Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Информационные сети. Основы безопасности

ОТЧЕТ

к лабораторной работе №5

на тему

**РАЗРАБОТКА ЗАЩИЩЕННЫХ ПРИЛОЖЕНИЙ**

Студент Д. С. Кончик

Преподаватель Е. А. Лещенко

Минск 2024

**СОДЕРЖАНИЕ**

[1 Цель работы 3](#_Toc157412482)

[2 Краткие теоретические сведения 4](#_Toc157412483)

[3 Демонстрация программного средства](#_Toc157412484) 6

[Приложение А Исходный код программы](#_Toc157412485) 7

# 1 ЦЕЛЬ РАБОТЫ

Цель данной лабораторной работы заключается в разработке приложения-инсталлятора, которое предназначено для установки на компьютер пользователя предварительно созданного приложения. Основная задача приложения-инсталлятора заключается в обеспечении удобного и безопасного процесса установки программного продукта на целевой компьютер. В частности, он должен гарантировать защиту от незаконного копирования и распространения приложения.

# 2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Программа установки, или установщик, является программой, предназначенной для размещения программного обеспечения на компьютере. Некоторые установщики специализированы на установке определенных файлов, в то время как другие могут работать с разнообразными комплектами программного обеспечения.

Установщики программ могут иметь различные функции, включая проверку системных требований, создание ярлыков и выбор директории для установки. Они также могут выполнять дополнительные действия, такие как регистрация компонентов или настройка параметров программы.

Вариативность установщиков также может выражаться в предоставлении пользователю выбора опций установки, таких как типы установки или компоненты для установки. Кроме того, они могут включать механизмы защиты, чтобы обеспечить безопасность программных компонентов и предотвратить несанкционированный доступ или изменения.

Электронная подпись в установщиках программ играет ключевую роль в обеспечении безопасности. Она гарантирует подлинность и целостность установочных файлов, что обеспечивает доверие к источнику их происхождения. Процесс создания электронной подписи включает криптографические алгоритмы для генерации уникального цифрового ключа и последующее его подписание частным ключом разработчика. При установке ПО установщик проверяет электронную подпись, используя открытый ключ, чтобы убедиться в ее подлинности и целостности.

# 3 ДЕМОНСТРАЦИЯ ПРОГРАММНОГО СРЕДСТВА

В результате лабораторной работы было создано приложение-инсталлятор, позволяющее установить программу на устройство. Защита инсталлятора обеспечивается с помощью цифровой подписи, созданной с помощью предварительно сгенерированного сертификата (рисунок 2).

Информация о цифровой подписи представлена на рисунке 1.

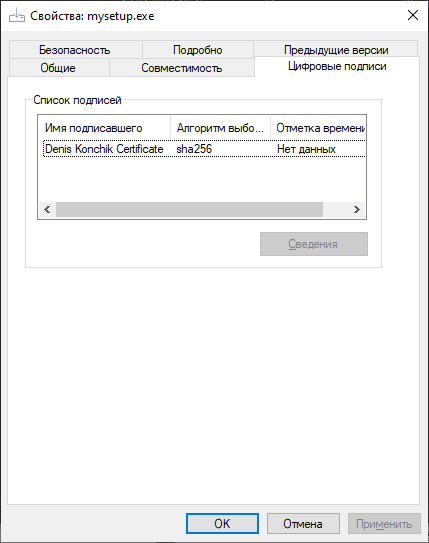


Рисунок 1 – Информация о цифровой подписи



Рисунок 2 – Сертификат

# ПРИЛОЖЕНИЕ А

**Исходный код программы**

import sys

from PyQt5.QtWidgets import QApplication, QMainWindow, QLabel, QLineEdit, QPushButton, QMessageBox

from PyQt5.QtGui import QPalette, QColor, QRadialGradient, QBrush

from admin\_window import AdminWindow

from user\_window import UserWindow

from database import add\_user, authenticate, create\_tables

class MainWindow(QMainWindow):

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

        self.setWindowTitle("Authentication System")

        self.resize(350, 150)

        # Создание и размещение элементов интерфейса

        self.username\_label = QLabel("Username:", self)

        self.username\_label.move(20, 20)

        self.username\_entry = QLineEdit(self)

        self.username\_entry.move(120, 20)

        self.password\_label = QLabel("Password:", self)

        self.password\_label.move(20, 60)

        self.password\_entry = QLineEdit(self)

        self.password\_entry.move(120, 60)

        self.password\_entry.setEchoMode(QLineEdit.Password)

        self.login\_button = QPushButton("Login", self)

        self.login\_button.move(20, 100)

        self.login\_button.clicked.connect(self.login)

        create\_tables()

        add\_user('denis', 'denis123#A', 'user')

        add\_user('admin', 'admin123#A', 'admin')

        # Установка фона градиентом

        palette = QPalette()

        gradient = QRadialGradient(200, 100, 200, 200, 100)

        gradient.setColorAt(0.0, QColor(255, 255, 0))

        gradient.setColorAt(1.0, QColor(255, 0, 0))

        palette.setBrush(QPalette.Window, QBrush(gradient))

        self.setPalette(palette)

    def login(self):

        username = self.validate\_input(self.username\_entry.text())

        password = self.validate\_input(self.password\_entry.text())

        user = authenticate(username, password)

        if user:

            if user[2] == "admin":

                self.admin\_window = AdminWindow()

                self.admin\_window.show()

            else:

                self.user\_window = UserWindow()

                self.user\_window.show()

        else:

            QMessageBox.critical(self, "Error", "Invalid username or password.")

    # defense from buffer overflow

    def validate\_input(self, value):

        if value is not None and len(value) > 50:

            return None

        return value

def main():

    app = QApplication(sys.argv)

    main\_window = MainWindow()

    main\_window.show()

    sys.exit(app.exec\_())

if \_\_name\_\_ == "\_\_main\_\_":

    main()