ИНФОРМАТИКА И ИНФОРМАЦИОННЫЕ ПРОЦЕССЫ

УДК 004.056.5 Научная статья
DOI: 10.35330/1991-6639-2024-26-1-39-47
EDN: GEVSAQ

Разработка подхода к обеспечению информационной безопасности в веб-ориентированных информационных системах при передаче данных с использованием интерфейса Web Cryptography API

М. В. Ступина

Донской государственный технический университет
344003, Россия, ЮФО, г. Ростов-на-Дону, пл. Гагарина, 1

Аннотация. Целью исследования является формулирование общих принципов обеспечения информационной безопасности в веб-ориентированных информационных системах. В работе описаны основные концепции интерфейса Web Cryptography API, а также представлены практические аспекты использования криптографических методов для обеспечения безопасности данных веб-ориентированных информационных систем. Предложенный подход, основанный на введении безопасной системы генерации и хранения приватных ключей пользователей через использование асинхронного алгоритма шифрования ECDSA средствами интерфейса Web Cryptography API, в сочетании с шифрованием приватных ключей кодовыми словами и дополнительной аутентификацией пользователей позволяет обеспечить высокий уровень защиты приватных ключей от несанкционированного доступа.

Ключевые слова: Web Cryptography API, криптография, электронная подпись, электронный документооборот, ECDSA

Поступила 05.02.2024, одобрена после рецензирования 08.02.2024, принята к публикации 12.02.2024

Для цитирования. Ступина М. В. Разработка подхода к обеспечению информационной безопасности в веб-ориентированных информационных системах при передаче данных с использованием интерфейса Web Cryptography API // Известия Кабардино-Балкарского научного центра РАН. 2024. Т. 26. № 1. С. 39–47. DOI: 10.35330/1991-6639-2024-26-1-39-47

MSC: 68P20; 68U35 Original article

Development of an approach to ensuring information security in web-based information systems when transferring data using the Web Cryptography API interface

M.V. Stupina

Don State Technical University
344003, Russia, Rostov-on-Don, 1 Gagarina square

Abstract. The aim of the research is to formulate general principles for ensuring information security in web-oriented information systems. The paper describes the main concepts of the Web Cryptography API interface, as well as presents practical aspects of using cryptographic methods to ensure data security in web-oriented information systems. The proposed approach, based on the introduction of a secure system for generating and storing users private keys through the use of the

asynchronous ECDSA encryption algorithm via the Web Cryptography API interface, combined with encrypting private keys with passphrases and additional user authentication, allows a high level of protection of private keys from unauthorized access.

Keywords: Web Cryptography API, cryptography, electronic signature, electronic document management, ECDSA

Submitted 05.02.2024, approved after reviewing 08.02.2024, accepted for publication 12.02.2024

For citation. Stupina M.V. Development of an approach to ensuring information security in web-based information systems when transferring data using the Web Cryptography API interface. *News of the Kabardino-Balkarian Scientific Center of RAS.* 2024. Vol. 26. No. 1. Pp. 39–47. DOI: 10.35330/1991-6639-2024-26-1-39-47

Введение

Сегодня все возрастающую значимость приобретает необходимость обеспечения безопасности персональных данных[1] в веб-ориентированных информационных системах (ВОИС), что обусловлено происходящими процессами цифровой трансформации современного общества, тенденциями повсеместного использования интернет-технологий, переводом большинства бизнес-процессов в онлайн-режим [1]. Переход множества компаний на электронный документооборот, позволяющий ускорить и упростить бизнес-процессы, минимизировать затраты на обработку документов, их физическое хранение и т. д. [1], в свою очередь требует должного уровня защиты конфиденциальной информации при ее хранении и передаче с использованием цифровых каналов связи [2]. Для ряда компаний критически важным является подтверждение подлинности отправителя, достоверности и целостности передаваемой информации, что определяет необходимость использования электронной цифровой подписи [1, 3] как неотъемлемой составляющей информационной безопасности при реализации электронного документооборота.

