Математические основы защиты информации
Лабораторная работа №4
Функции хеширования

БГУ, ММФ, Каф. ДУиСА,
доцент Чергинец Д.Н.

Определение

Хеш-функцией (hash function) называется отображение $h : \mathcal{A}^* \to \mathcal{B}^n$ слов произвольной конечной длины над алфавитом $\mathcal{A}$ в слова фиксированной длины n над алфавитом $\mathcal{B}$. Значение функции $h(m)$ называется хешем, хеш-кодом, хеш-значением или дайджестом сообщения m. Считается, что алгоритм вычисления хеш-кода является общедоступным и полиномиальным. Сообщения m_1 и m_2 называются коллизиями (colliding), если они имеют одинаковые хеш-коды $h(m_1) = h(m_2)$.

Хеш-функции используются, например, в следующих случаях.

1. Для контроля целостности данных при их передаче или хранении.

2. Для сравнения данных.

3. Для хранения паролей.

4. Для формирования электронной цифровой подписи.

5. Для генерации случайных чисел.

6. Криптовалюты.

Алгоритм вычисления контрольной суммы

Для контроля целостности данных при их передаче или хранении можно использовать алгоритм вычисления контрольной суммы CRC (Cyclic redundancy code, CRC — циклический избыточный код). В качестве алфавита возьмем $\mathcal{A} = \mathcal{B} := \mathbb{Z}_2 = \{0, 1\}$. Каждому сообщению $m = (a_k, a_{k-1}, \ldots, a_0)$, $a_i \in \{0, 1\}$, поставим во взаимно однозначное соответствие многочлен

$$p(x) = a_k x^k + a_{k-1} x^{k-1} + \ldots + a_0$$

над полем $\mathbb{Z}_2$. Пусть $q(x)$ – примитивный (т. е. не раскладывающийся на множители) полином из кольца полиномов $\mathbb{Z}_2[x]$ степени n. Тогда хеш-функция CRC определяется соотношением

$$h(m) := p(x)\, x^n \pmod{q(x)},$$

т. е. хеш-значением является остаток $r(x)$ от деления многочлена $p(x)\, x^n$ на многочлен $q(x)$. Данные многочлены удовлетворяют соотношению

$$p(x)\, x^n = g(x)\, q(x) + r(x),$$

где степень многочлена $r(x)$ меньше n.

При отправке сообщения m к нему справа добавляют хеш-код $r = h(m)$. При получении бинарной строки m || r ("||" – операция конкатенации) соответствующий ей многочлен делят на $q(x)$ и если остаток отличен от нуля, то сообщение дошло с ошибками и его отправку необходимо повторить.

Задание 1.

Написать функцию y = CRC(m), которая вычисляет контрольную сумму $r = (r_{n-1}, r_{n-2}, \ldots, r_0)$ сообщения $m = (a_k, a_{k-1}, \ldots, a_0)$, $a_i \in \{0, 1\}$, и выдает $y = m \,||\, r = (a_k, \ldots, a_0, r_{n-1}, \ldots, r_0)$. Реализовать функцию Ver(y), которая выдает True, если многочлен, соответствующий сообщению y, делится на примитивный многочлен $q(x)$ и False в противном случае. Сравнить хеш-коды сообщений m_1 и m_2. Вычислить Ver(y_1) и Ver(y_2). При реализации данных функций удобно пользоваться следующими встроенными функциями Exponent, PolynomialReduce, PadLeft, Reverse, CoefficientList.

Вариант 1.

```
q[x] = 1 + x + x^3 + x^5 + x^7 + x^8 + x^14 + x^16 + x^22 + x^24 + x^31 + x^32;
m1 = {1, 0, 0, 0, 1};
m2 = {0, 1, 0, 0, 0, 1};
y1 = {1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0,
    1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0};
y2 = {0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0,
    0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0};
```

Вариант 2.

```
q[x] = 1 + x + x^4 + x^5 + x^9 + x^10;
m1 = {0, 0, 1, 0, 1, 1, 1, 1, 0, 0};
m2 = {0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0};
y1 = {0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1};
y2 = {1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1};
```

Вариант 3.

```
q[x] = 1 + x^2 + x^7 + x^8 + x^9 + x^11;
m1 = {0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1};
m2 = {0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1};
y1 =
  {0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1};
y2 = {1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1};
```

