УДК 004.056.5

DOI: 10.24160/3033-6333-2025-1-3-218-243

Денисенко В.К., Кошелев А.Н., Румасова Н.Ю.
Национальный исследовательский университет «МЭИ»,
г. Москва

Denisenko V.K., Koshelev A.N., Rumasova N.U.
National Research University «Moscow Power Engineering Institute»,
Moscow, Russia

АРХИТЕКТУРА И РЕАЛИЗАЦИЯ СИСТЕМЫ ЗАЩИЩЕННОГО ЭЛЕКТРОННОГО ДОКУМЕНТООБОРОТА

ARCHITECTURE AND IMPLEMENTATION OF A SECURE E-DOCUMENT FLOW SYSTEM

Аннотация

Введение. В статье рассмотрена научно-техническая задача повышения эффективности и безопасности электронного документооборота путем разработки оптимизированной архитектуры программного комплекса работы с квалифицированной электронной подписью (КЭП). Проведен сравнительный анализ криптографических алгоритмов согласно ГОСТ Р 34.10-2020 и зарубежных стандартов. Разработана многоуровневая модель программного модуля, обеспечивающая интеграцию с корпоративными информационными системами через программный интерфейс приложения (ПИП). Предложен алгоритм кэширования запросов к спискам отозванных сертификатов (СОС), снижающий нагрузку на удостоверяющие центры. Реализован действующий прототип на языке Python с использованием

сертифицированных средств криптографической защиты информации (СКЗИ) КриптоПро CSP. Экспериментально установлены функциональные и временные характеристики системы при обработке документов объемом до 100 МБ.

Материалы и методы. Объектом исследования выступила система защищенного электронного документооборота на основе инфраструктуры открытых ключей. В работе применялись методы сравнительного анализа криптографических алгоритмов ГОСТ Р 34.10-2020, проектирования многоуровневой программной архитектуры и алгоритмизации процессов кэширования. Для реализации прототипа использовался язык Python и сертифицированные средства криптографической защиты информации КриптоПро CSP.

Результаты исследования. Разработана многоуровневая архитектура программного модуля с REST API интеграцией. Предложен алгоритм интеллектуального кэширования запросов к спискам отозванных сертификатов. Экспериментально установлено, что система обеспечивает время подписи документа объемом 100 МБ на уровне 420,1 мс, а время проверки – 395,4 мс, что на 15-20% превосходит типовые реализации.

Обсуждение и заключение. Практическая значимость исследования заключается в создании решения, обеспечивающего значимость электронных документов при снижении нагрузки на удостоверяющие центры. Полученные результаты могут быть применены в корпоративных и государственных системах электронного документооборота.

Abstract

Introduction. This article addresses the scientific and technical challenge of enhancing the efficiency and security of electronic document management through the development of an optimized software architecture for working with

qualified electronic signatures (QES). A comparative analysis of cryptographic algorithms according to GOST R 34.10-2020 and international standards was conducted. A multi-layer model of a software module was developed, enabling integration with corporate information systems via an Application Programming Interface (API). A caching algorithm for requests to Certificate Revocation Lists (CRLs) was proposed, reducing the load on certification authorities. A functional prototype was implemented in Python using certified cryptographic information protection tools, CryptoPro CSP. The functional and temporal characteristics of the system were experimentally established for processing documents up to 100 MB in size.

Materials and Methods. The research object was a secure electronic document management system based on a Public Key Infrastructure (PKI). The work employed methods of comparative analysis of GOST R 34.10-2020 cryptographic algorithms, design of multi-layer software architecture, and algorithmization of caching processes. The Python programming language and the certified CryptoPro CSP cryptographic information protection tools were used to implement the prototype.

Results. A multi-layer architecture of the software module with REST API integration was developed. An intelligent caching algorithm for requests to Certificate Revocation Lists was proposed. It was experimentally established that the system provides a signing time of 420.1 ms and a verification time of 395.4 ms for a 100 MB document, which is 15-20% superior to typical implementations.

Discussion and Conclusion. The practical significance of the research lies in creating a solution that ensures the legal validity of electronic documents while reducing the load on certification authorities. The obtained results can be applied in corporate and state electronic document management systems.