Безопасность при организации онлайн-взаимодействия в ВОИС требует применения криптографических методов защиты на стороне браузера, гарантирующих безопасность онлайн-протоколов и защиту конфиденциальных данных. Реализация криптографических операций в браузере может быть выполнена с использованием ряда инструментов, одним из которых является рекомендованный W3C интерфейс Web Cryptography API, представляющий стандартизированный способ доступа к криптографическим функциям веб-браузера без необходимости обращения к сторонним библиотекам [4, 5]. Возможности Web Cryptography API поддерживают шаблоны использования ключей в соответствии с лучшими практиками в криптографии [5], а созданные объекты Web Crypto API исключают случайное использование ключей или алгоритмов в недопустимом состоянии.

В рамках настоящего исследования рассмотрим подход к обеспечению информационной безопасности при реализации обмена данными в ВОИС с использованием программного интерфейса Web Cryptography API.

Описание подхода к реализации электронной подписи на базе Web Cryptography API

Интерфейс Web Cryptography API, выбранный для создания методов электронной подписи, предоставляет для генерации ключей множество криптографических алгоритмов [6]. Однако для реализации электронной подписи принято использовать асимметричные алгоритмы шифрования, например, RSA или ECDSA – один из вариантов DSA-алгоритма

[1] Федеральный закон «О персональных данных» от 27.07.2006 № 63-ФЗ // Российская газета. 2006 г. [Электронный ресурс]: https://rg.ru/documents/2006/07/29/personaljnye-dannye-dok.html (дата обращения: 20.01.2024).

с более высокой степенью надежности. С учетом проведенного анализа[2] для реализации электронной подписи был выбран алгоритм ECDSA, демонстрирующий высокий уровень криптостойкости, производительности, а также возможность достижения программной совместимости и масштабируемости.

В целом подход к реализации электронной подписи заключается в выполнении следующей последовательности действий и рекомендаций, сформулированных с учетом требований Web Cryptography API [4–6]:

1. Генерация ключей должна происходить на клиентской части ВОИС (непосредственно в браузере) при помощи асинхронного алгоритма шифрования ECDSA средствами интерфейса Web Cryptography API.
2. Приватный ключ должен зашифровываться кодовым словом пользователя и в зашифрованном виде храниться на сервере.
3. Для подписания электронного документа пользователю ВОИС будет необходимо повторно ввести кодовое слово для развертывания закрытого ключа, полученного с сервера, и дополнительного подтверждения своей личности.

Соблюдение указанных рекомендаций гарантирует защиту приватного ключа и невозможность его использования для создания электронной подписи без знания кодового слова. Подтверждение операций, отсутствие ошибок во время процедуры подписи документа и защита от фальсификации данных реализуются путем процедуры верификации электронной подписи открытым ключом со стороны сервера при выполнении соответствующего запроса. Таким образом пользователь ВОИС подтверждает свою личность при подписании электронного документа.

Реализация методов создания электронной подписи и подписания электронных документов

Для выпуска электронной подписи необходимо сгенерировать приватный и публичный ключи посредством алгоритма ECDSA.

Создадим асинхронный метод generateKeys(), в котором будет происходить вызов метода generateKey() интерфейса WebCryptoApi. Код метода generateKeys() представлен на рисунке 1.

```
/* Сгенерировать пару ключей */
const generateKeys = async () => {
  return window.crypto.subtle.generateKey(
    {
      name: "ECDSA",
      namedCurve: "P-384",
    },
    true,
    ['sign']
  );
};
```

Рис. 1. *Код метода generateKeys()*

Fig. 1. *The code for the generateKeys() method*

[2]Ступина М. В., Илющенко А. Н. Сравнительный анализ алгоритмов электронной подписи [Электронный ресурс] // Молодой исследователь Дона. 2023. № 8 (3). С. 78–81. URL: https://mid-journal.ru/upload/iblock/952/17_1711-Ilyushchenko_78_81.pdf (дата обращения: 29.01.2024).

