МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**Факультет безопасности информационных технологий**

**Направление подготовки: 10.03.01 Информационная безопасность**

**Образовательная программа: "Информационная безопасность / Information security"**

**Дисциплина:**

**«*Информационная безопасность баз данных*»**

**ОТЧЕТ ПО ЛАБОРАТОРНОЙРАБОТЕ № 5**

***«Шифрование в PostgreSQL»***

**Выполнил студент:**

группа/поток 1.3

Бардышев Артём Антонович/\_\_\_\_\_\_\_\_\_\_

*Подпись*

**Проверил:**

Карманова Наталья Андреевна/\_\_\_\_\_\_\_\_\_\_

*Подпись*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Отметка о выполнении (один из вариантов:*

*отлично, хорошо, удовлетворительно, зачтено)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Дата*

Санкт-Петербург

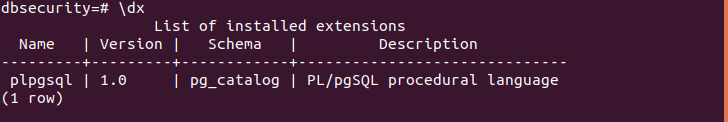
2025г.

**Шифрование в PostgreSQL**

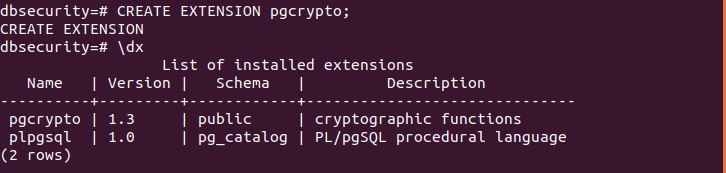
***Хеширование паролей***

**PostgreSQL предлагает несколько типов шифрования данных. Первым типом является хеширование паролей, которое позволяет хранить в базе данных хеши парольных фраз и ключей для работы приложения. Обычно в процессе аутентификации проверяется именно совпадение хешей паролей. Хеширование – это необратимый процесс без возможности расшифровать входные значения. Однако хеш можно проверить путем повторного хеширования входного значения.**

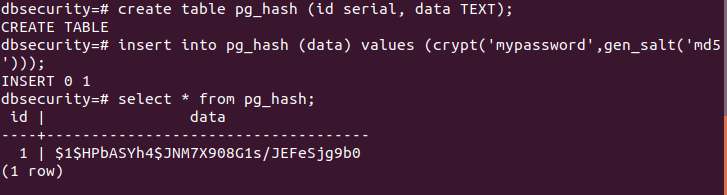
**Во-первых, нужно добавить расширение для активации функции шифрования. Проверьте список установленных расширений:**

****

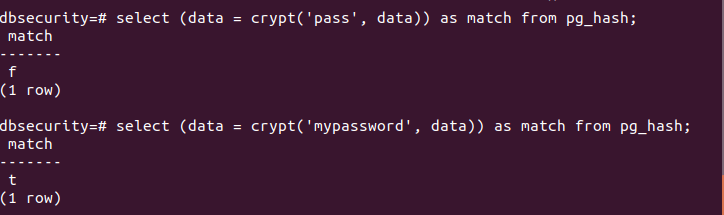
**Далее устанавливаем расширение pgcrypto:**

****

**Создадим выделенную таблицу для хешированных значений и вставим в нее хеш с помощью следующих нескольких запросов:**

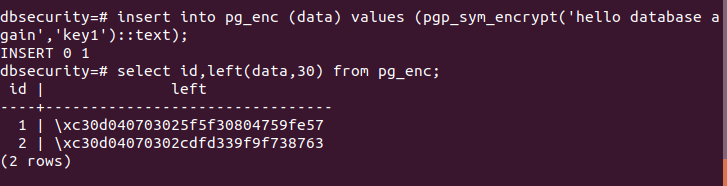
****

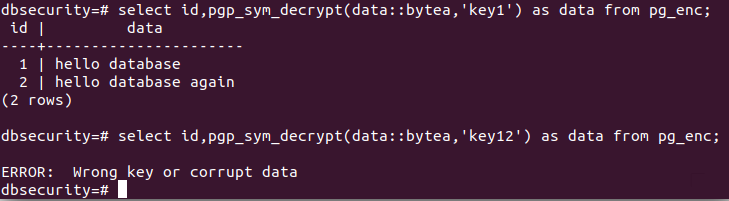
**Введенный пароль можно проверить запросом вызова функции *crypt ()*.**