Вариант 4.

```
q[x] = 1 + x + x^2 + x^3 + x^11 + x^12;
m1 = {1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0};
m2 = {0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0};
y1 = {1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0,
    0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1};
y2 = {0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0,
    1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1};
```

Вариант 5.

```
q[x] = 1 + x^3 + x^4 + x^7 + x^8 + x^10 + x^14 + x^15;
m1 = {0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0};
m2 = {0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0};
y1 = {0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0,
    1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1};
y2 = {1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0,
    0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1};
```

Вариант 6.

```
q[x] = 1 + x^2 + x^15 + x^16;
m1 = {1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1,
   0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1};
m2 = {0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1,
   0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1};
y1 = {1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1,
   0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0};
y2 = {0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1,
   0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0};
```

Вариант 7.

```
q[x] = 1 + x^5 + x^12 + x^16;
m1 = {0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1,
   1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1};
m2 = {0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0,
   0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1};
y1 = {0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0,
   1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0};
y2 = {1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0,
   1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0};
```

Вариант 8.

```
q[x] = 1 + x + x^2 + x^4 + x^5 + x^7 + x^8 + x^9 + x^11 + x^15 + x^16;
m1 = {1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0,
   1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0};
m2 = {0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0,
   0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0};
y1 = {1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1,
   0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1,
   0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1};
y2 = {0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1,
   0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1,
   1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1};
```

Вариант 9.

```
q[x] = 1 + x + x^3 + x^4 + x^64;
m1 = {0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0,
    0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0};
m2 = {0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0,
    0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0};
y1 = {0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0,
    0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0,
    0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1,
    1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0};
y2 = {1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0,
    0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0,
    0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1,
    1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0};
```

Вариант 10.

```
q[x] = 1 + x + x^4 + x^7 + x^9 + x^10 + x^12 + x^13 + x^17 + x^19 +
    x^21 + x^22 + x^23 + x^24 + x^27 + x^29 + x^31 + x^32 + x^33 + x^35 + x^37 + x^38 +
    x^39 + x^40 + x^45 + x^46 + x^47 + x^52 + x^53 + x^54 + x^55 + x^57 + x^62 + x^64;
m1 = {0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1,
    1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0};
m2 = {0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0,
    1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0};
y1 = {0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0,
    0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1,
    0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
    1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0,
    0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1};
y2 = {1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0,
    0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1,
    0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
    1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0,
    0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1};
```

Вариант 11.

```
q[x] = 1 + x + x^4;
m1 = {0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,
    0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0,
    1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1};
m2 = {0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0,
    1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1,
    1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1};
y1 = {0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0,
    0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0,
    1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0};
y2 = {1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0,
    0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0,
    0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0};
```

Вариант 12.

```
q[x] = 1 + x^3 + x^5;
m1 = {0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1,
    0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1,
    1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1};
m2 = {0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1,
    1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1,
    0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1};
y1 = {0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1,
    1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0,
    0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0};
y2 = {1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1,
    1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0,
    0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0};
```

Вариант 13.

```
q[x] = 1 + x^2 + x^4 + x^5;
m1 = {1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1,
    0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0,
    1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1};
m2 = {0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0,
    1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0,
    1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1};
y1 = {1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0,
    1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1,
    1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0};
y2 = {0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0,
    1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1,
    1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0};
```

Вариант 14.

```
q[x] = 1 + x^2 + x^5;
m1 = {0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1,
    1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
    1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1};
m2 = {0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1,
    1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0,
    0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1};
y1 = {0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1,
    1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0,
    1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0};
y2 = {1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1,
    1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0,
    1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0};
```

Вариант 15.

```
q[x] = 1 + x + x^6;
m1 = {1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0,
   1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0,
   1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0};
m2 = {0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
   0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1,
   0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0};
y1 = {1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1,
   1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1,
   0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0,
   1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0};
y2 = {0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1,
   1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1,
   0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0,
   1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0};
```

