МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Практическая работа №9**

по дисциплине «Основы информационной безопасности»

на тему: Изучение стандартных средств для реализации симметричного и ассиметричного шифрование с использованием SubtleCrypto в js

Выполнил студент 2 курса 7 группы специальность ПОИБМС Володькин Никифор Дмитриевич

(Ф.И.О.)

Преподаватель Ржеутская Надежда Викентьевна

(Ф.И.О.)

**Цель: изучить интерфейс SubtleCrypto, Основные классы и структуры данных, разработать приложение для шифрования файлов использующих симметричные и ассиметричные алгоритмы шифрования.**

**Теоретическое введение**

Web Crypto API - это интерфейс, позволяющий использовать криптографические примитивы для построения систем с использованием криптографии. Данный интерфейсвключают в себя возможность генерировать, использовать и применять пары криптографических ключей; шифровать и дешифровать сообщения; надежно генерировать случайные числа

Некоторые браузеры реализовали интерфейс под названием Crypto, без точной структуры. Чтобы избежать путаницы, методы и свойства этого интерфейса были удалены из браузеров, реализующих Web Crypto API, и все методы Web Crypto API доступны в новом интерфейсе: SubtleCrypto.

Интерфейс SubtleCrypto Web Crypto API предоставляет ряд низкоуровневых криптографических функций. Доступ к функциям SubtleCrypto осуществляется через объёкт Crypto.subtle .

Этот объект содержит набор методов для выполнения общих криптографических функций, таких как шифрование, хеширование, подписывание и генерация ключей. Поскольку все криптографические операции выполняются с необработанными двоичными данными, каждый метод SubtleCrypto имеет дело с типами ArrayBuffer и ArrayBufferView. Из-за того, что строки так часто становятся предметом криптографических операций, классы TextEncoder и TextDecoder будут часто использоваться вместе с SubtleCrypto для преобразования в строки и обратно.

**Выполнение работы**

Было написано следующее приложение на языке JavaScript, реализующее генерацию случайных чисел, алгоритмы шифрования и хеширования AES-GCM и SHA384, упаковку ключа с использованием AES-KW и подпись сообщения с использованием ECDSA:

const array = new Uint8Array(10);

console.table(crypto.getRandomValues(array));

const createHash = async message => {

const encoder = new TextEncoder();

const data = encoder.encode(message);

const hash = await crypto.subtle.digest('SHA-384', data);

const hashArray = Array.from(new Uint8Array(hash));

const hashHex = hashArray.map((b) => b.toString(16).padStart(2, '0')).join('');

return hashHex;

};

createHash("Wolodkin")

.then(hash => console.log(hash));

const generateKey = async () => {

const params = {

name: 'AES-GCM',

length: 256

};

const keyUsages = ['encrypt', 'decrypt'];

const key = await crypto.subtle.generateKey(params, true, keyUsages);

return key;

};

const encryptAESGCM = async (message, key) => {

const originalPlaintext = (new TextEncoder()).encode(message);

const encryptDecryptParams = {

name: 'AES-GCM',

iv: crypto.getRandomValues(new Uint8Array(16))

};

const ciphertext = await crypto.subtle.encrypt(encryptDecryptParams, key, originalPlaintext);

return { ciphertext, encryptDecryptParams };

};

const wrapKey = async key => {

const wrappingKeyUsages = ['wrapKey', 'unwrapKey'];

const wrappingKeyParams = {

name: 'AES-KW',

length: 256

};

const keyAlgoIdentifier = 'AES-GCM';

const keyUsages = ['encrypt', 'decrypt'];

const keyParams = {

name: 'AES-GCM',

length: 256

};

const wrappingKey = await crypto.subtle.generateKey(wrappingKeyParams, true, wrappingKeyUsages);

const wrappedKey = await crypto.subtle.wrapKey('raw', key, wrappingKey, 'AES-KW');

return { wrappingKey, wrappedKey, wrappingKeyParams, keyParams, keyUsages };

};

const encryptDecryptAndWrapKey = async message => {

const key = await generateKey();

const encryptedValue = await encryptAESGCM(message, key);

const decryptedValue = await crypto.subtle.decrypt(encryptedValue.encryptDecryptParams, key, encryptedValue.ciphertext);

console.log('Key:');

console.table(key);

console.log('Encrypted message:');

console.table(encryptedValue);

console.log(`Decrypted message: ${(new TextDecoder()).decode(decryptedValue)}`);

const wrappedKey = await wrapKey(key);

const unwrappedKey = await crypto.subtle.unwrapKey('raw', wrappedKey.wrappedKey,

wrappedKey.wrappingKey, wrappedKey.wrappingKeyParams, wrappedKey.keyParams, true, wrappedKey.keyUsages);

console.log('Wrapped key:');

console.table(wrappedKey);

console.log('Unrapped key:');

console.table(unwrappedKey);

console.log(`Unwrapped key matches original key: ${JSON.stringify(unwrappedKey) === JSON.stringify(key)}`);

};

encryptDecryptAndWrapKey("Wolodkin");

const signAndVerify = async message => {

const keyParams = {

name: 'ECDSA',

namedCurve: 'P-256'

};

const keyUsages = ['sign', 'verify'];

const { publicKey, privateKey } = await crypto.subtle.generateKey(keyParams, true, keyUsages);

const encodedMessage = (new TextEncoder()).encode(message);

const signParams = {

name: 'ECDSA',

hash: 'SHA-384'

};

const signature = await crypto.subtle.sign(signParams, privateKey, encodedMessage);

const verified = await crypto.subtle.verify(signParams, publicKey, signature, encodedMessage);

console.log(`Verified: ${verified}`);

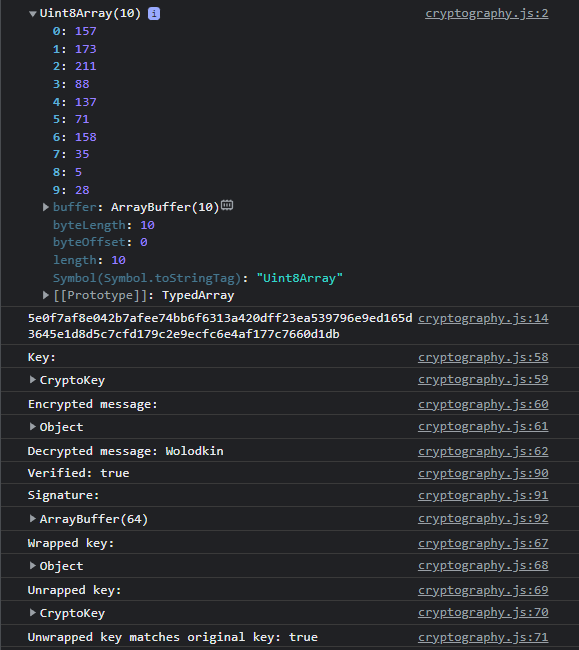
console.log('Signature:');

console.table(signature);

};

signAndVerify("Wolodkin");

Результат выполнения программы:



**Вывод: изучен интерфейс SubtleCrypto, Основные классы и структуры данных, разработано приложение для шифрования файлов использующих симметричные и ассиметричные алгоритмы шифрования.**