Zad1

Hashowanie:

- Hashowanie jest jednokierunkowym procesem, gdzie dane wejściowe (hasło) są przekształcane na stałą długość ciągu znaków, zwanej "hashem".
- Popularne funkcje haszujące to np. SHA-256, SHA-3, MD5 (choć nie jest zalecane ze względu na swoje ograniczone bezpieczeństwo).
- Hasło jest zapisane w postaci hasha, a nie w oryginalnej postaci, co utrudnia odzyskanie pierwotnego hasła.

Sól (Salt):

- Dodanie soli do hasła oznacza dodanie losowego ciągu znaków przed lub po haśle przed zastosowaniem funkcji haszującej.
- Każde hasło ma swoją unikalną sól, co uniemożliwia atakującemu korzystanie z tych samych haseł w różnych kontekstach.

Funkcje Key Derivation:

- Funkcje key derivation, takie jak bcrypt, scrypt, Argon2, są zaprojektowane specjalnie do bezpiecznego przechowywania haseł.
- Obejmują one również dodatkowe mechanizmy opóźniania (iteracji), co utrudnia atakom typu brute force.

PBKDF2 (Password-Based Key Derivation Function 2):

- Jest to funkcja key derivation oparta na hasłach, która powtarza operacje haszowania wielokrotnie, aby utrudnić atakującym.
- PBKDF2 można dostosować, aby zwiększyć ilość iteracji wraz z postępem technologii, zapewniając większe bezpieczeństwo.

AES (Advanced Encryption Standard):

- AES jest symetrycznym algorytmem szyfrowania, który można stosować do szyfrowania danych, w tym haseł.
- W przypadku przechowywania haseł zaszyfrowanych AES, konieczne jest bezpieczne przechowywanie klucza szyfrowania.

SSL/TLS:

- W kontekście komunikacji z bazą danych, używanie protokołów SSL/TLS jest kluczowe do zabezpieczenia transmisji danych, w tym haseł.

- Uwierzytelnianie dwuskładnikowe (2FA):

- Chociaż nie jest to bezpośrednio związane z kryptografią w bazie danych, implementacja 2FA dodaje dodatkową warstwę zabezpieczeń, nawet jeśli hasła zostaną skompromitowane.

Zad2	SHA-256	SHA-512
Ochrona	SHA-256 jest bezpiecznym algorytmem i jest najczęściej używany. Jest obliczany przy użyciu słów 32-bitowych.	SHA-512 zapewnia lepsze bezpieczeństwo niż SHA-256, ale obecnie nie jest powszechnie stosowany. Jest obliczany przy użyciu słów 64-bitowych.
kontatybilność	SHA 256 jest kompatybilny z systemami operacyjnymi Apple, Android, Blackberry, Chrome i Windows. Jest także obsługiwany przez przeglądarki Chrome, Firefox, Internet Explorer, Mozilla, Opera i Safari.	SHA 512 jest obsługiwany przez system operacyjny Windows, gdy protokół TLS 1.2 nie jest używany.
zastosowania	SHA 256 jest używany w protokołach uwierzytelniania. Przydaje się przy mieszaniu haseł w systemach Unix i Linux. Kryptowaluty mogą używać SHA-256 do weryfikacji transakcji.	SHA 512 jest używany do mieszania adresów e-mail i weryfikacji rekordów cyfrowych. Podobnie jak SHA 256, jest również przydatny do mieszania haseł i w łańcuchu bloków.
Wielkość HASHU	Rozmiar skrótu SHA 256 wynosi 256 bitów.	Rozmiar skrótu SHA 512 wynosi 512 bitów.

Zad3

Długość klucza w algorytmach kryptograficznych wpływa na poziom bezpieczeństwa danego systemu. Ogólnie rzecz biorąc, im dłuższy klucz, tym trudniejsze jest jego złamanie za pomocą ataków kryptoanalitycznych, takich jak brute force czy ataki z użyciem zaawansowanych algorytmów.

Zad4

Za pomocą sol lub key defision

Zad5

Szyfrowanie danych za pomocą klucza, który jest łatwy do złamania, nie spełnia głównego celu szyfrowania, którym jest zapewnienie bezpieczeństwa i poufności informacji. Szyfrowanie jest stosowane w celu zabezpieczenia danych przed nieautoryzowanym dostępem, a używanie łatwo złamanych kluczy podważa tę ochronę.

Zad6

```
// Dane do zapisania w bazie

$username = "example_user";

$password = "tajneHaslo";
```

```
// Generowanie soli
$salt = bin2hex(random_bytes(16));

// Haszowanie hasła z solą
$hashedPassword = password_hash($password . $salt, PASSWORD_BCRYPT);

// Przykładowy SQL do zapisania danych
$sql = "INSERT INTO users (username, password, salt) VALUES (?, ?, ?)";
$stmt = $conn->prepare($sql);
$stmt->bind_param("sss", $username, $hashedPassword, $salt);
$stmt->execute();

echo "Dane zaszyfrowane zapisane do bazy.";

// Zamknięcie połączenia z bazą danych
$stmt->close();
$conn->close();
?>
```

Zad7

```
<?php

// Dane logowania z formularza (przykład)
$inputUsername = "example_user";
$inputPassword = "tajneHaslo";

// Pobranie danych użytkownika z bazy danych
$sql = "SELECT * FROM users WHERE username = ?";
$stmt = $conn->prepare($sql);
$stmt->bind_param("s", $inputUsername);
$stmt->execute();
$result = $stmt->get_result();
```

```
if ($result->num_rows > 0) {
    // Użytkownik istnieje, sprawdź hasło
    $row = $result->fetch_assoc();
    $hashedPassword = $row['password'];
    $salt = $row['salt'];

    // Sprawdzenie hasła z użyciem funkcji password_verify
    if (password_verify($inputPassword . $salt, $hashedPassword)) {
        echo "Logowanie udane. Witaj, " . $inputUsername . "!";
    } else {
        echo "Nieprawidłowe hasło.";
    }
} else {
    echo "Nieprawidłowy użytkownik.";
}

// Zamknięcie połączenia z bazą danych
$stmt->close();
$conn->close();
?>
```