Коллизии

Предположим, что нам известен хеш-код $r = h(m)$ некоторого сообщения m, причем может быть даже не известно само сообщение m, попробуем найти коллизии сообщения m, то есть такое сообщение m_2 , что $h(m) = h(m_2)$. Пусть

$$q(x) = x^n + q_{n-1}\,x^{n-1} + \ldots + q_1\,x + 1$$

примитивный многочлен из кольца многочленов $\mathbb{Z}_2[x]$ (если бы $q(0) = 0$, то тогда бы многочлен $q(x)$ делился бы на x, а он является примитивным), $r(x) = r_{n-1}\,x^{n-1} + \ldots + r_1\,x + r_0$. Подберем такой многочлен

$$g(x) = g_k\,x^k + \ldots + g_1\,x + g_0,$$

чтобы

$$g(x)\,q(x) + r(x) = f(x)\,x^n,$$

где $f(x)$ – некоторый многочлен. Очевидно, что коэффициенты $g_0, g_1, \ldots, g_{n-1}$ единственным образом последовательно находятся методом неопределенных коэффициентов из соотношений

$$g_0 = r_0,$$

$$g_1 = r_1 - g_0\,q_1,$$

$$g_2 = r_2 - g_0\,q_2 - g_1\,q_1,$$

$$\ldots$$

$$g_{n-1} = r_{n-1} - g_0\,q_{n-1} - g_1\,q_{n-2} - \ldots - g_{n-2}\,q_1,$$

а коэффициенты $g_n, g_{n+1}, \ldots, g_k$ можно выбрать произвольными. Таким образом, для каждого сообщения за полиномиальное количество

арифметических операций $1+3+5+7+\ldots+(1+2(n-1))=n^2$ можно найти сколько угодно много коллизий.

Задание 2.

Для сообщения m_1 из Задания 1, используя тот же примитивный многочлен $q(x)$, найти 5 коллизий, первый бит которых равен 1.

Криптографические хеш-функции

Для того чтобы хеш-функцию $h: X \to Y$ можно было использовать в криптографии, необходимо, чтобы не существовало полиномиальных алгоритмов решения следующих задач.

1. Восстановление прообраза. По заданному $y \in h(X)$ найти такой $x \in X$, что $h(x) = y$.

2. Нахождение второго прообраза. Для данного $x_1 \in X$ найти такой $x_2 \in X$, что $x_2 \neq x_1$, $h(x_1) = h(x_2)$.

3. Нахождение коллизий. Найти такие $x_1, x_2 \in X$, что $x_2 \neq x_1$, $h(x_1) = h(x_2)$.

Если не существует полиномиального алгоритма для решения задач 1, 2, 3, то хеш-функцию называют соответственно стойкой к вычислению прообраза, стойкой в вычислению второго прообраза, стойкой в вычислению коллизий.

Вообще говоря, еще пока не доказано существование хеш-функций стойких к вычислению прообраза и коллизиям.

Задание 3.

Какие функции хеширования встроены в пакете *Mathematica?* Какие из них являются криптографическими?

Структура Меркля-Дамгарда

Структура Меркла - Дамгарда — метод построения криптографических хеш-функций, 1979 год.

1. Приведение сообщения к блочному виду

$$m \to M = \{M_1, M_2, \ldots, M_L\},$$

M_k – блоки одинаковой длины, параметр L зависит от длины сообщения m.

2. Применение функции сжатия

$$h_k = f(h_{k-1}, M_k),$$

$k = 1, \ldots, L$, h_0 – константа.

3. $h(m) := h_k$ – результат алгоритма.

SHA-1

Преобразование сообщения к блочному виду

Алгоритм Message Padding

Вход: m – сообщение.

Выход: $M = \{M_1, M_2, \ldots, M_{L(m)}\}$, $M_i = \{M_{i1}, M_{i2}, \ldots, M_{i16}\}$, $M_{ij} \in \{0, 1\}^{32}$.

1. Каждую букву в сообщении m заменяем её битами в соответствии с кодировкой (8 битами в кодировке ASCII (IntegerDigits[ToCharacterCode[m],2,8])).

$m = \{m_1, \ldots, m_l\}$, $m_k \in \{0, 1\}^8$.

2. Добавляем справа бит, равный “1”, к полученному на первом шаге списку бит

$m := m \,||\, 1$.

3. К полученному на втором шаге списку бит добавляем справа нулевые биты пока длина списка не будет равна 448 (mod 512).

Пока Length(m) $\not\equiv$ 448 (mod 512), выполняем $m := m \,||\, 0$.

4. Осталось добавить 64 бита, чтобы полученный список был кратен 512. К полученному на третьем шаге списку добавляем справа длину бит исходного сообщения по модулю 2^{64}:

$m := m \,||$ IntegerDigits$[8 * l, 2, 64]$.

5. Разбиваем полученный список m на блоки M_i по 512 бит, а каждый блок на 16 слов по 32 бита.

Задание 4.