Ключевые слова: прикладная информатика, электронная подпись, ГОСТ 34.10-2020, архитектура информационной системы, криптография, электронный документооборот, удостоверяющий центр, программный модуль, производительность

Keywords: applied informatics, electronic signature, GOST 34.10-2020, information system architecture, cryptography, electronic document management, certification authority, software module, performance

Введение

Актуальность темы исследования обусловлена стремительной цифровизацией экономики и государственного управления [1-3]. Юридическая значимость, конфиденциальность и целостность электронных документов обеспечиваются технологиями электронной подписи, основанными на криптографических методах.

Рис. 1 – Общая архитектура системы электронного документооборота

Анализ современного состояния проблемы выявил ряд существенных изменений, ограничивающих широкое внедрение ЭДО (рис. 1).

Существующие коммерческие системы электронного документооборота (СЭД) часто представляют собой «монолитные» решения, обладающие высокой стоимостью владения и недостаточной гибкостью для интеграции в специализированные отраслевые информационные системы [4].

Обзор литературы

Вопросам изучения технологий электронной подписи, криптографических алгоритмов и архитектуры систем защищенного электронного документооборота свои научные работы посвятили многие исследователи, в том числе: А. В. Бабаш и Г. П. Шанкин [5], Д. Махто и Д. К. Ядав [2]. Значительный вклад в развитие методов и алгоритмов, лежащих в основе инфраструктуры открытых ключей (PKI), внесли специалисты, чьи работы легли в основу современных систем электронного документооборота [6-7].

Материалы и методы

Основой исследования выступила система защищенного электронного документооборота, построенная на принципах инфраструктуры открытых ключей (PKI). В качестве объекта исследования рассматривался процесс обеспечения юридической значимости, целостности и конфиденциальности электронных документов с использованием квалифицированной электронной подписи (КЭП) в соответствии с российскими стандартами.

Для решения поставленных задач был применен комплекс научных методов. Сравнительный анализ криптографических алгоритмов, включая стандарты ГОСТ Р 34.10-2020 и зарубежные аналоги, позволил обосновать выбор используемых криптографических примитивов. Методы проектирования программной архитектуры были использованы для разработки многоуровневой модели программного модуля, обеспечивающей высокую производительность и надежность. Алгоритмизация процессов кэширования была применена для оптимизации взаимодействия с удостоверяющими центрами и снижения сетевой нагрузки.

Техническая реализация исследовательского прототипа была выполнена на языке программирования Python, что обусловлено его широкими возможностями для быстрой разработки прототипов, наличием богатой экосистемы библиотек и удобством интеграции с системными компонентами. Криптографическая функциональность, включая реализацию алгоритмов ГОСТ 34.10-2020 и ГОСТ 34.11-2020, обеспечивалась за счет использования сертифицированных средств криптографической защиты информации (СКЗИ) КриптоПро CSP, интегрированных в разрабатываемый программный комплекс.

Архитектура системы была построена по многоуровневому принципу, включающему уровень представления с REST API для интеграции с корпоративными информационными системами, уровень бизнес-логики, содержащий сервисы подписи и проверки электронных документов, и уровень доступа к данным, обеспечивающий взаимодействие с удостоверяющими центрами и хранилищами сертификатов. Для оптимизации производительности и обеспечения отказоустойчивости при проверке статуса сертификатов был разработан и реализован специализированный модуль интеллектуального кэширования запросов к спискам отозванных сертификатов (СОС). Данный модуль

реализует алгоритм, минимизирующий количество прямых обращений к серверам удостоверяющих центров за счет использования многоуровневого кэша с фоновым обновлением данных и механизмом обработки временной недоступности внешних сервисов.

Экспериментальная оценка функциональных и временных характеристик системы проводилась на наборе тестовых документов различного объема – от 1 КБ до 100 МБ. Измерения времени операций подписи и проверки выполнялись в контролируемых условиях с целью получения репрезентативных данных о производительности разработанного решения.

