УДК 004.6 https://doi.org/10.35546/kntu2078-4481.2020.1.1.18

Є.В. ЛЄПА
Херсонський національний технічний університет
ORCID: 0000-0002-2066-2998

РЕАЛІЗАЦІЯ ТА ДОСЛІДЖЕННЯ АЛГОРИТМУ DSA ДЛЯ СТВОРЕННЯ ТА ПЕРЕВІРКИ ЕЛЕКТРОННОГО ЦИФРОВОГО ПІДПИСУ

У роботі розглянутий і реалізований алгоритм DSA для створення й перевірки електронному цифровому підпису, проведені дослідження швидкості виконання дій по створенню ключової пари, підписи і її перевірки.

Для реалізації алгоритму створення використана мова програмування Java, яка має велику кількість різних бібліотек для роботи із криптографією.

Програма складається із двох модулів. Один створює цифровий електронний підпис, а другий модуль забезпечує її перевірку.

Програма має два варіанта роботи. Перший варіант передбачає створення ключової пари (відкритий та закритий ключ) для користувача та підписання документу за допомогою закритого ключа. Також програма реалізовує механізм перевірки автентичності підпису (чи був змінений документ після підписання).

Другий варіант передбачає наявність у користувача ключової пари, у цьому випадку програма використовую закритий ключ користувача (на відміну від генерування ключів у першому варіанті).

Було проведено 10000 симуляцій для дослідження швидкості створення та перевірки цифровому підпису.

Час на створення підпису при використанні хеш функції SHA-1 при довжині ключа 512 біт – 409 мс, при довжині 1024 біт – 480 мс. А при використанні хеш функції SHA-2 та довжини ключа у 2048 біт – 351 мс. Така динаміка часу, що при збільшенні довжини ключа та з тим надійності підпис створюється швидше, пов'язана перш за все з використанням більш сучасного алгоритму хешування.

За результатами досліду спостерігається лінійна залежність, яка показує що зі збільшенням довжини ключа збільшується час який необхідний на виконання операцій з підпису документа та її перевірки. Так, при довжині ключа у 512 біт для того аби поставити підпис під документом необхідно 4 мс, для перевірки – 8 мс. При розмірі ключа 1024 біт для підпису необхідно 8 мс, для перевірки підпису – 18 мс. Якщо розмір ключа 2048 біт, час який знадобиться на підпис документа 13 мс, для перевірки цього підпису – 33 мс.

Ключові слова: DSA, підпис, хеш- функція, ключ, java.

Е.В. ЛЕПА
Херсонский национальный технический университет
ORCID: 0000-0002-2066-2998

РЕАЛИЗАЦИЯ И ИССЛЕДОВАНИЕ АЛГОРИТМА DSA ДЛЯ СОЗДАНИЯ И ПРОВЕРКИ ЭЛЕКТРОННОЙ ЦИФРОВОЙ ПОДПИСИ

В работе рассмотрен и реализован алгоритм DSA для создания и проверки электронной цифровой подписи, проведены исследования скорости выполнения действий по созданию ключевой пары, подписи и ее проверки.

Для реализации алгоритма использован язык программирования java, которая имеет большое количество разных библиотек для работы с криптографией.

Программа состоит из двух модулей. Один модуль создает цифровую электронную подпись, а второй модуль обеспечивает ее проверку.

Программа имеет два варианта работы. Первый вариант предусматривает создание ключевой пары (открытый и закрытый ключ) для пользователя и подписания документа с помощью закрытого ключа. Также программа реализовывает механизм проверки аутентичности подписи (был ли измененный документ после подписания).

Второй вариант предусматривает наличие у пользователя ключевой пары, в этом случае программа использую закрытый ключ пользователя (в отличие от генерирования ключей в первом варианте).

Было проведено 10000 симуляций для исследования скорости создания и проверки цифровой подписи.

Время на создание подписи при использовании хеш- функции SHA-1 при длине ключа 512 бит – 409 мс, при длине 1024 бит – 480 мс. При использовании хеш- функции SHA-2 и длины ключа в 2048 бит – 351 мс. При увеличении длины ключа и с той же надежностью подпись создается быстрее. Это связано прежде всего с использованием более современного алгоритма хеширования.