При генерации ключей посредством ECDSA алгоритма параметр algorithm представляет собой EcKeyGenParams объект. Данный объект должен включать имя алгоритма name – ECDSA и namedCurve – имя используемой эллиптической кривой в соответствии с рекомендациями NIST. Как показано в работе [7], оптимальным является выбор P-384. Параметр extractable передадим true, т.к. в дальнейшем потребуется использовать метод SubtleCrypto.wrapKey(). В массив keyUsages включим метод sign, потому что впоследствии данным ключом будет производиться подпись документов. Результатом выполнения метода generateKeys() будет объект типа CryptoKeyPair, включающий созданные публичный и приватный ключи электронной подписи.

Далее, согласно сформулированному подходу, необходимо зашифровать полученный приватный ключ кодовым словом пользователя.

Создадим асинхронный метод getCodewordAsKey(), внутри которого будет происходить преобразование кодового слова из строки в CryptoKey. При помощи вызова enc.encode(codeword).buffer преобразуем кодовое слово – строку в ArrayBuffer – двоичное представление данных. Далее последовательно выполним методы интерфейса WebCryptoApi digest() и importKey(). В метод digest() передадим параметром алгоритм дайджеста (SHA-256) в формате объекта и ранее полученное кодовое слово в формате ArrayBuffer. Код метода getCodewordAsKey() представлен на рисунке 2.

```
/* Преобазовать кодовое слово в CryptoKey */
const getCodewordAsKey = async (codeword: string) => {
  const enc = new TextEncoder();

  let codewordAsArrayBuffer: ArrayBuffer = enc.encode(codeword).buffer;

  return crypto.subtle.digest({ name: 'SHA-256' }, codewordAsArrayBuffer).then((result) =>
    crypto.subtle.importKey(
      'raw',
      result,
      {
        name: 'AES-CBC',
        length: 256,
      },
      false,
      ['wrapKey', 'unwrapKey']
    )
  );
};
```

Рис. 2. *Код метода getCodewordAsKey()*

Fig. 2. *The code for the getCodewordAsKey() method*

Далее создадим асинхронный метод wrapPrivateKey(), который в качестве параметров будет принимать приватный ключ электронной подписи, кодовое слово из заполненного пользователем поля формы и вектор инициализации – случайную последовательность байтов, который понадобится в дальнейшем также и для дешифровки ключа.

В этом методе вызовем ранее созданный метод getCodewordAsKey() для того, чтобы преобразовать кодовое слово из строки в CryptoKey, и после этого вызовем метод интерфейса WebCryptoApi wrapKey(). Параметр wrapAlgo – объект, указывающий

алгоритм, который будет использоваться для шифрования ключа (AES-CBC), где iv – вектор инициализации, передаваемый аргументом в метод wrapPrivateKey(). Код метода wrapPrivateKey() продемонстрирован на рисунке 3.

```
/* Зашифровать privateKey кодовым словом */
const wrapPrivateKey = async (keyToWrap: CryptoKey, codeword: string, iv: Uint8Array) => {
  const wrappingKey: CryptoKey = await getCodewordAsKey(codeword);

  const wrapParams = { name: 'AES-CBC', iv: iv };
  return window.crypto.subtle.wrapKey('pkcs8', keyToWrap, wrappingKey, wrapParams);
};
```

Рис. 3. *Код метода wrapPrivateKey()*

Fig. 3. *The code for the wrapPrivateKey() method*

Для отправки сгенерированных ключей на сервер необходимо преобразовать их в формат BASE64. Для этого реализуем асинхронные методы preparePublicKey() и preparePrivateKey().

В методе preparePublicKey() вызовем метод интерфейса WebCryptoApi exportKey(), который принимает в качестве входных данных CryptoKey объект и возвращает ключ во внешнем переносимом формате. Результат выполнения предыдущей операции преобразуем в Uint8Array, а затем в BASE64. Код метода preparePublicKey() представлен на рисунке 4.

```
// преобразовать publicKey из CryptoKey в BASE64
const preparePublicKey = async (publicKey: CryptoKey) => {
  const publicKeyToArrayBuffer = await window.crypto.subtle.exportKey('spki', publicKey);
  const publicKeyToUint8Array = new Uint8Array(publicKeyToArrayBuffer);
  return Base64.fromUint8Array(publicKeyToUint8Array);
};
```