****

***Шифрование столбцов таблицы***

**Еще одной особенностью PostgreSQL является возможность шифрования выделенных столбцов таблицы. Клиент может зашифровать критичные значения с точки зрения безопасности, которые находятся в некоторых столбцах. Вместо одностороннего хеширования шифрование может быть обратимым (симметричным). Пользователь может расшифровать данные, если знает ключ, который использовался при шифровании.**

****

****

**Задачи для практической работы:**

**1. Создайте таблицу, в которой два столбца содержат хешированные значения, где одно из них сгенерировано с помощью алгоритма SHA-1. Покажите, как можно выполнить проверку, используя данные двух хешей.**

**2. Создайте таблицу, в которой данные имеют байтовый тип. Зашифруйте этот столбец и покажите, как пользователь может расшифровать данные во время обычного select-запроса к зашифрованному столбцу.**

ХОД РАБОТЫ:

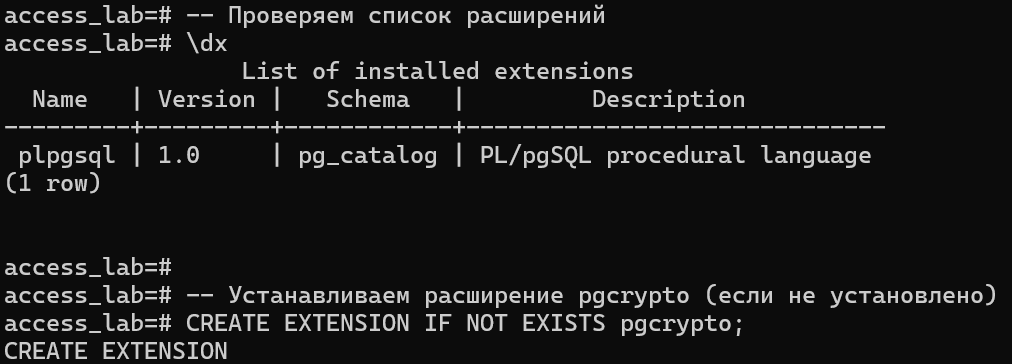
1. установка расширения pgcrypto

-- Проверяем список расширений

\dx

-- Устанавливаем расширение pgcrypto (если не установлено)

CREATE EXTENSION IF NOT EXISTS pgcrypto;



1) **Создание таблицы:**

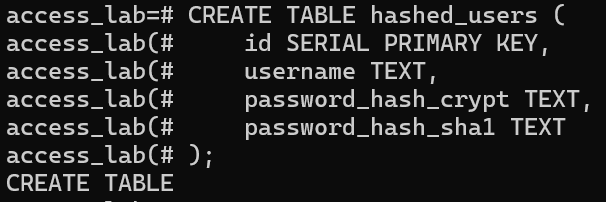
CREATE TABLE hashed\_users (

id SERIAL PRIMARY KEY,

username TEXT,

password\_hash\_crypt TEXT,

password\_hash\_sha1 TEXT );



