ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ им. проф. М.А. БОНЧ-БРУЕВИЧА» (СПбГУТ)

Факультет Информационных технологий и программной инженерии
Кафедра Систем обработки данных

ОТЧЕТ

по практическому занятию №3

Тема: «Настройка шифрования базы данных»

Выполнил: студент 3 курса, гр. ИБ-31вп
_____ Ворожцов А. Е.

Принял:
_____ Медведев С. А.

Санкт-Петербург, 2025 г.

Содержание

Отчет по настройке SSL, FDW и шифрованию данных

Создание таблицы с конфиденциальными данными

SQL-скрипт создаёт таблицу сотрудников в `db2` и резервирует столбец `salary` под шифрование (`db2/init/02-employees.sql`).

```
CREATE TABLE IF NOT EXISTS employees (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    phone VARCHAR(20),
    "position" VARCHAR(50),
    salary BYTEA,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Логи генерации сертификатов OpenSSL в контейнерах

`docker-compose` автоматически запускает openssl из init-скриптов. Ниже приведены ключевые строки журналов (`docker compose logs db1` / `db2`).

```
# db1 (извлечение)
/usr/local/bin/docker-entrypoint.sh: sourcing /docker-entrypoint-initdb.d/05-ssl.sh
....+++++++++++++++++++++++++++++++++++++*
.....+++++++++++++++++++++++++++++++++++++*
-----
writing RSA key
Warning: Not placing -key in cert or request since request is used
Warning: No -copy_extensions given; ignoring any extensions in the request
ALTER SYSTEM
ALTER SYSTEM
ALTER SYSTEM
```

```
# db2 (извлечение)
/usr/local/bin/docker-entrypoint.sh: sourcing /docker-entrypoint-initdb.d/06-ssl.sh
..+.+.....+..+.......+.....+.+..+.......+...
-----
writing RSA key
Warning: Not placing -key in cert or request since request is used
Warning: No -copy_extensions given; ignoring any extensions in the request
ALTER SYSTEM
ALTER SYSTEM
ALTER SYSTEM
```

Состояние файловой системы с сертификатами

Сертификаты обеих БД лежат в общем каталоге `/certs` внутри контейнеров.

```
$ docker compose exec db1 ls -l /certs
-rw-r--r--    1 root     root          4104 Nov 12 20:33 db1-server.crt
-rw-r--r--    1 root     root          4104 Nov 12 20:33 db2-server.crt
```

```
$ docker compose exec db2 ls -l /certs
-rw-r--r--    1 root     root          4104 Nov 12 20:33 db1-server.crt
-rw-r--r--    1 root     root          4104 Nov 12 20:33 db2-server.crt
```

Скрипт подключения сервера db2 к db1 через FDW

`db2/init/03-fdw.sql` создаёт пользователя, FDW-сервер и включает проверку сертификата `verify-full`.

```
CREATE EXTENSION IF NOT EXISTS postgres_fdw;

CREATE SERVER my_work_server_name
FOREIGN DATA WRAPPER postgres_fdw
OPTIONS (
    host 'db1',
    port '5432',
    dbname 'db1',
    sslmode 'verify-full',
    sslrootcert '/certs/db1-server.crt'
);

CREATE USER MAPPING IF NOT EXISTS FOR user_connect
SERVER my_work_server_name
OPTIONS (user 'user_connect', password 'user_connect');
```

Результат запроса к таблице ключей через FDW

```
$ SELECT encode(key_name_sha256,'hex') AS key_name, key_material, created_at FROM keys ORDER BY created_at LIMIT 6;"
                            key_name                              |   key_material    |          created_at
------------------------------------------------------------------+-------------------+-------------------------------
 e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855 | 41c991eb6a66242c… | 2025-11-12 20:34:01.511136+00
 ff8d9819fc0e12bf0d24892e45987e249a28dce836a85cad60e28eaaa8c6d976 | EiZXzgCzh4Li2fYX… | 2025-11-12 20:34:01.587456+00
 5ff860bf1190596c7188ab851db691f0f3169c453936e9e1eba2f9a47f7a0018 | xEX67JWLblhEXFqJ… | 2025-11-12 20:34:01.597175+00
 e0d47ca1bc1eb62e650fc1fd660a9bfbf7cba8dc6337d81df7ea9aa9071a24a5 | lfAkuXHbZwBUF6ot… | 2025-11-12 20:34:01.598027+00
 7b34211350ff567970974e1e2b98d319a601969e74fd1a957bc889b8332d00eb | jq7nJ9dUlosQm066… | 2025-11-12 20:34:01.598849+00
 d0574c4966d2c326193622feebc64991c5b59807ae68fa8255b26c79f4bf917a | dpxWcIWJIIFyqkeU… | 2025-11-12 20:34:01.601825+00
(6 rows)
```

Хранимая процедура вставки шифрованных данных

`db2/init/07-insert-employee.sql` генерирует ключ, синхронизирует его через FDW и шифрует зарплату.

