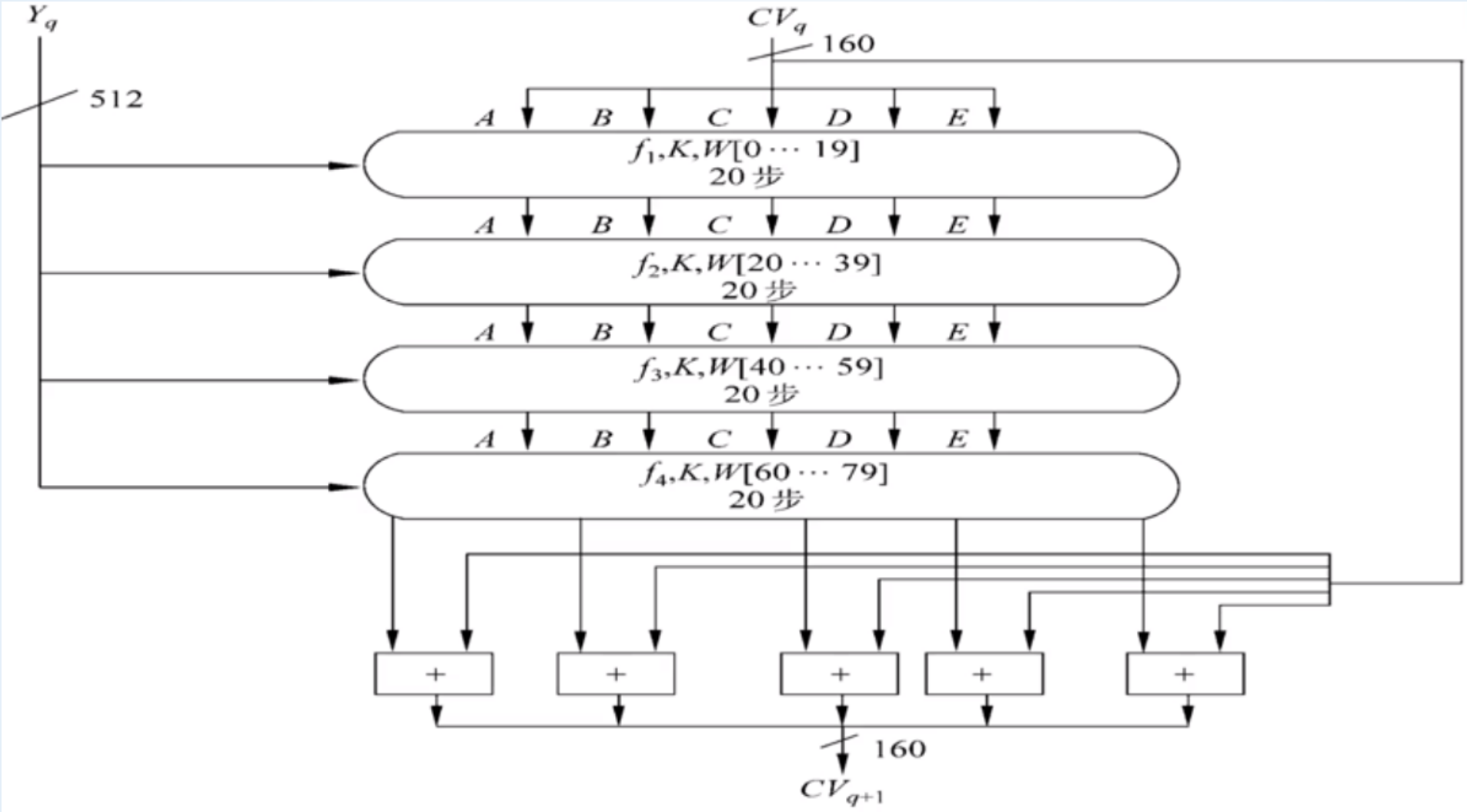
L：消息(包括填充位和长度字段)的分组数

SUM32：对应字的模 $2^{32}$ 加法

MD：最终的摘要值。



# 实验原理



SHA1 分组处理框图



## 实验原理

SHA的压缩函数由4轮处理过程组成，每轮处理过程20步迭代运算组成，每一步迭代运算的形式为

$$A, B, C, D, E \leftarrow (E + f_t(B, C, D) + CLS_5(A) + W_t + K_t), A, CLS_{30}(B), C, D$$

$A, B, C, D, E$ : 缓冲区的5个字

$t$ : 迭代的步数 ( $0 \leq t \leq 79$ )