Рис. 4. *Код метода preparePublicKey()*

Fig. 4. *The code for the preparePublicKey()method*

Метод preparePrivateKey() дополнительно позволяет объединить приватный ключ и вектор инициализации для дальнейшей возможности их совместной передачи и разъединения. Код данного метода представлен на рисунке 5.

```
// Объединить iv и wrapped privateKey и опреобразовать в BASE64
const preparePrivateKey = async (privateWrapKey: ArrayBuffer, iv: Uint8Array) => {
  const privateWrapKeyToU8A = new Uint8Array(privateWrapKey);
  const mergedWrappedPrivateKeyAndIV = new Uint8Array(privateWrapKeyToU8A.length + iv.length);
  mergedWrappedPrivateKeyAndIV.set(iv);
  mergedWrappedPrivateKeyAndIV.set(privateWrapKeyToU8A, iv.length);

  return Base64.fromUint8Array(mergedWrappedPrivateKeyAndIV);
};
```

Рис. 5. *Код метода preparePrivateKey()*

Fig. 5. *The code for the preparePrivateKey() method*

Вектор инициализации, полученный при оборачивании ключа кодовым словом, впоследствии будет нужен для обратной операции – расшифровки приватного ключа для осуществления подписи документа. Преобразуем зашифрованный кодовым словом приватный ключ в Uint8Array и выполним объединение массивов. Далее преобразуем полученное значение в BASE64.

На рисунке 6 представлен код метода createSignature(), который предназначен непосредственно для выпуска электронной подписи.

```
/* возвращает пару ключей в BASE64 (public и зашифрованный кодовым словом privateKey+iv) */
export const createSignature = async (codeword: string) => {
  const keyPair: CryptoKeyPair = await generateKeys();
  const iv: Uint8Array = window.crypto.getRandomValues(new Uint8Array(16));
  const privateWrapKey: ArrayBuffer = await wrapPrivateKey(keyPair.privateKey, codeword, iv);
  const publicKey: string = await preparePublicKey(keyPair.publicKey);
  const privateKey: string = await preparePrivateKey(privateWrapKey, iv);

  return {publicKey, privateKey};
};
```

Рис. 6. *Код метода createSignature()*

Fig. 6. *The code for the createSignature() method*

Далее создадим метод getUnwrapKey(), принимающий параметрами приватный ключ, вектор инициализации и кодовое слово. Результатом выполнения данного метода будет распакованный приватный ключ в формате CryptoKey. Код данного метода представлен на рисунке 7.

```
/* Расшифровать privateKey введенным кодовым словом */
const getUnwrapKey = async (privateKey: ArrayBuffer, iv: Uint8Array, codeword: string) => {
  const unwrappingKey: CryptoKey | undefined = await getCodewordAsKey(codeword);
  const unwrapAlgo = { name: 'AES-CBC', iv };
  const unwrappedKeyAlgo = { name: 'ECDSA', hash: 'P-384', };

  return window.crypto.subtle.unwrapKey(
    'pkcs8',
    privateKey,
    unwrappingKey,
    unwrapAlgo,
    unwrappedKeyAlgo,
    true,
    ['sign']
  );
};
```

Рис. 7. *Код метода getUnwrapKey()*

Fig. 7. *The code for the getUnwrapKey() method*

Для подписания документа необходимо сначала расшифровать приватный ключ, получаемый с сервера. Создадим асинхронный метод unwrapPrivateKey(), принимающий кодовое слово и приватный ключ (объединенный с вектором инициализации). Здесь происходит вызов метода getUnwrapKey() с использованием механизма для обработки исключений. Если операция распаковки ключа выполнена успешно, метод вернет расшифрованный приватный ключ в формате CryptoKey. Код метода продемонстрирован на рисунке 8.

```
/* Отделить iv от privateKey и расшифровать кодовым словом */
const unwrapPrivateKey = async (codeword: string, privateKeyBase64: string) => {
  const privateKeyAndIv: Uint8Array = Base64.toUint8Array(privateKeyBase64);
  const iv: Uint8Array = privateKeyAndIv.subarray(0, 16);
  const privateKey: ArrayBufferLike = privateKeyAndIv.slice(16).buffer;
  let unwrappedKey: CryptoKey | undefined;
  try {
    unwrappedKey = await getUnwrapKey(privateKey, iv, codeword);
  } catch (error) {
    throw new AppError(AppErrorCode.WRONG_CODEWORD, `Неверное кодовое слово.`);
  }
  return unwrappedKey;
};
```