```
CREATE OR REPLACE FUNCTION add_employee(
    p_name TEXT,
    p_email TEXT,
    p_phone TEXT,
    p_position TEXT,
    p_salary TEXT
)
RETURNS INTEGER
LANGUAGE plpgsql
AS $$
DECLARE
    v_key_name BYTEA := digest(p_email, 'sha256');
    v_key_material TEXT;
    v_salary_encrypted BYTEA;
    v_employee_id INTEGER;
BEGIN
    SELECT key_material
      INTO v_key_material
      FROM keys
     WHERE key_name_sha256 = v_key_name;

    IF NOT FOUND THEN
        v_key_material := encode(gen_random_bytes(32), 'base64');
        INSERT INTO keys (key_name_sha256, key_material, created_at)
        VALUES (v_key_name, v_key_material, now());
    END IF;

    v_salary_encrypted := pgp_sym_encrypt(p_salary::text, v_key_material);

    INSERT INTO employees(name, email, phone, "position", salary)
    VALUES (p_name, p_email, p_phone, p_position, v_salary_encrypted)
    RETURNING id INTO v_employee_id;

    RETURN v_employee_id;
END;
$$;
```

Функция расшифровки

`db2/init/08-get-employees.sql` валидирует мастер-пароль и расшифровывает зарплату.

```
CREATE OR REPLACE FUNCTION get_employees_with_salary(p_password TEXT)
RETURNS TABLE (
    id INTEGER,
```

```
    name VARCHAR(100),
    email VARCHAR(100),
    phone VARCHAR(20),
    "position" VARCHAR(50),
    salary TEXT,
    created_at TIMESTAMP
)
LANGUAGE plpgsql
AS $$
DECLARE
    v_master_hash TEXT;
BEGIN
    SELECT key_material
      INTO v_master_hash
      FROM keys
     WHERE key_name_sha256 = digest('', 'sha256');

    IF v_master_hash IS NULL THEN
        RAISE EXCEPTION 'Master key is not configured in keys table';
    END IF;

    IF crypt(coalesce(p_password, ''), v_master_hash) <> v_master_hash THEN
        RAISE EXCEPTION 'Invalid password';
    END IF;

    RETURN QUERY
    SELECT e.id,
           e.name,
           e.email,
           e.phone,
           e."position",
           pgp_sym_decrypt(e.salary, k.key_material)::text AS salary,
           e.created_at
      FROM employees e
      JOIN keys k ON k.key_name_sha256 = digest(e.email, 'sha256');
END;
$$;
```

Скрипт демонстрации и создание мастер-пароля

`scripts/demo.sh` добавляет мастер-пароль 7777, вставляет сотрудников, показывает шифрование и проверяет оба варианта расшифровки.

```
#!/bin/bash
set -euo pipefail

MASTER_PASSWORD="7777"
```

```
echo "[demo] Добавляем мастер-пароль db1"
docker compose exec -T db1 psql -U user -d db1 <<SQL
WITH master_key AS (
  SELECT digest('', 'sha256') AS key_name,
         crypt('${MASTER_PASSWORD}', gen_salt('bf')) AS key_material
)
INSERT INTO keys (key_name_sha256, key_material)
SELECT key_name, key_material FROM master_key
ON CONFLICT (key_name_sha256) DO UPDATE
  SET key_material = EXCLUDED.key_material,
      created_at = now();
SQL

echo "[demo] Сбрасываем предыдущие демо данные"
docker compose exec -T db2 psql -U user -d db2 <<'SQL'
WITH victims AS (
  SELECT digest(email, 'sha256') AS key_name FROM employees
)
DELETE FROM keys WHERE key_name_sha256 IN (SELECT key_name FROM victims);
TRUNCATE TABLE employees RESTART IDENTITY;
SQL

echo "[demo] Добавляем демо данные"
docker compose exec -T db2 psql -U user -d db2 <<'SQL'
SELECT add_employee('Alice', 'alice@example.com', '+1-111-111-1111', 'Engineer',
 '120000');
SELECT add_employee('Bob', 'bob@example.com', '+1-222-222-2222', 'Manager',
 '150000');
SELECT add_employee('Carol', 'carol@example.com', '+1-333-333-3333', 'Scientist',
 '135000');
SELECT add_employee('Dave', 'dave@example.com', '+1-444-444-4444', 'Analyst',
 '90000');
SELECT add_employee('Eve', 'eve@example.com', '+1-555-555-5555', 'CISO',
 '200000');
SQL