Реализовать функцию перевода текстового сообщения m к блочному виду.

```
MessagePadding["1"]
MessagePadding[""]
448
---
 8
m = "12345678901234567890123456789012345678901234567890123456";
StringLength[m]
длина строки
MessagePadding[m]
```

```
{{{0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0}}}
```

```
{{{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}}}
```

```
56

56
```

```
{{{0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0},
  {0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0},
  {0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0},
  {0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0},
  {0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0},
  {0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0},
  {0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0},
  {0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0},
  {0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0},
  {0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0},
  {0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0},
  {0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0},
  {0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0},
  {0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0},
  {1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}},
 {{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
  {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
   0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0}}}
```

Согласно структуре Меркля-Дамгарда функция SHA-1 работает следующим образом.

Задаем начальные условия:

```
H0 = IntegerDigits[FromDigits["67452301", 16], 2, 32];
H1 = IntegerDigits[FromDigits["EFCDAB89", 16], 2, 32];
H2 = IntegerDigits[FromDigits["98BADCFE", 16], 2, 32];
H3 = IntegerDigits[FromDigits["10325476", 16], 2, 32];
H4 = IntegerDigits[FromDigits["C3D2E1F0", 16], 2, 32];
IV = {H0, H1, H2, H3, H4}
SHA1[m_String] := Block[{H = IV, M},
  M = MessagePadding@m;
  Do[H = Compress1Block[M[[i]], H], {i, Length[M]}];
  FromDigits[H // Flatten, 2]
 ]
```

```
{{0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1},
 {1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1},
 {1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0},
 {0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0},
 {1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0}}
```

Определим функцию сжатия Compress1Block

В данной функции исользуется сложение по модулю 2^{32}, его можно задать следующей формулой:

```
a_ ± b_ := IntegerDigits[FromDigits[a, 2] + FromDigits[b, 2], 2, 32]
```

SHA-1 использует последовательность логических функций f_1, f_2,....., f_{80}. Каждая функция определена следующим образом:

$$f_i(b,c,d)=\begin{cases}(b\wedge c)\oplus(\neg b\wedge d), & \text{при } 1\le i\le 20;\\ b\oplus c\oplus d, & \text{при } 21\le i\le 40;\\ (b\wedge c)\oplus(b\wedge d)\oplus(c\wedge d), & \text{при } 41\le i\le 60;\\ b\oplus c\oplus d, & \text{при } 61\le i\le 80.\end{cases}$$

```
ff[i_, b_, c_, d_] := Which[(i < 21), BitOr[BitAnd[b, c], BitAnd[BitNot@b, d]],
   ((i > 20) && (i < 41)), BitXor[b, c, d],
   (i > 40) && (i < 61), BitOr[BitAnd[b, c], BitAnd[b, d], BitAnd[c, d]],
   (i > 60) && (i < 81), BitXor[b, c, d]];
```

SHA-1 использует последовательность 80-ти постоянных 32-битных слов, K_1, K_1 ,... , K_{80}, заданных следующим образом:

$$K_i=\begin{cases} 5\,A827999, & \text{при } 1\le i\le 20;\\ 6\,ED9EBA1, & \text{при } 21\le i\le 40;\\ 8\,F1BBCDC, & \text{при } 41\le i\le 60;\\ CA62C1D6, & \text{при } 61\le i\le 80. \end{cases}$$

```
K19 = IntegerDigits[FromDigits["5A827999", 16], 2, 32];
      цифры целого ч··· число по ряду цифр
K39 = IntegerDigits[FromDigits["6ED9EBA1", 16], 2, 32];
      цифры целого ч··· число по ряду цифр
K59 = IntegerDigits[FromDigits["8F1BBCDC", 16], 2, 32];
      цифры целого ч··· число по ряду цифр
K79 = IntegerDigits[FromDigits["CA62C1D6", 16], 2, 32];
      цифры целого ч··· число по ряду цифр
Kj_ := Which[(j < 21), K19,
      условный оператор с множественными ветвями
   ((j > 20) && (j < 41)), K39,
   (j > 40) && (j < 61), K59,
   (j > 60) && (j < 81), K79];
```

Алгоритм Compress1Block

Вход: $M = \{M_1, \ldots, M_{16}\}, M_k \in \{0, 1\}^{32}$,

$H = \{H_1, H_2, H_3, H_4, H_5\},\ H_k \in \{0, 1\}^{32}$.

Выход: $H = \{H_1, H_2, H_3, H_4, H_5\},\ H_k \in \{0, 1\}^{32}$.