Рис. 8. *Код метода unwrapPrivateKey()*

Fig. 8. *The code for the unwrapPrivateKey() method*

Для осуществления подписи документов создадим асинхронный метод signDocument(), принимающий параметрами файл, содержащий документ, и приватный ключ. В рамках данного метода вызовем метод интерфейса WebCryptoApi sign(). Результатом выполнения данного метода является подписанный документ в формате ArrayBuffer. Код данного метода представлен на рисунке 9.

```
/* Подписать документ */
const signDocument = async (unwrappedKey: CryptoKey, hashedDocument: ArrayBuffer) => {
  const signAlgo = {name: 'ECDSA', hash: 'SHA-256'};
  return window.crypto.subtle.sign(
    signAlgo,
    unwrappedKey,
    hashedDocument
  );
}
```

Рис. 9. *Код метода signDocument()*

Fig. 9. *The code for the signDocument() method*

Далее создадим асинхронный метод signatureDocument(), принимающий параметрами файл документа, кодовое слово и приватный ключ (объединенный с вектором инициализации), полученные с сервера. Вызовем созданный ранее метод unwrapPrivateKey(), передав в него кодовое слово, объединенный приватный ключ и вектор инициализации. Далее вызовем метод signDocument(), передав в него полученный распакованный приватный ключ и файл документа. Полученную последовательность в формате ArrayBuffer подписанного файла преобразуем в BASE64 формат для отправки на сервер. Код данного метода продемонстрирован на рисунке 10.

```
/* Расшифровать приватный ключ кодовым словом и подписать документ */
export const signatureDocument = async (file: any, codeword: string, privateKey: string) => {
  const unwrappedKey: CryptoKey = await unwrapPrivateKey(codeword, privateKey);
  const signature = await signDocument(unwrappedKey, file);
  const signatureToBase64 = Base64.fromUint8Array(new Uint8Array(signature));

  return signatureToBase64;
};
```

Рис. 10. *Код метода signatureDocument()*

Fig. 10. *The code for the signatureDocument() method*

Заключение

В целом подход, основанный на применении методов спецификации Web Cryptography API, обеспечивает защиту данных пользователей на клиентской стороне ВОИС. Помимо описанных возможностей создания электронной цифровой подписи, область применения Web Cryptography API варьирует от аутентификации пользователя или службы, подписи документов или кода, а также обеспечения конфиденциальности и целостности связи.

Применение Web Cryptography API является перспективным направлением в области определения криптографических примитивов, которые должны быть развернуты в браузерах в JS-приложениях и могут быть использованы, например, для взаимодействия с криптомодулем Node.js. Однако следует отметить, что достижение должного уровня информационной безопасности ВОИС требует разработки формализованных моделей, учитывающих противоречивые политики безопасности сочетания множества API, которые могут быть использованы при веб-разработке.

СПИСОК ЛИТЕРАТУРЫ

1. *Мехдиев Э. Т., Плеханова Е. А.* Развитие систем электронного документооборота в цифровой экономике // Дискуссия. 2023. № 1(116). С. 58–70. DOI: 10.46320/2077-7639-2022-6-115-52-70

2. *Гончаров Е. И., Шатковская Т. В.* Проблемы применения цифровой подписи в электронном документообороте России // Северо-Кавказский юридический вестник. 2020. № 2. С. 97–103. DOI: 10.22394/2074-7306-2020-1-2-97-103

3. *Баранов А. С.* Использование средств криптографической защиты информации в организациях // Международный научно-исследовательский журнал. 2020. № 6-1 (96). С. 131–133. DOI: 10.23670/IRJ.2020.96.6.023

4. *Былинский М. Д.* Защита приложений javascript с помощью Web Cryptography Api // Вестник Балтийского федерального университета им. И. Канта. Серия: Физико-математические и технические науки. 2022. № 1. С. 53–60.