$f_t(B, C, D)$ : 第 $t$ 步迭代使用的基本逻辑函数

$CLS_s$ : 左循环移 $s$ 位

$W_t$ : 由当前512比特长的分组导出的一个32比特长的字

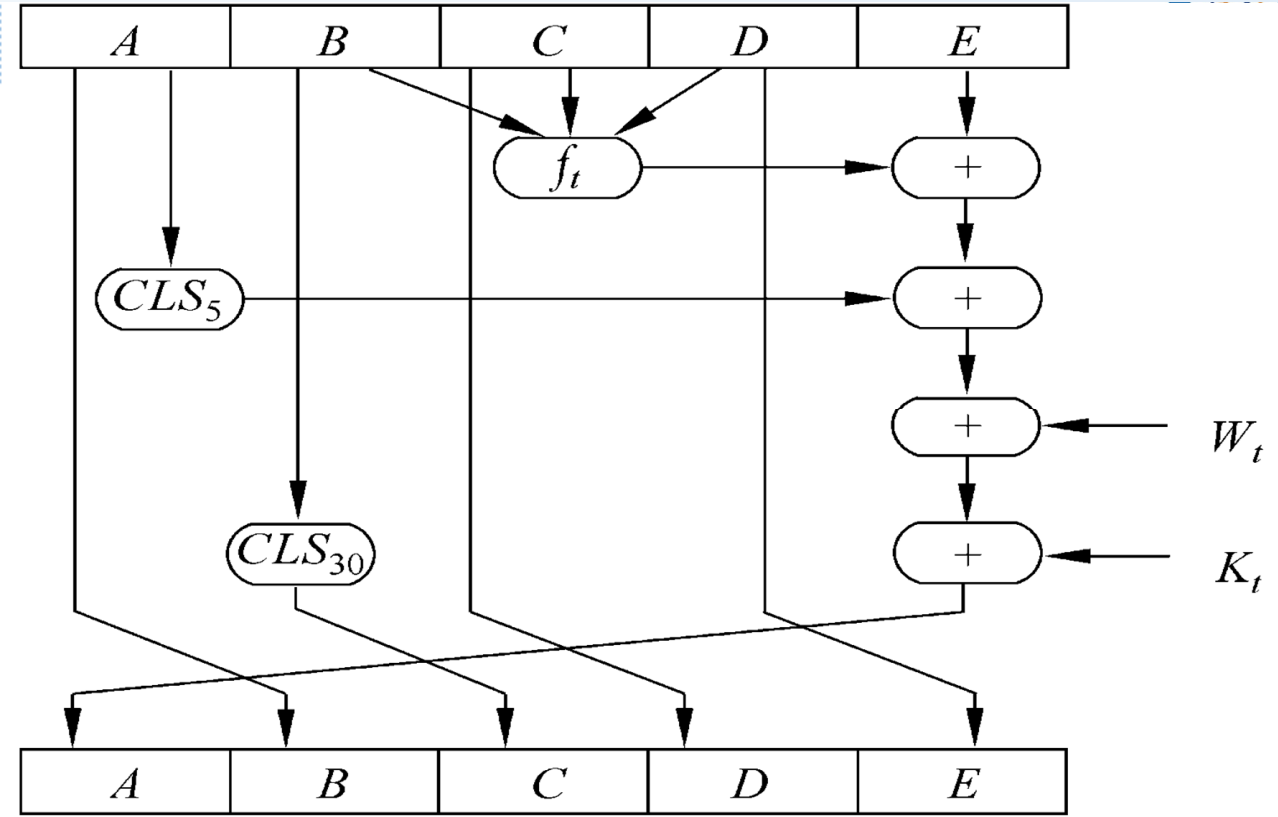
$K_t$ : 加法常量

$+$ : 模 $2^{32}$ 加法。



# 实验原理

一步运算的情况



$$A, B, C, D, E \leftarrow (E + f_t(B, C, D) + CLS_5(A) + W_t + K_t), A, CLS_{30}(B), C, D$$



# 实验原理—基本逻辑函数f

迭代的步数	函数名	定义
$0 \leq t \leq 19$	$f_1 = f_t(B, C, D)$	$(B \wedge C) \vee (\overline{B} \wedge D)$
$20 \leq t \leq 39$	$f_2 = f_t(B, C, D)$	$B \oplus C \oplus D$
$40 \leq t \leq 59$	$f_3 = f_t(B, C, D)$	$(B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$
$60 \leq t \leq 79$	$f_4 = f_t(B, C, D)$	$B \oplus C \oplus D$