1. Задаем начальные данные рекуррентной формулы

$\{R_0, R_{-1}, R_{-2}, R_{-3}, R_{-4}\} = H$.

2. Дополняем блок M до 80 слов по рекуррентной формуле:

$M_k = \text{RotateLeft}[\text{BitXor}[M_{k-3}, M_{k-8}, M_{k-14}, M_{k-16}], 1], 17 \le k \le 80$.

3. Для i от 1 до 80 выполняем

$R_i = \text{RotateLeft}[R_{i-1}, 5] \pm \text{ff}[i, R_{i-2}, R_{i-3}, R_{i-4}] \pm R_{i-5} \pm K_i \pm M[\![i]\!]$,

$R_{i-2} = \text{RotateLeft}[R_{i-2}, 30]$.

4. Выдаем результат

$H=\{R_{80} \pm H[\![1]\!], R_{79} \pm H[\![2]\!], R_{78} \pm H[\![3]\!], R_{77} \pm H[\![4]\!], R_{76} \pm H[\![5]\!]\}$.

Задание 5.

Реализовать функцию Compress1Block.

R_i для m = "1" на четвертом шаге алгоритма имеют вид

```
SHA1["1"]
```

```
R[-4]={1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0}
R[-3]={0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0}
R[-2]={1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0}
R[-1]={0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0}
R[0]={0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0}
R[1]={1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0}
```

R[2]={1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1}

R[3]={0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0}

R[4]={0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0}

R[5]={0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0}

R[6]={1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0}

R[7]={1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1}

R[8]={0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0}

R[9]={1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0}

R[10]={0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0}

R[11]={1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0}

R[12]={1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0}

R[13]={0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0}

R[14]={0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0}

R[15]={0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1}

R[16]={0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1}

R[17]={1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0}

R[18]={1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0}

R[19]={1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0}

R[20]={0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1}

R[21]={1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0}

R[22]={1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1}

R[23]={1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0}

R[24]={1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1}

R[25]={0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1}

R[26]={0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0}

R[27]={1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1}

R[28]={0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1}

R[29]={1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0}

R[30]={0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1}

R[31]={0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0}

R[32]={0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0}

R[33]={1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0}

R[34]={0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0}

R[35]={1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1}

R[36]={0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0}

R[37]={0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0}

R[38]={1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1}

R[39]={1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0}

R[40]={1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0}

R[41] = {1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1}

R[42] = {0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1}

R[43] = {0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1}

R[44] = {0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0}

R[45] = {0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0}

R[46] = {0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0}

R[47] = {0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1}

R[48] = {1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0}

R[49] = {1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0}

R[50] = {0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1}

R[51] = {0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0}

R[52] = {1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1}

R[53] = {0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0}

R[54] = {0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0}

R[55] = {1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0}

R[56] = {0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1}

R[57] = {1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0}

R[58] = {0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0}

R[59] = {1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0}

R[60] = {1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1}

R[61] = {0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0}

R[62] = {0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1}

R[63] = {0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0}

R[64] = {0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1}

R[65] = {1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1}

R[66] = {1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1}

R[67] = {0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1}

R[68] = {0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1}

R[69] = {0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0}

R[70] = {1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0}

R[71] = {1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1}

R[72] = {0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0}

R[73] = {0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0}

R[74] = {0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1}

R[75] = {0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1}

R[76] = {0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1}

R[77] = {1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0}

R[78] = {1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0}

R[79] = {1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1}

R[80] = {1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0}

304 942 582 444 936 629 325 699 363 757 435 820 077 590 259 883

BaseForm[SHA1[m], 16]
запись в системе с основанием
BaseForm[Hash[m, "SHA"], 16]
запись в ⋯ хэш

$a84666b66e843a4146088fb46aabaa998b4c2b1_{16}$

$a84666b66e843a4146088fb46aabaa998b4c2b1_{16}$

Задание 6.

Сравнить результаты вычисления

SHA1["±α"]
SHA1["α±"]

1 232 282 100 786 404 337 666 601 077 793 289 415 869 680 301 191

1 232 282 100 786 404 337 666 601 077 793 289 415 869 680 301 191

In[ ]:= **Hash["±α", "SHA"]**
хэш
Hash["α±", "SHA"]
хэш

Out[ ]= 339 642 187 376 354 215 367 232 373 880 271 214 020 785 718 835

Out[ ]= 184 281 848 769 261 220 973 398 056 760 638 991 223 797 287 413

Какая особенность сообщений повлияла на ответ?

