/\* Задача №1 \*/

/\*1.Устанавливаем расширение pgcrypto\*/

CREATE EXTENSION IF NOT EXISTS pgcrypto;

/\*2.Создаём таблицу для шифрования данных\*/

CREATE TABLE x (

id SERIAL PRIMARY KEY,

data bytea

);

/\*3.Создаём функцию для шифрования\*/

CREATE OR REPLACE FUNCTION encrypt\_data()

RETURNS trigger AS $$

DECLARE

encryption\_key TEXT := 'djaJdLPukajMIjuUJlqetr';

BEGIN

NEW.data := pgp\_sym\_encrypt(

NEW.data::text,

encryption\_key,

'cipher-algo=aes256'

);

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

/\*4. Создаём триггер \*/

CREATE TRIGGER trig\_encrypt\_data

BEFORE INSERT ON x

FOR EACH ROW

EXECUTE PROCEDURE encrypt\_data();

/\*5. Тестируем шифрование \*/

INSERT INTO x (data)

VALUES ('Секретная информация');

SELECT id, data FROM x;

/\*6. Создание процедуры для расшифровки \*/

CREATE OR REPLACE FUNCTION decrypt\_data(p\_id INT)

RETURNS TEXT AS $$

DECLARE

encryption\_key TEXT := 'djaJdLPukajMIjuUJlqetr';

decrypted\_data TEXT;

BEGIN

SELECT pgp\_sym\_decrypt\_bytea(data, encryption\_key)

INTO decrypted\_data

FROM x

WHERE id = p\_id;

RETURN decrypted\_data;

END;

$$ LANGUAGE plpgsql;

/\*7. Расшифровываем данные \*/

SELECT decrypt\_data(1);

/\* Задача №2 \*/

/\*1.Создание функции для хеширование строки \*/

CREATE OR REPLACE FUNCTION hash\_string(p\_input TEXT)

RETURNS TEXT AS $$

DECLARE

salt TEXT;

hashed\_result TEXT;

BEGIN

salt := gen\_salt('bf');

hashed\_result := crypt(p\_input, salt);

RETURN hashed\_result;

END;

$$ LANGUAGE plpgsql;

/\*2. Проверяем функцию для хеширования строки \*/

SELECT hash\_string('MyPassword');

/\* Задача №3 \*/

В PostgreSQL для crypt() доступны алгоритмы:

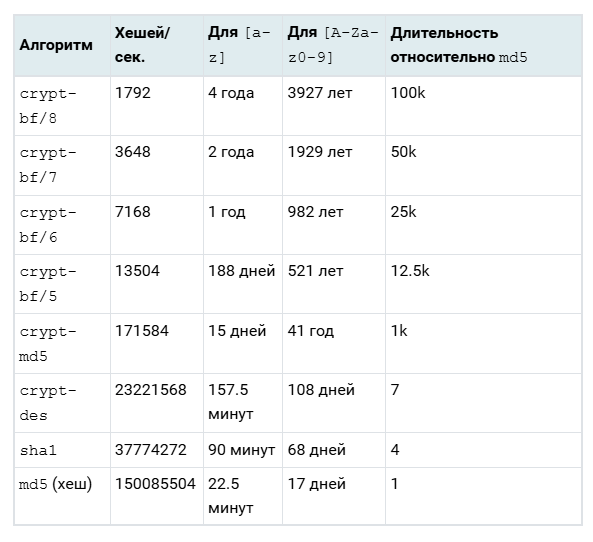
1.bf(Blowfish)

2.md5

3.xdes

4.des

Согласно данным, приведённым в таблице из официальной документации PostgreSQL, самым безопасным алгоритмом является bf.



Кроме того, в другой таблице даётся более подробная информация о характеристиках каждого алгоритма и bf по этим показателям также оказывается самым безопасным: максимальная длина позволяет задать достаточно сложный пароль, адаптивность увеличивает количество итераций и сложность перебора, а

большой размер соли уменьшает вероятность коллизии и не даёт использовать единый «предрассчитанный» словарь хэшей.