По результатам эксперимента наблюдается линейная зависимость, которая показывает, что с увеличением длины ключа увеличивается время необходимое для выполнения операций по подписи документа и ее проверки. Так, при длине ключа в 512 бит чтобы поставить подпись под документом необходимо 4 мс, для проверки – 8 мс. При размере ключа 1024 бит для подписи необходимо 8 мс, для проверки подписи – 18 мс. Если размер ключа 2048 бит, время который понадобится на подпись документа 13 мс, для проверки этой подписи – 33 мс.

Ключевые слова: DSA, подпись, хеш- функция, ключ, java.

E.V. LEPA
Kherson National Technical University
ORCID: 0000-0002-2066-2998

IMPLEMENTATION AND RESEARCH OF THE DSA ALGORITHM FOR CREATION AND VERIFICATION OF A DIGITAL ELECTRONIC SIGNATURE

The paper considers and implements the DSA algorithm for creating and verifying a digital electronic signature, investigates the speed of actions to create a key pair, signature and its verification.

To implement the algorithm, the java programming language was used, which has a large number of different libraries for working with cryptography.

The program consists of two modules. One module creates a digital electronic signature, and the second module provides its verification.

The program has two working options. The first option involves creating a key pair (public and private key) for the user and signing the document using the private key. The program also implements a mechanism for verifying the authenticity of a signature (whether there was a changed document after signing).

The second option provides for the user to have a key pair, in this case the program uses the user's private key (as opposed to generating keys in the first option).

10,000 simulations were conducted to examine the speed of creating and verifying a digital signature.

The time to create a signature when using the SHA-1 hash function with a key length of 512 bits is 409 ms, with a length of 1024 bits - 480 ms. When using the SHA-2 hash function and a key length of 2048 bits - 351 ms. By increasing the key length and with the same reliability, the signature is created faster. This is primarily due to the use of a more modern hashing algorithm.

According to the results of the experiment, a linear relationship is observed, which shows that with an increase in the length of the key, the time required for performing operations to sign the document and verify it increases. So, with a key length of 512 bits, it takes 4 ms to sign a document, and 8 ms for verification. With a key size of 1024 bits, 8 ms is required for signature, 18 ms for signature verification. If the key size is 2048 bits, the time it takes to sign the document is 13 ms, to verify this signature - 33 ms.

Keywords: DSA, signature, hash function, key, java.

Постановка проблеми

Поява і стрімкий розвиток комп'ютерних мереж забезпечила ефективні засоби передачі даних і швидкий доступ до інформації як для окремих особистостей, так і для великих організацій, що призвело до появи електронного документообігу. Однак відомо, що локальні і глобальні комп'ютерні мережі (втім, як і інші засоби передачі інформації) можуть представляти загрозу для безпеки даних, особливо при відсутності належних заходів захисту від несанкціонованого доступу.

Для вирішення проблеми захисту при використання електронного документообігу (захист від несанкціонованого копіювання, модифікації та підробки інформації) необхідно використовувати сучасні методи захисту. Одним із поширених методів такого захисту є електронний цифровий підпис (ЕЦП). У загальному випадку, ЕЦП – це блок інформації, який додається до файлу даних автором та захищає файл від несанкціонованої модифікації і вказує на власника підпису.

Аналіз останніх досліджень і публікацій

Порівняльний аналіз методів і алгоритмів створення й перевірки електронному цифровому підпису показав, що одними з найпоширеніших є алгоритми DSA и RSA з використанням відкритого ключа, які засновані на обчислювальній складності взяття логарифмів в кінцевих полях. Тільки один суб'єкт може створити хеш-значення повідомлення, але будь-хто може перевірити її коректність.

Алгоритм DSA, має, як і RSA, теоретико-числовий характер, і заснований на криптографічній системі Ель-Гамаля у варіанті Шнорра. Його надійність заснована на практичній нерозв'язності певного

окремого випадку завдання обчислення дискретного логарифма. Сучасні методи вирішення цього завдання мають приблизно ту ж ефективність, що і методи розв'язання задачі факторизації. В зв'язку з цим пропонується використовувати ключі довжиною від 512 до 1024 біт з тими ж характеристиками надійності, що і в системі RSA. Довжина підпису в системі DSA менше, ніж в RSA, і становить 320 біт.

Формулювання мети дослідження

Метою роботи було реалізація алгоритма DSA для створення цифрового підпису, підписання документу та перевірки автентичності підпису тестування роботи програми на конкретних прикладах, дослідження швидкості процесу створення та перевірки підпису залежно від довжини ключа.