echo "[demo] Чистый вывод данных"
docker compose exec -T db2 psql -U user -d db2 -c "SELECT id, name, email, phone,
 \"position\", left(encode(salary,'hex'),3) || '***' AS salary_enc, created_at
  FROM employees ORDER BY id;"

echo "[demo] Попытка с неправильным паролем"
set +e
docker compose exec -T db2 psql -U user -d db2 -c "SELECT * FROM
get_employees_with_salary('wrong-password');"
WRONG_STATUS=$?
```

```
set -e
if [ $WRONG_STATUS -eq 0 ]; then
  echo "[demo] Внезапно успех" >&2
  exit 1
else
  echo "[demo] Пароль неправильный"
fi

echo "[demo] Вывод с расшифровкой"
docker compose exec -T db2 psql -U user -d db2 -c "SELECT * FROM
get_employees_with_salary('${MASTER_PASSWORD}');"
```

В разделе “Чистый вывод данных” команда возвращает псевдоним `salary_enc`, где выводится только `left(encode(salary,'hex'),3) || '***'`, чтобы не раскрывать полное содержимое зашифрованной колонки.

Результаты `make run`

Команда запускает скрипт выше и демонстрирует шифрование/расшифровку (фрагмент вывода).

```
$ make run
./scripts/demo.sh
[demo] Добавляем мастер-пароль db1
INSERT 0 1
[demo] Сбрасываем предыдущие демо данные
DELETE 5
TRUNCATE TABLE
[demo] Добавляем демо данные
 add_employee
--------------
            1
(1 row)

 add_employee
--------------
            2
(1 row)

 add_employee
--------------
            3
(1 row)

 add_employee
--------------
            4
(1 row)
```

```
 add_employee
--------------
            5
(1 row)
```

```
[demo] Чистый вывод данных
 id | name  |       email       |      phone      | position  | salary_enc |         created_at
----+-------+-------------------+-----------------+-----------+------------+----------------------------
  1 | Alice | alice@example.com | +1-111-111-1111 | Engineer  | c30***     | 2025-11-12 20:43:28.367
  2 | Bob   | bob@example.com   | +1-222-222-2222 | Manager   | c30***     | 2025-11-12 20:43:28.375805
  3 | Carol | carol@example.com | +1-333-333-3333 | Scientist | c30***     | 2025-11-12 20:43:28.376897
  4 | Dave  | dave@example.com  | +1-444-444-4444 | Analyst   | c30***     | 2025-11-12 20:43:28.37779
  5 | Eve   | eve@example.com   | +1-555-555-5555 | CISO      | c30***     | 2025-11-12 20:43:28.378469
(5 rows)

[demo] Попытка с неправильным паролем
ERROR:  Invalid password
CONTEXT:  PL/pgSQL function get_employees_with_salary(text) line 16 at RAISE
[demo] Пароль неправильный
[demo] Вывод с расшифровкой
 id | name  |       email       |      phone      | position  | salary |         created_at
----+-------+-------------------+-----------------+-----------+--------+----------------------------
  1 | Alice | alice@example.com | +1-111-111-1111 | Engineer  | 120000 | 2025-11-12 20:43:28.367
  2 | Bob   | bob@example.com   | +1-222-222-2222 | Manager   | 150000 | 2025-11-12 20:43:28.375805
  3 | Carol | carol@example.com | +1-333-333-3333 | Scientist | 135000 | 2025-11-12 20:43:28.376897
  4 | Dave  | dave@example.com  | +1-444-444-4444 | Analyst   | 90000  | 2025-11-12 20:43:28.37779
  5 | Eve   | eve@example.com   | +1-555-555-5555 | CISO      | 200000 | 2025-11-12 20:43:28.378469
(5 rows)
```

Выводы

- Каждая база генерирует собственный сертификат и сохраняет его в общем каталоге `/certs`, что позволяет `db2` проверять сертификат `db1` в режиме `verify-full`.
- Таблица `employees` хранит зашифрованные зарплаты, а ключи лежат на `db1` и доступны через FDW только после валидации мастер-пароля.
- Скрипт `make run` демонстрирует полный сценарий: генерацию мастер-ключа, вставку сотрудников, просмотр зашифрованных данных, ошибку при неверном пароле и успешную расшифровку при правильном.