2) **Вставка хешированных значений:**

**-- Хеширование с помощью crypt (по умолчанию используется Blowfish)**

**INSERT INTO hashed\_users (username, password\_hash\_crypt, password\_hash\_sha1)**

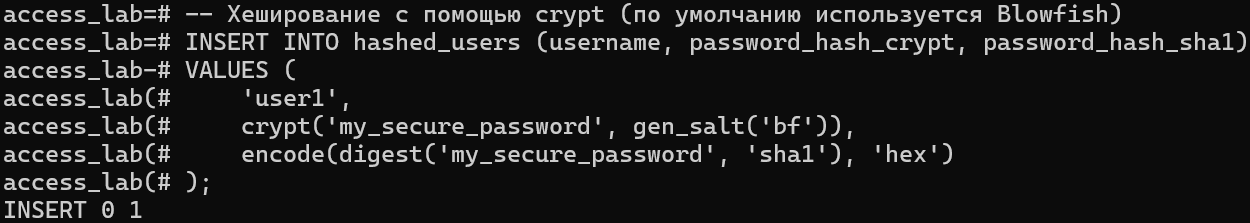
**VALUES (**

**'user1',**

**crypt('my\_secure\_password', gen\_salt('bf')),**

**encode(digest('my\_secure\_password', 'sha1'), 'hex')**

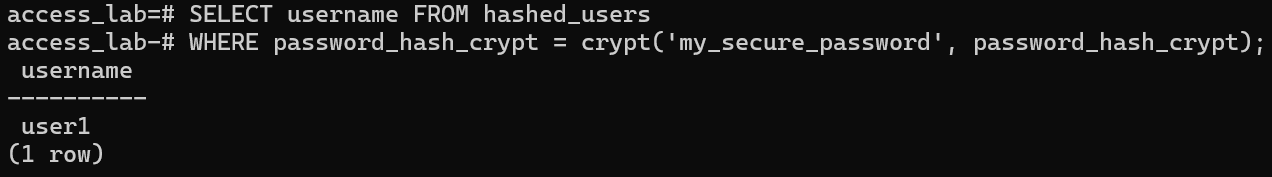
**);**



3) **Проверка пароля:**

SELECT username FROM hashed\_users

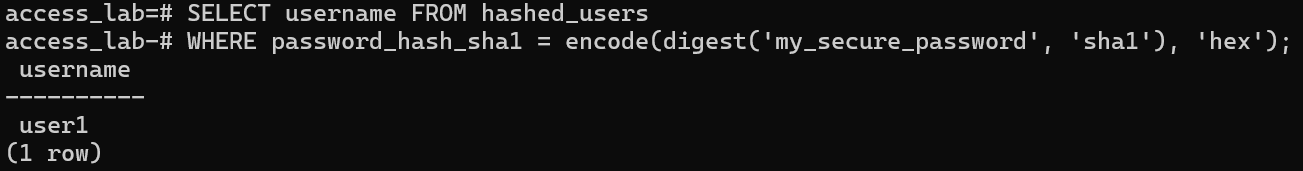
WHERE password\_hash\_crypt = crypt('my\_secure\_password', password\_hash\_crypt);



Проверка SHA-1 хеша:

SELECT username FROM hashed\_users

WHERE password\_hash\_sha1 = encode(digest('my\_secure\_password', 'sha1'), 'hex');



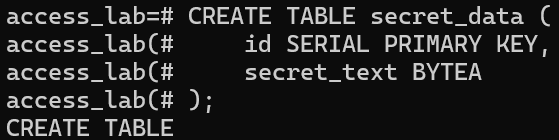
4) **Создание таблицы с байтовым полем:**

CREATE TABLE secret\_data (

id SERIAL PRIMARY KEY,

secret\_text BYTEA

);



5) **Шифрование данных (симметричное шифрование):**

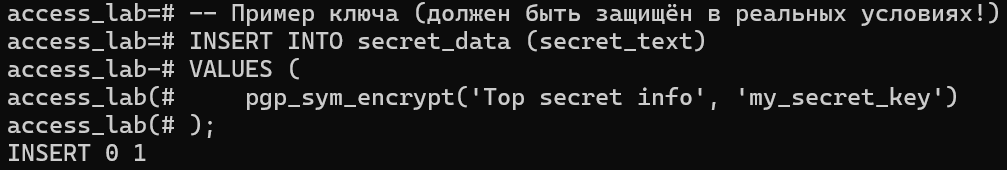
**-- Пример ключа (должен быть защищён в реальных условиях!)**

**INSERT INTO secret\_data (secret\_text)**

**VALUES (**

**pgp\_sym\_encrypt('Top secret info', 'my\_secret\_key')**

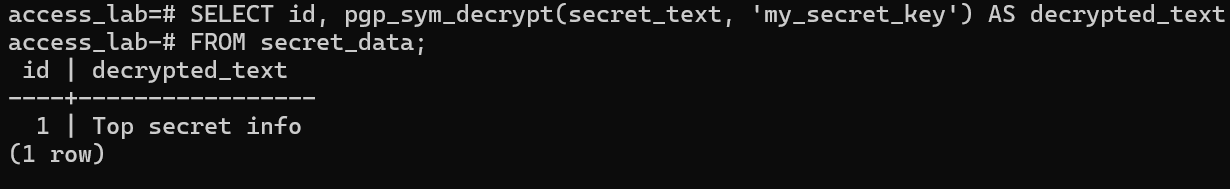
**);**



6) **Расшифровка данных при SELECT:**

SELECT id, pgp\_sym\_decrypt(secret\_text, 'my\_secret\_key') AS decrypted\_text

FROM secret\_data;



Вывод:

В ходе выполнения лабораторной работы были изучены основные методы криптографической защиты данных в PostgreSQL. Мы подключили расширение pgcrypto, которое позволяет использовать функции хеширования и шифрования прямо внутри базы данных.

В первом задании мы создали таблицу с хешированными значениями, используя алгоритмы crypt() и SHA-1. Мы научились проверять пароль, сравнивая хеши, что важно для безопасного хранения и проверки данных без их раскрытия.

Во втором задании мы реализовали симметричное шифрование и расшифровку данных в таблице с использованием типа BYTEA и функций pgp\_sym\_encrypt() и pgp\_sym\_decrypt(). Это позволяет безопасно хранить чувствительную информацию и получать доступ к ней только при наличии ключа.

Таким образом, были получены практические навыки защиты данных средствами СУБД PostgreSQL, что является важной частью обеспечения информационной безопасности современных приложений и систем.