Викладення основного матеріалу дослідження

Для підписування повідомлень необхідна пара ключів - відкритий і закритий. При цьому закритий ключ повинен бути відомий тільки тому, хто підписує повідомлення, а відкритий - будь-кому, хто бажає перевірити справжність повідомлення. Також загальнодоступними є параметри самого алгоритму.

1) Вибір хеш-функції H (x). Для використання алгоритму необхідно, щоб повідомлення, що підписується, було числом. Хеш-функція повинна перетворити будь-яке повідомлення в число.
2) Вибір великого простого числа q, розмірність якого в бітах збігається з розмірністю в бітах значень хеш-функції H (x)
3) Вибір простого числа p, такого, що (p-1) ділиться на q. Розмірність p задає криптостійкість системи. Раніше рекомендувалася довжина в 1024 біта. В даний момент для систем, які повинні бути стійкими до 2010 (2030) року, рекомендується довжина в 2048 (3072) біта.
4) Вибір числа g такого, що його мультиплікативний порядок по модулю p дорівнює q. Для його обчислення можна скористатися формулою

$$g = h^{(p-1)/q} \bmod p, \tag{1}$$

де h - деяке довільне число, $h \in (1; p-1)$ таке, що $g \neq 1$. У більшості випадків значення h = 2 задовольняє цій вимозі.

Підпис повідомлення виконується за наступним алгоритмом:

1) Вибір випадкового числа $k \in (0; q)$
2) Обчислення

$$r = (g^k \bmod p) \bmod q \tag{2}$$

3) Обчислення

$$s = (k^{-1}(H(m) + x * r)) \bmod q \tag{3}$$

4) Вибір іншого k, якщо виявилося, що r = 0 або s = 0

Підписом є пара чисел (r, s)

Перевірка підпису виконується за наступним алгоритмом:

1) Обчислення

$$w = s^{-1} \bmod q \tag{4}$$

2) Обчислення

$$u_1 = (H(m) * w) \bmod q \tag{5}$$

3) Обчислення

$$u_2 = (r * w) \bmod q \tag{6}$$

4) Обчислення

$$v = ((g^{u_1} * g^{u_2}) \bmod p) \bmod q \tag{7}$$

Підпис вірний, якщо v = r

У зв’язку з тим, що на мові програмування Java є велика кількість зручних бібліотек для роботи з криптографією, кросплатформеність середовища та попередній досвід роботи з нею для реалізації

програми для розробки програми створення та перевірки електронного цифрового підпису була обрана мова java.

Програма має декілька варіантів роботи. Перший варіант передбачає створення ключової пари(відкритий та закритий ключ) для користувача та підписання документу за допомогою закритого ключа. Також програма реалізовує механізм перевірки автентичності підпису (чи був змінений документ після підписання). Другий варіант передбачає наявність у користувача ключової пари, у цьому випадку програма використовую закритий ключ користувача (на відміну від генерування ключів у першому варіанті).

Файл README.txt з текстом, якій треба підписати, зображено на рис. 1.

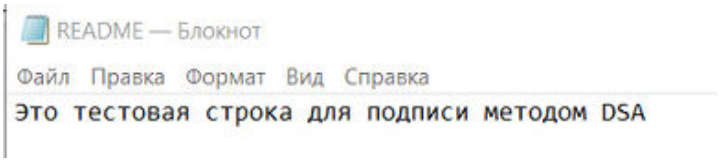

Рис. 1. Зміст тестового файлу

Після запуску програми вибирається варіант роботи з генеруванням ключової пари (рис. 2).

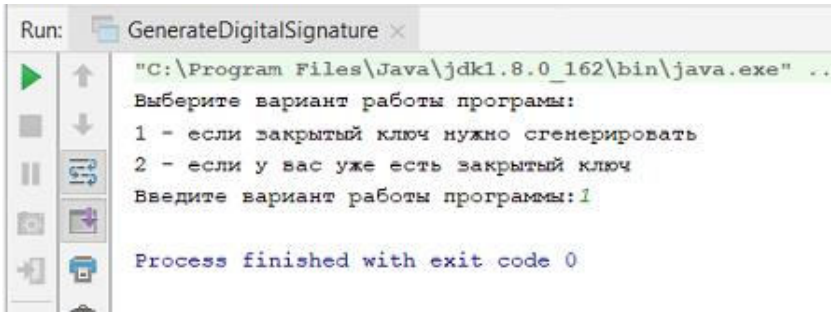

Рис. 2. Діалогове вікно програми

В результаті буде створено 3 файли з відкритим ключем, таємним ключем та з підписом для тестового файлу.