5. *Cairns K., Halpin H., Steel G.* Security Analysis of the W3C Web Cryptography API // Proceedings of Security Standardisation Research (SSR). Gaithersberg. 2017. Pp. 112–140. DOI: 10.1007/978-3-319-49100-4_5

6. *Wichmann P., Blochberger M., Federrath H.* Web Cryptography API // Prevalence and Possible Developer Mistakes. In Proceedings of the 17th International Conference on Availability, Reliability and Security (ARES '22). Association for Computing Machinery. New York. 2022. Pp. 1–10. DOI: 10.1145/3538969.3538977

7. *Samir A., Abo-Taleb M., Shalaby, Nabil M., Elramly S.* A Side-Channel Attack Resistive ECDSA // International Conference on Advanced Information Systems and Engineering. Journal of Physics: Conference Series. Cairo, Egypt. 2019. Pp. 112–140. DOI: 10.1088/1742-6596/1454/1/012003

REFERENCES

1. Mekhdiev E.T., Plekhanova E.A. Development of electronic document management systems in the digital economy. *Diskussiya* [Discussion]. 2023. No. 1(116). Pp. 58–70. DOI: 10.46320/2077-7639-2022-6-115-52-70. (In Russian)

2. Goncharov E.I., Shatkovskaya T.V. Problems of using digital signatures in electronic document management in Russia. *Severo-Kavkazskiy yuridicheskiy vestnik* [North Caucasian Legal Bulletin]. 2020. No. 2. Pp. 97–103. DOI: 10.22394/2074-7306-2020-1-2-97-103. (In Russian)

3. Baranov A.S. Use of cryptographic information protection tools in organizations. *Mezhdunarodnyy nauchno-issledovatel'skiy zhurnal* [International Scientific Research Journal]. 2020. No. 6-1 (96). Pp. 131–133. DOI: 10.23670/IRJ.2020.96.6.023. (In Russian)

4. Bylinskiy M.D. Protecting JavaScript applications using the Web Cryptographs Api. *Vestnik Baltiyskogo federal'nogo universiteta im. I. Kanta. Seriya: Fiziko-matematicheskie i tekhnicheskie nauki* [Bulletin of the Baltic Federal University. I. Kant. Series: Physics, mathematics and technical sciences]. 2022. No. 1. Pp. 53–60. (In Russian)

5. Cairns K., Halpin H., Steel G. Security Analysis of the W3C Web Cryptography API. *Proceedings of Security Standardisation Research (SSR).* Gaithersberg. 2017. Pp. 112–140. DOI: 10.1007/978-3-319-49100-4_5

6. Wichmann P., Blochberger M., Federrath H. Web Cryptography API. Prevalence and Possible Developer Mistakes. *In Proceedings of the 17th International Conference on Availability, Reliability and Security (ARES '22).* Association for Computing Machinery. New York. 2022. Pp. 1–10. DOI: 10.1145/3538969.3538977

8. Samir A., Abo-Taleb M., Shalaby et al. A Side-Channel Attack Resistive ECDSA. *International Conference on Advanced Information Systems and Engineering. Journal of Physics: Conference Series.* Cairo, Egypt. 2019. Pp. 112–140. DOI: 10.1088/1742-6596/1454/1/012003

Информация об авторе

Ступина Мария Валерьевна, канд. пед. наук, доцент кафедры информационных технологий, Донской государственный технический университет;

344003, Россия, г. Ростов-на-Дону, пл. Гагарина, 1;

masamvs@bk.ru, ORCID: https://orcid.org/0000-0002-6394-6966

Information about the author

Maria V. Stupina, Candidate of Pedagogical Sciences, Associate Professor, Department of Information Technology, Don State Technical University;

344003, Russia, Rostov-on-Don, 1 Gagarina square;

masamvs@bk.ru, ORCID: https://orcid.org/0000-0002-6394-6966