Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина «Методы защиты информации»

**ОТЧЕТ**

к лабораторной работе

на тему:

**«Хэш функции»**

|  |  |
| --- | --- |
|  | Выполнил:  Слуцкий Никита Сергеевич,  студент группы 053505 |
|  | Проверил: Лещенко Евгений Александрович, ассистент каф. Информатики |

Минск 2023

СОДЕРЖАНИЕ

[Введение 3](#_Toc148956685)

[1 Краткие теоретические сведения 4](#_Toc148956686)

[1.1 SHA-1 4](#_Toc148956687)

[1.2 ГОСТ 34.11 5](#_Toc148956688)

[2 Ход выполнения работы 5](#_Toc148956689)

[Заключение 6](#_Toc148956690)

[Приложение А 7](#_Toc148956691)

# Введение

Целью данной лабораторной работы является реализовать программное средство хэширования в соответствии с выданным вариантом задания. В связи с нечётным номером в журнале необходимо реализовать программное средство хэширования по стандарту ГОСТ 34.11 и SHA-1. А также оформить отчёт в соответствии со стандартом предприятия БГУИР.

# 1 Краткие теоретические сведения

## 1.1 SHA-1

Алгоритм SHA-1 (Secure Hash Algorithm) предназначен для вычисления дайджеста документа, файла или сообщения. Когда входное сообщение имеет произвольную длину < 264 бит, программа SHA-1 выдает 160-битовый код, называемый дайджестом сообщения. Дайджест сообщения может стать, например, исходной информацией для алгоритма электронной подписи, которая формируется для верификации сообщения. Подпись сообщения улучшает эффективность процесса, так как дайджест обычно имеет размер много меньше размера сообщения. Тот же самый алгоритм используется при верификации цифровой подписи. Любая модификация сообщения при транспортировке с крайне высокой вероятностью приведет к тому, что дайджест изменится и подпись не будет верифицирована.

Позднее были разработаны более эффективные версии SHA-2 (2001, 256-512 бит) и SHA-3 (Йоан Даймен, 2013, 256-512 бит).

Алгоритм SHA-1 называется безопасным, так как на его основе невозможно вычислить сообщение, которому соответствует дайджест, или найти другое сообщение, которое может соответствовать полученному дайджесту. Любое изменение сообщения при передаче приведет с большой вероятностью к существенному изменению дайджеста, что исключит успешную сверку цифровой подписи. На рисунке 1 представлена схема вычисления дайджеста сообщения с помощью SHA-1.