Задание 7.

Используя ToCharacterCode[m,"UTF8"], написать функцию SHA1U[m_String], моделирующую встроенную функцию Hash[m,"SHA"] в версии Wolfram Mathematica 12 и выше.

In[ ]:= **BaseForm[SHA1["±α"], 16]**
запись в системе с основанием
BaseForm[SHA1U["±α"], 16]
запись в системе с основанием
BaseForm[Hash["±α", "SHA"], 16]
запись в ⋯ хэш

Out[ ]//BaseForm=
$d7d970c43c0e96561039d3ae632c53d17c8bb487_{16}$

Out[ ]//BaseForm=
$3b7e1519679219ef610f9180bb567fce0de9c633_{16}$

Out[ ]//BaseForm=
$3b7e1519679219ef610f9180bb567fce0de9c633_{16}$

Криптоанализ SHA-1

Задание 8.

Блок дополняется по формуле

M_k = RotateLeft[BitXor[M_{k-3}, M_{k-8}, M_{k-14}, M_{k-16}], 1], k = 17, ..., 80.

Можно ли при k=33,...,80 дополнять его по формуле

M_k = RotateLeft[BitXor[M_{k-6}, M_{k-16}, M_{k-28}, M_{k-32}], 2]?

SHAttered

23 февраля 2017 года специалисты из Google и CWI объявили о практическом взломе алгоритма, опубликовав 2 PDF-файла с одинаковой контрольной суммой SHA-1. Это потребовало перебора $9 * 10^{18}$ вариантов. Они нашли следующие коллизии:

```
In[•]:= prefix1 =
  "255044462d312e330a25e2e3cfd30a0a0a312030206f626a0a3c3c2f57696474682032203020522f48⋱
    656967687420332030205222f547970652034203020522f537562747970652035203020522f466⋱
    96c7465722036203020522f436f6c6f7253706163652037203020522f4c656e677468203820 30⋱
    20522f42697473506572436f6d706f6e656e7420383e3e0a73747265616d0affd8fffe0024534⋱
    8412d3120697320646561642121212121852fec092339759c39b1a1c63c4c97e1fffe017346dc⋱
    9166b67e118f029ab621b2560ff9ca67cca8c7f85ba84c79030c2b3de218f86db3a90901d5df4⋱
    5c14f26fedfb3dc38e96ac22fe7bd728f0e45bce046d23c570feb141398bb552ef5a0a82be331⋱
    fea48037b8b5d71f0e332edf93ac3500eb4ddc0decc1a864790c782c76215660dd309791d06bd⋱
    0af3f98cda4bc4629b1";
prefix2 =
  "255044462d312e330a25e2e3cfd30a0a0a312030206f626a0a3c3c2f57696474682032203020522f48⋱
    656967687420332030205222f547970652034203020522f537562747970652035203020522f466⋱
    96c7465722036203020522f436f6c6f7253706163652037203020522f4c656e677468203820 30⋱
    20522f42697473506572436f6d706f6e656e7420383e3e0a73747265616d0affd8fffe0024534⋱
    8412d3120697320646561642121212121852fec092339759c39b1a1c63c4c97e1fffe017f46dc⋱
    93a6b67e013b029aaa1db2560b45ca67d688c7f84b8c4c791fe02b3df614f86db1690901c56b4⋱
    5c1530afedfb76038e972722fe7ad728f0e4904e046c230570fe9d41398abe12ef5bc942be335⋱
    42a4802d98b5d70f2a332ec37fac3514e74ddc0f2cc1a874cd0c78305a21566461309789606bd⋱
    0bf3f98cda8044629a1";

In[•]:= l = StringLength[prefix1]
            длина строки
l = StringLength[prefix2]
            длина строки

Out[•]= 640

Out[•]= 640
```

In[•]:=
```
FromDigits[prefix1, 16]
```
число по ряду цифр
```
FromCharacterCode[FromDigits[#, 2]] & /@
```
символ по его коду число по ряду цифр
```
  (Partition[IntegerDigits[FromDigits[prefix2, 16], 2, l 4], 8])
```
разбиение··· цифры целого ч··· число по ряду цифр
```
s1 = StringJoin[%]
```
соединить строки