Вигляд створеного підпису представлено на рис. 3.

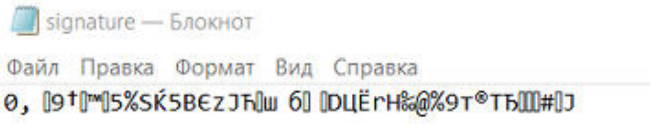

Рис. 3. Зміст файлу з підписом

Перевірка підпису буде виконано за допомогою відповідного модулю програми. Для цього треба вказати шлях до вихідного файлу, підпису та відкритого ключа (рис. 4).

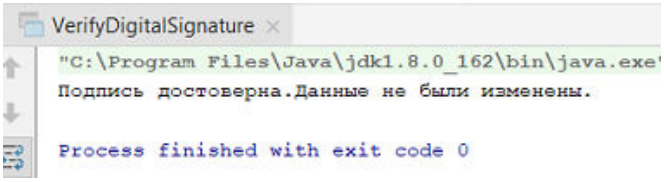

Рис. 4. Діалогове вікно програми після перевірки підпису

Підпис буде достовірний.

Хай буде змінено дані у текстовому файлі, як зображено на рис. 5. Та проведена повторна перевірка.

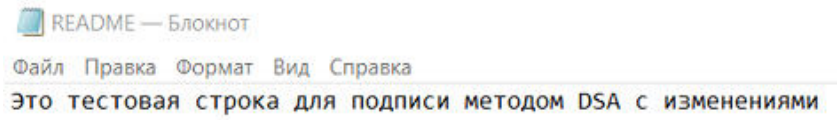

Рис. 5. Змінений зміст тестового файлу

Повторна перевірка підпису дає результати, представлені на рис. 6.

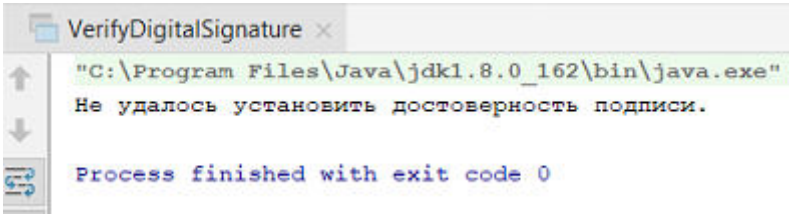

Рис. 6. Результат перевірки цифрового підпису після внесення змін у файл

Після зміни даних у файлі достовірність підпису була порушена.

Для дослідження швидкості та якісних характеристик алгоритмів створення та перевірки цифрового підпису, було розроблено окрема програма.

Для проведення дослідів використаємо System.nanoTime() та контрольні точки, які розставлені по ходу виконання програми. nanoTime() дає наносекундно точний час відносно деякої довільної точки. Повертає поточне значення найбільш точного доступного системного таймера в наносекундах.

Для того аби дізнатись який час необхідний на виконання тієї чи іншої ділянки коду, зробимо дві контрольні точки, де будемо вимірювати час (перед початком блоку коду та після нього). Для того аби дізнатись скільки часу знадобилось на виконання ділянки коду віднімемо від значення часу у кінцевій точці значення у початковій, та конвертуємо значення з наносекунд у мілісекунди.

Вимірювання будемо проводити на системі з Intel Core i7 (7^{th} gen), Nvidia GeForce GT960M та 12gb RAM, тому кількісні показники можуть відрізнятися але загальний принцип та залежності у алгоритмах ЕЦП зберігаються.

Дослідимо залежність швидкості виконання дій з створення ключової пари, виконання підпису документу та перевірки підпису. Розмір документу візьмемо у 100 байт (аби у подальшому перевірити залежність швидкодії від розміру документу). Для більшої точності результатів виконаємо 10000 симуляцій, як результат візьмемо середнє значення упродовж симуляцій.

Виконаємо дослід побудови цифрового електронного підпису для алгоритму DSA, результати якого представлено у табл.1 та рис. 7.

