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:**

```
dbsecurity=# \dx
                List of installed extensions
  Name   | Version |   Schema   |         Description
---------+---------+------------+------------------------------
 plpgsql | 1.0     | pg_catalog | PL/pgSQL procedural language
(1 row)
```

**Далее устанавливаем расширение pgcrypto:**

```
dbsecurity=# CREATE EXTENSION pgcrypto;
CREATE EXTENSION
dbsecurity=# \dx
                 List of installed extensions
   Name   | Version |   Schema   |         Description
----------+---------+------------+------------------------------
 pgcrypto | 1.3     | public     | cryptographic functions
 plpgsql  | 1.0     | pg_catalog | PL/pgSQL procedural language
(2 rows)
```

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

```
dbsecurity=# create table pg_hash (id serial, data TEXT);
CREATE TABLE
dbsecurity=# insert into pg_hash (data) values (crypt('mypassword',gen_salt('md5
')));
INSERT 0 1
dbsecurity=# select * from pg_hash;
 id |                data
----+-----------------------------------
  1 | $1$HPbASYh4$JNM7X908G1s/JEFeSjg9b0
(1 row)
```

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

```
dbsecurity=# select (data = crypt('pass', data)) as match from pg_hash;
 match
-------
 f
(1 row)

dbsecurity=# select (data = crypt('mypassword', data)) as match from pg_hash;
 match
-------
 t
(1 row)
```

***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.**

```
dbsecurity=# insert into pg_enc (data) values (pgp_sym_encrypt('hello database a
gain','key1')::text);
INSERT 0 1
dbsecurity=# select id,left(data,30) from pg_enc;
 id |              left
----+--------------------------------
  1 | \xc30d040703025f5f30804759fe57
  2 | \xc30d04070302cdfd339f9f738763
(2 rows)
```

```
dbsecurity=# select id,pgp_sym_decrypt(data::bytea,'key1') as data from pg_enc;
 id |         data
----+----------------------
  1 | hello database
  2 | hello database again
(2 rows)

dbsecurity=# select id,pgp_sym_decrypt(data::bytea,'key12') as data from pg_enc;

ERROR:  Wrong key or corrupt data
dbsecurity=#
```

**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:

0) установка расширения pgcrypto

-- Проверяем список расширений

\dx

-- Устанавливаем расширение pgcrypto (если не установлено)

CREATE EXTENSION IF NOT EXISTS pgcrypto;

```
access_lab=# -- Проверяем список расширений
access_lab=# \dx
                List of installed extensions
  Name   | Version |   Schema   |         Description
---------+---------+------------+------------------------------
 plpgsql | 1.0     | pg_catalog | PL/pgSQL procedural language
(1 row)


access_lab=#
access_lab=# -- Устанавливаем расширение pgcrypto (если не установлено)
access_lab=# CREATE EXTENSION IF NOT EXISTS pgcrypto;
CREATE EXTENSION
```

1) **Создание таблицы:**

CREATE TABLE hashed_users (

id SERIAL PRIMARY KEY,

username TEXT,

password_hash_crypt TEXT,

password_hash_sha1 TEXT );

```
access_lab=# CREATE TABLE hashed_users (
access_lab(#     id SERIAL PRIMARY KEY,
access_lab(#     username TEXT,
access_lab(#     password_hash_crypt TEXT,
access_lab(#     password_hash_sha1 TEXT
access_lab(# );
CREATE TABLE
```

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')**

**);**

```
access_lab=# -- Хеширование с помощью crypt (по умолчанию используется Blowfish)
access_lab=# INSERT INTO hashed_users (username, password_hash_crypt, password_hash_sha1)
access_lab-# VALUES (
access_lab(#     'user1',
access_lab(#     crypt('my_secure_password', gen_salt('bf')),
access_lab(#     encode(digest('my_secure_password', 'sha1'), 'hex')
access_lab(# );
INSERT 0 1
```

3) **Проверка пароля:**

SELECT username FROM hashed_users

WHERE password_hash_crypt = crypt('my_secure_password', password_hash_crypt);

```
access_lab=# SELECT username FROM hashed_users
access_lab-# WHERE password_hash_crypt = crypt('my_secure_password', password_hash_crypt);
 username
----------
 user1
(1 row)
```

Проверка SHA-1 хеша:

SELECT username FROM hashed_users

WHERE password_hash_sha1 = encode(digest('my_secure_password', 'sha1'), 'hex');

```
access_lab=# SELECT username FROM hashed_users
access_lab-# WHERE password_hash_sha1 = encode(digest('my_secure_password', 'sha1'), 'hex');
 username
----------
 user1
(1 row)
```

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

CREATE TABLE secret_data (

id SERIAL PRIMARY KEY,

secret_text BYTEA

);

```
access_lab=# CREATE TABLE secret_data (
access_lab(#     id SERIAL PRIMARY KEY,
access_lab(#     secret_text BYTEA
access_lab(# );
CREATE TABLE
```

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')**

**);**

```
access_lab=# -- Пример ключа (должен быть защищён в реальных условиях!)
access_lab=# INSERT INTO secret_data (secret_text)
access_lab-# VALUES (
access_lab(#     pgp_sym_encrypt('Top secret info', 'my_secret_key')
access_lab(# );
INSERT 0 1
```

6) **Расшифровка данных при SELECT:**

SELECT id, pgp_sym_decrypt(secret_text, 'my_secret_key') AS decrypted_text

FROM secret_data;

```
access_lab=# SELECT id, pgp_sym_decrypt(secret_text, 'my_secret_key') AS decrypted_text
access_lab-# FROM secret_data;
 id | decrypted_text
----+----------------
  1 | Top secret info
(1 row)
```

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.