Out[•]=
```
63 156 115 232 790 738 757 732 099 090 329 369 829 865 901 543 014 549 474 989 589 938 833 604 053 232 ↘
 010 005 346 924 708 771 276 742 649 051 869 781 325 325 980 059 674 613 141 301 092 737 750 746 368 ↘
 664 597 338 902 241 284 563 057 591 529 514 054 279 024 356 573 336 804 348 396 150 626 565 341 853 ↘
 388 765 058 107 373 212 332 778 048 362 587 625 858 091 191 280 396 695 812 339 795 202 534 135 861 ↘
 513 623 478 920 478 375 054 908 809 183 125 681 714 548 383 255 537 997 868 320 359 914 877 759 583 ↘
 363 053 575 781 913 190 817 504 568 118 622 329 130 450 396 707 489 912 179 815 141 959 179 644 495 ↘
 100 873 191 795 791 375 342 072 096 426 625 081 960 602 161 967 659 439 519 133 981 120 676 440 123 ↘
 101 170 795 515 929 186 917 519 990 566 990 720 515 062 773 348 299 506 753 944 547 926 834 438 605 ↘
 265 139 197 517 027 997 310 920 986 323 266 449 858 205 266 949 510 063 809 899 350 885 890 404 341 ↘
 431 045 504 636 502 796 468 899 322 998 154 469 858 810 115 030 588 108 922 330 870 966 177 395 348 ↘
 589 136 357 993 359 793
```

Out[•]=
```
{%, P, D, F, –, 1, ., 3,
, %, â, ã, Ï, Ó,
,
,
, 1,  , 0,  , o, b, j,
, <, <, /, W, i, d, t, h,  , 2,  , 0,  , R, /, H, e, i, g, h, t,  , 3,  , 0,  , R, /, T, y,
 p, e,  , 4,  , 0,  , R, /, S, u, b, t, y, p, e,  , 5,  , 0,  , R, /, F, i, l, t, e, r,
  , 6,  , 0,  , R, /, C, o, l, o, r, S, p, a, c, e,  , 7,  , 0,  , R, /, L, e, n, g, t,
 h,  , 8,  , 0,  , R, /, B, i, t, s, P, e, r, C, o, m, p, o, n, e, n, t,  , 8, >, >,
, s, t, r, e, a, m,
, ÿ, Ø, ÿ, þ, , $, S, H, A, –, 1,  , i, s,  , d, e, a, d, !, !, !, !, !, , /, ì,      ,
 ⌗, 9, u, , 9, ±, ¡, Æ, <, L, , á, ÿ, þ, ,  , F, Ü, , ¦, ¶, ~, , ;,  , ,  ª, ,  ², V,  ,
 E, Ê, g, Ö, , Ç, ø, K, , L, y, , à, +, =, ö, ▨, ø, m, ±, i,      , , Å, k, E, Á, S,
, þ, ß, ·, `, 8, é, r, r, /, ç, , r, ,  , I,  , à, F, Â, 0, W,  , é, Ô, ▨, , «, á, ., õ, ¼,
  , +, ã, 5, B, ¤, , –, , µ, ×,  , *, 3, ., Ã,  , ¬, 5, ▨, ç, M, Ü,  , ,, Á, …, t, Í,
, x, 0, Z, !, V, d, a, 0, , , `, k, Ð, ¿, ?, , Í, …,  , F, ), ¡}
```

Out[•]=
```
%PDF–1.3
%âãÏÓ


1 0 obj
<</Width 2 0 R/Height 3 0 R/Type 4 0 R/Subtype 5 0
  R/Filter 6 0 R/ColorSpace 7 0 R/Length 8 0 R/BitsPerComponent 8>>
stream
ÿØÿþ     –1 is dead!!!!!/ì     ⌗9u9±¡Æ<Láÿþ FÜ¦¶~; ª²V EÊgÖÇøKLyà+=ö▨øm±i     ÅkEÁS
þß·`8érr/çr I àFÂ0W éÔ▨«á.õ¼+ã5B¤–µ× *3.Ã ¬5▨çMÜ ,Á…tÍ
x0Z!Vda0`kÐ¿?Í… F)¡
```

In[•]:=
```
FromCharacterCode[FromDigits[#, 2]] & /@
```
символ по его коду число по ряду цифр
```
  (Partition[IntegerDigits[FromDigits[prefix1, 16], 2, l 4], 8])
```
разбиение… цифры целого ч… число по ряду цифр
```
s2 = StringJoin[%]
```
соединить строки