Таблиця 1

Залежність часу на виконання операцій від розміру ключа DSA

Алгоритм	Розмір ключа (біт)	Час необхідний на генерування підпису (мс)	Час необхідний на підписання документу (мс)	Час необхідний на перевірку підпису (мс)
DSA	512	409	4	8
DSA	1024	480	8	18
DSA	2048	351	13	33

Час на створення підпису при використанні хеш-функції SHA-1 при довжині ключа 512 біт – 409 мс, при довжині 1024 біт – 480 мс. А при використанні хеш-функції SHA-2 та довжини ключа у 2048 біт – 351 мс. Така динаміка часу, що при збільшенні довжини ключа та з тим надійності підпис створюється швидше, пов’язана перш за все з використанням більш сучасного алгоритму хешування.

Залежність часу від розміру ключа DSA представлено на рис. 8.

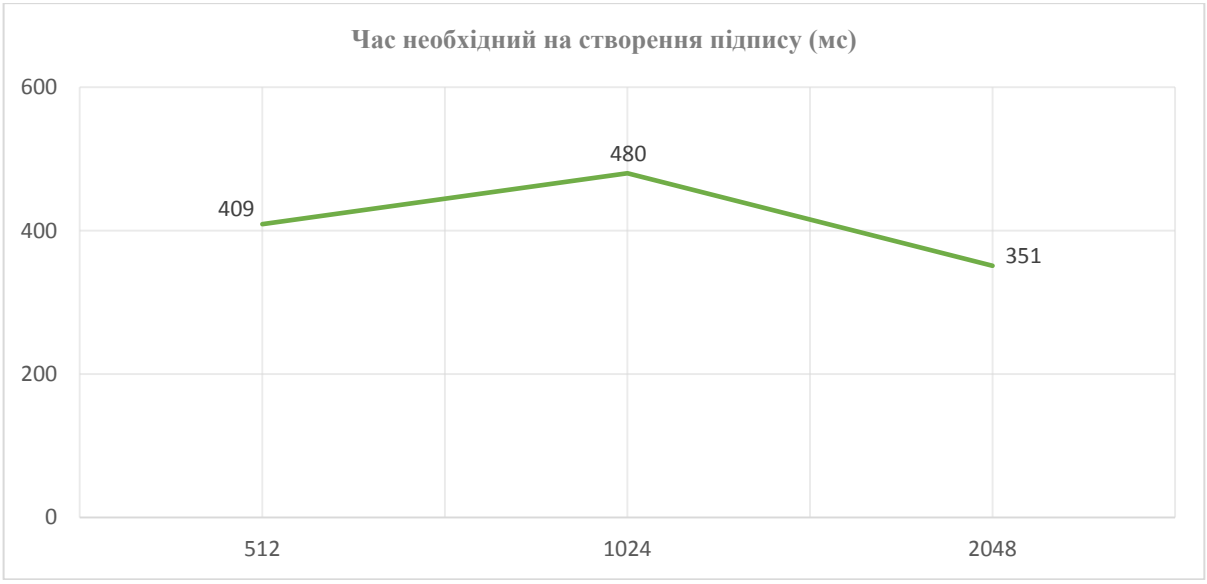

Рис. 7. Залежність часу на створення підпису від розміру ключа DSA

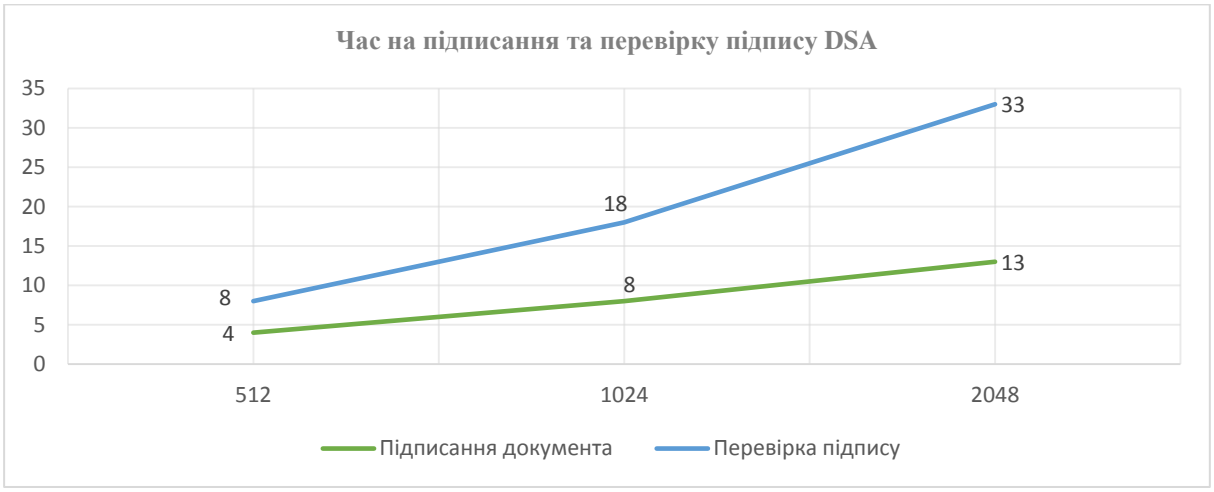

Рис. 8. Залежність часу на підписання документа та перевірку підпису від розміру ключа DSA

За результатами досліду зроблено висновок про лінійну залежність, яка показує що зі збільшенням довжини ключа збільшується час який необхідний на виконання операцій з підпису документа та перевірки цього підпису. Так, при довжині ключа у 512 біт для того аби поставити підпис під документом необхідно 4 мс, для перевірки – 8 мс. При розмірі ключа 1024 біт для підпису необхідно 8 мс, для перевірки підпису – 18 мс. Якщо розмір ключа 2048 біт, час який знадобиться на підпис документа 13 мс, для перевірки цього підпису – 33 мс.

Висновки

На мові програмування java розроблено програму, яка дозволяє генерувати ключові пари за алгоритмом DSA, виконувати підписання та перевірку підписаних цифровим підписом файлів. Роботу програми перевірено на тестових прикладах. Розроблена програма для визначення якісних характеристик алгоритмів створення та перевірки електронного цифрового підпису, знайдені залежності часу на створення підпису та часу на підписання документа і перевірку підпису від розміру ключа за алгоритмом DSA.