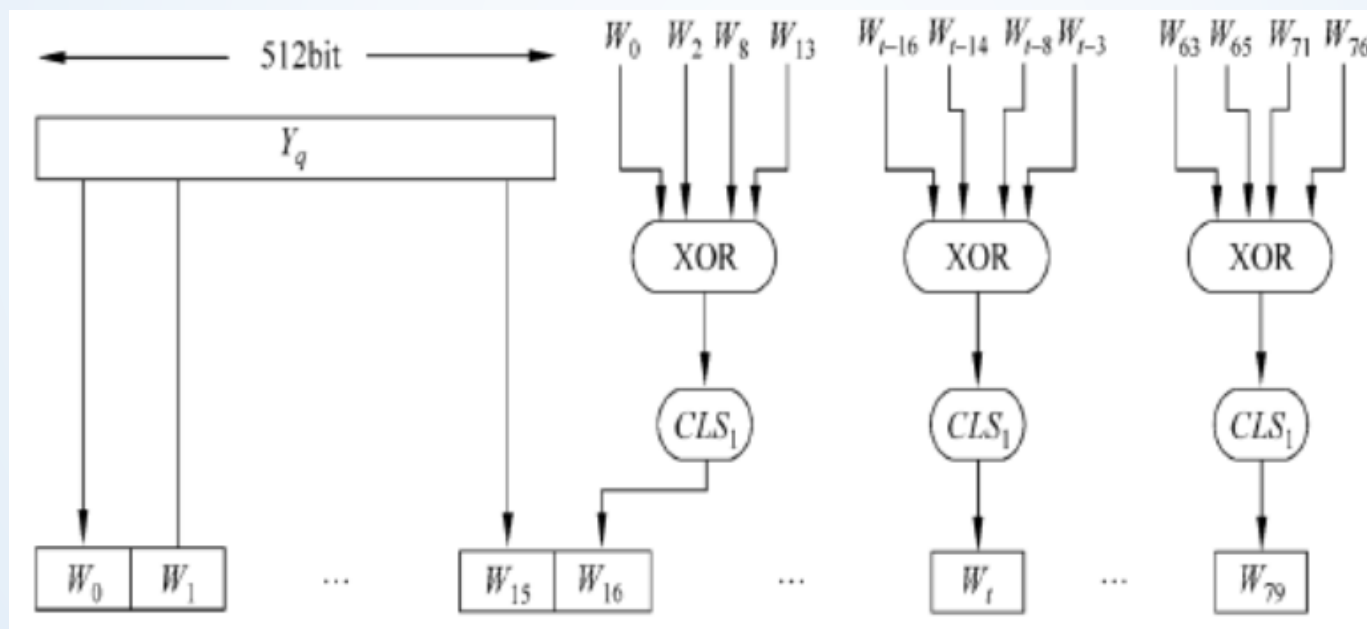
基本逻辑函数的输入为3个32比特的字，输出是一个32比特的字。表中 $\wedge, \vee, -, \oplus$ 分别是与、或、非、异或4个逻辑运算。



## 实验原理-- $W_t$

根据的输入分组（512比特长）导出 $W_0, W_1, \dots, W_{15}$ ,  
 $W_{16}, W_{17}, \dots, W_{79}$

$$W_t = CLS_1(W_{t-16} \oplus W_{t-14} \oplus W_{t-8} \oplus W_{t-3})$$





## 实验原理—常量字 $K_t$

常量字 $K_0, K_1, \dots, K_{79}$ , 以16进制形式显示如下:

$$K_t = 0x5A827999 \quad (0 \leq t \leq 19)$$

$$K_t = 0x6ED9EBA1 \quad (20 \leq t \leq 39)$$

$$K_t = 0x8F1BBCDC \quad (40 \leq t \leq 59)$$

$$K_t = 0xCA62C1D6 \quad (60 \leq t \leq 79)$$



## 实验内容

- 1、编写一个SHA1加密程序，可参考demo中的内容，计算它的 Hash 值，提交程序代码和运算结果。
- 2、【扩展，可选】对编写好的 SHA1 算法程序，测试其雪崩效应，要求给出消息文本改变前和改变后的 Hash 值，并计算出改变的位数。对比每次测试的结果并做好记录。





# 实验要求

## ➤ 截止时间

- ① 两周时间内提交 (2020-12-18 00:00)
- ② 平台链接 <http://10.249.182.83:8000/#/login>

用户名/密码: 学号/学号  
初次登录, 请修改密码!

## ➤ 提交内容

- ① 将源码和截图打成zip包上传
- ② 以学号\_姓名命名

查验加密结果是否正确

<http://www.metools.info/code/c22.html>

谢谢