Out[•]=
```
{%, P, D, F, −, 1, ., 3,
, %, â, ã, Ï, Ó,
,
,
, 1,  , 0,  , o, b, j,
, <, <, /, W, i, d, t, h,  , 2,  , 0,  , R, /, H, e, i, g, h, t,  , 3,  , 0,  , R, /, T, y,
 p, e,  , 4,  , 0,  , R, /, S, u, b, t, y, p, e,  , 5,  , 0,  , R, /, F, i, l, t, e, r,
  , 6,  , 0,  , R, /, C, o, l, o, r, S, p, a, c, e,  , 7,  , 0,  , R, /, L, e, n, g, t,
 h,  , 8,  , 0,  , R, /, B, i, t, s, P, e, r, C, o, m, p, o, n, e, n, t,  , 8, >, >,
, s, t, r, e, a, m,
, ÿ, Ø, ÿ, þ, , $, S, H, A, −, 1,  , i, s,  , d, e, a, d, !, !,
 !, !, !, , /, ì,  , ⌗, 9, u, , 9, ±, ¡, Æ, <, L, , á, ÿ, þ, , s, F, Ü,
 , f, ¶, ~, ▨, ,  , , ¶, !, ², V,  , ù, Ê, g, Ì, …, Ç, ø, [, …, L, y,  ,
, +, =, â, ▨, ø, m, ³, ©,  , , Õ, ß, E, Á, O, &, þ, ß, ³, Ü, 8, é,
 j, Â, /, ç, ¼, r, ,  , E, ¼, à, F, Ò, <, W,  , ë, ▨, ▨, , », U, ., õ,
  , …, +, ã, 1, þ, ¤, , 7, ¸, µ, ×, ,  , 3, ., ß, , ¬, 5, , ë, M, Ü,
, ì, Á, …, d, y,
, x, ,, v, !, V, `, Ý, 0, , , Ð, k, Ð, ¯, ?, , Í, ¤, ¼, F, ), ±}
```

Out[•]=
```
%PDF−1.3
%âãÏÓ


1 0 obj
<</Width 2 0 R/Height 3 0 R/Type 4 0 R/Subtype 5 0
  R/Filter 6 0 R/ColorSpace 7 0 R/Length 8 0 R/BitsPerComponent 8>>
stream
ÿØÿþ    −1 is dead!!!!!/ì    ⌗9u9±¡Æ<LáÿþsFÜf¶~▨ ¶!²V ùÊgÌ…Çø[…Ly
+=â▨øm³©    ÕßEÁO&þß³Ü8éjÂ/ç½r E¼àFÒ<W ë▨▨»U.õ …+ã1þ¤7¸µ× 3.ß¬5
ìÁ…dy
x,v!V`Ý0ÐkÐ¯?Í¤¼F)±
```

In[•]:=
```
SHA1[s1]
SHA1[s2]
```

Out[•]= 1 422 552 417 918 142 094 400 947 145 252 742 830 701 779 435 164

Out[•]= 1 422 552 417 918 142 094 400 947 145 252 742 830 701 779 435 164

In[•]:=
```
Hash[s1, "SHA"]
```
хэш
```
Hash[s2, "SHA"]
```
хэш

Out[•]= 1 248 638 279 647 414 391 929 208 848 037 997 343 635 436 112 371

Out[•]= 415 435 019 684 222 752 580 514 425 425 863 961 178 638 657 009

Благодаря данной коллизии можно конструировать другие коллизии

```
In[•]:= s = RandomInteger[{1, 255}, 20]
          случайное целое число
Out[•]= {52, 233, 164, 26, 179, 124, 191, 95, 209, 17, 17, 157, 122, 171, 177, 4, 51, 139, 181, 115}

In[•]:= SHA1[StringJoin[s1, FromCharacterCode[s]]]
          соединить строки  символ по его коду
        SHA1[StringJoin[s2, FromCharacterCode[s]]]
          соединить строки  символ по его коду
Out[•]= 1 385 187 830 085 242 478 970 286 572 348 711 658 706 029 701 895

Out[•]= 1 385 187 830 085 242 478 970 286 572 348 711 658 706 029 701 895
```

Данная атака называется Chosen-prefix collision attack. Эта атака возможна из-за Структуры Меркля-Дамгарда. То есть когда первые биты сообщения влияют только на первый блок.
При помощи найденных коллизий prefix1, prefix2 можно легко конструировать различные pdf-файлы с одинаковым хешем.

Функция SHA-3 разработа по принципу “губки” и на нее данная атака не действует.