Список використаної літератури

1. Відомості про алгоритми ЕЦП. URL: http://cryptowiki.net/index.php?title=Алгоритмы_ЭЦП (дата звернення: 23.12.2019).
2. Электронно-цифровая подпись : алгоритм DSA. URL: http://www.volpi.ru/umkd/zki/examples/ds/ecp_dsa.html (дата звернення: 23.12.2019).
3. Мова програмування Java : Введення в Java. URL: https://metanit.com/java/tutorial/1.1.php (дата звернення: 23.12.2019).
4. Ігочкіна Є.В., Аналіз алгоритмів цифрового підпису. URL: http://www.security.ase.md/publ/ru/pubru86/ (дата звернення: 23.12.2019).
5. Керівництво по мові програмування Java. URL: https://metanit.com/java/tutorial/ (дата звернення: 23.12.2019).
6. Ткач Ю.М., Електронний цифровий підпис. URL: http://uchil.net/?cm=167737 (дата звернення: 23.12.2019).

References

1. Vidomosti pro alhorytmy ETsP. (Information about EDS algorithms) Available at: http://cryptowiki.net/index.php?title=Алгоритмы_ЭЦП (accessed 23 December 2019).
2. Elektronno-tsifrovaya podpis : algoritm DSA (Digital Signature: DSA Algorithm) Available at: http://www.volpi.ru/umkd/zki/examples/ds/ecp_dsa.html (accessed 23 December 2019).
3. Mova prohramuvannia Java : Vvedennia v Java (Java programming language: Introduction to Java) Available at: https://metanit.com/java/tutorial/1.1.php (accessed 23 December 2019).
4. Ihochkina Ye.V., Analiz alhorytmiv tsyfrovoho pidpysu (Analysis of digital signature algorithms) Available at: http://www.security.ase.md/publ/ru/pubru86/ (accessed 23 December 2019).
5. Kerivnytstvo po movi prohramuvannia Java (Java Programming Language Guide) Available at: https://metanit.com/java/tutorial/ (accessed 23 December 2019).
6. Tkach Yu.M., Elektronnyi tsyfrovyi pidpys (Electronic digital signature) Available at: http://uchil.net/?cm=167737 (accessed 23 December 2019).