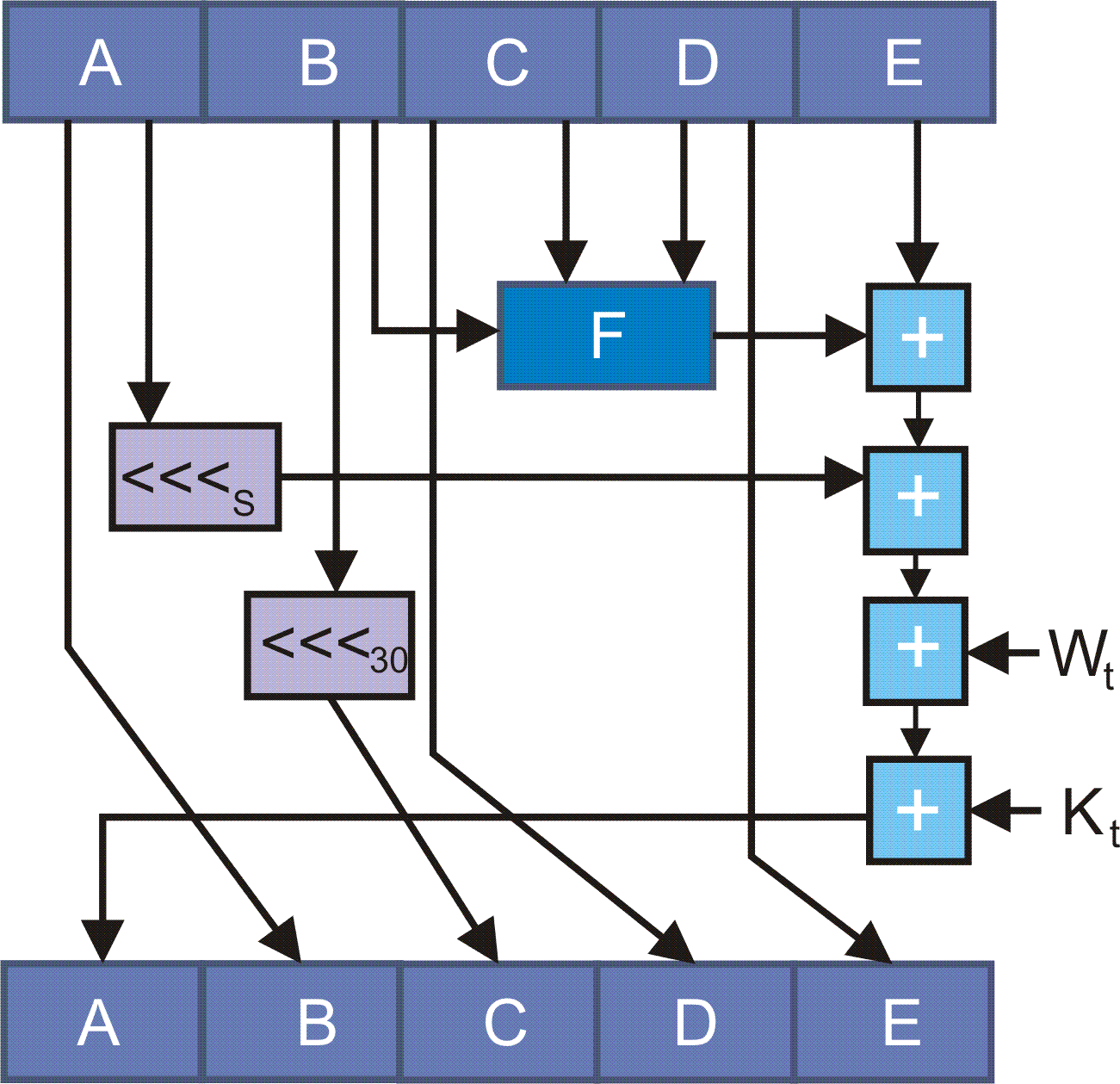


Рисунок 1 – Алгоритм вычисления дайджеста

## 1.2 ГОСТ 34.11

Настоящий стандарт определяет алгоритм и процедуру вычисления хэш-функции для любой последовательности двоичных символов, которые применяются в криптографических методах обработки и защиты информации, в том числе для реализации процедур обеспечения целостности, аутентичности, электронной цифровой подписи (ЭЦП) при передаче, обработке и хранении информации в автоматизированных системах.

Определенная в настоящем стандарте функция хэширования используется при реализации систем электронной цифровой подписи на базе ассиметричного криптографического алгоритма по ГОСТ 34.11 Стандарт рекомендуется использовать при создании, эксплуатации и модернизации систем обработки информации различного назначения

# 2 Ход выполнения работы

В рамках лабораторной работы были созданы два проекта на языке программирования C++: gost\_34\_11 и sha-1. Соответственно, каждый отвечает

В приложении с листингом кода предоставлена вырезка кода с функцией хэширования SHA-1.

Реализованное программное средство имеет возможность считать файл, захэшировать его содержимое и вывести на экран.

# Заключение

В ходе выполнения данной работы было произведено ознакомление с теорией об алгоритмах хэширования ГОСТ 34.11 и SHA-1, реализовано программное средство для хэширования этими двумя алгоритмами.

Индустрия инфраструктуры открытых рекомендует, чтобы любой объект инфраструктуры, использующий SHA-1, был переведён на более безопасный SHA-2.

Старые модификации ГОСТ 34.11 считаются потенциально уязвимыми.

Цель лабораторной работы, поставленная во введении настоящего отчёта, можно считать достигнутой.

# Приложение А

(Листинг кода)

class SHA1 {

public:

SHA1();

void update(const std::string &s);

void update(std::istream &is);

std::string final();

static std::string hash\_from\_file(const std::string &filename);

private:

uint32\_t digest[5];

std::string buffer;

uint64\_t transforms;

};

constexpr const size\_t BLOCK\_INTS {16}; /\* number of 32bit integers per SHA1 block \*/

constexpr const size\_t BLOCK\_BYTES {BLOCK\_INTS \* 4};

void reset(uint32\_t digest[], std::string &buffer, uint64\_t &transforms) ;

uint32\_t rol(const uint32\_t value, const size\_t bits) ;

uint32\_t blk(const uint32\_t block[BLOCK\_INTS], const size\_t i) ;

void R1(uint32\_t block[BLOCK\_INTS], const uint32\_t v, uint32\_t &w, const uint32\_t x, const uint32\_t y, uint32\_t &z, const size\_t i) ;

void R2(uint32\_t block[BLOCK\_INTS], const uint32\_t v, uint32\_t &w, const uint32\_t x, const uint32\_t y, uint32\_t &z, const size\_t i) ;

void R3(uint32\_t block[BLOCK\_INTS], const uint32\_t v, uint32\_t &w, const uint32\_t x, const uint32\_t y, uint32\_t &z, const size\_t i) ;

void R4(uint32\_t block[BLOCK\_INTS], const uint32\_t v, uint32\_t &w, const uint32\_t x, const uint32\_t y, uint32\_t &z, const size\_t i) ;