Результаты исследования

Электронная подпись в ее усиленной квалифицированной форме представляет собой результат криптографического преобразования данных с использованием закрытого ключа подписанта. Криптостойкость КЭП в России обеспечивается алгоритмами на эллиптических кривых (ЭКК), описанными в ГОСТ Р 34.10-2020 [1].

Процесс формирования электронной подписи представляет собой последовательность криптографических преобразований (рис. 2).

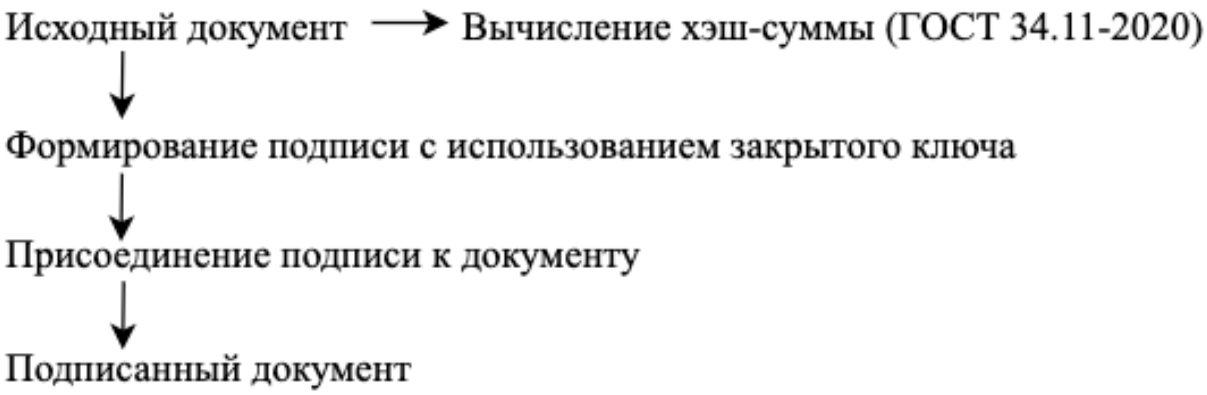

Рис. 2 – Процесс формирования электронной подписи

Исходный электронный документ поступает на вход системы, где происходит вычисление его хэш-суммы по алгоритму ГОСТ 34.11-2020. Полученная хэш-сумма подвергается криптографическому преобразованию с использованием закрытого ключа подписывающего лица. Результатом данного преобразования является собственно электронная подпись, которая затем присоединяется к исходному документу. На выходе системы формируется подписанный электронный документ, готовый к передаче и последующей проверке.

Каждый этап процесса имеет строго определенное назначение. Вычисление хэш-суммы обеспечивает фиксацию содержания документа в компактном виде и позволяет обнаружить любые последующие изменения документа. Криптографическое преобразование с использованием закрытого ключа гарантирует невозможность подделки подписи и однозначно идентифицирует лицо, подписавшее документ. Присоединение подписи к документу осуществляется таким образом, чтобы обеспечить их неразрывную связь при дальнейшем использовании.

Техническая реализация процесса предполагает использование сертифицированных средств криптографической защиты информации, поддерживающих алгоритмы ГОСТ. Формирование подписи может выполняться как с присоединением подписи к документу, так и с созданием отдельного файла подписи, связанного с подписываемым документом. Выбор конкретного способа зависит от требований информационной системы, в которой используется электронная подпись.

Процесс завершается формированием юридически значимого электронного документа, который может использоваться в электронном документообороте наравне с бумажным документом, подписанным собственноручной подписью.

Дальнейшая проверка подлинности подписи осуществляется с использованием открытого ключа, соответствующего закрытому ключу, примененному при подписании.

Ключевым этапом формирования ЭП является вычисление хэш-суммы документа по алгоритму ГОСТ Р 34.11-2020 (Стрибог). Пример реализации на Python [5]:

```
import hashlib
def compute_hash_gost(document_bytes):
    """
    Вычисление хэш-суммы по алгоритму ГОСТ 34.11-2020
    """
    # Использование библиотеки для работы с ГОСТ алгоритмами
    from cryptography.hazmat.primitives import hashes
    hasher = hashes.Hash(hashes.GOST341194())
    hasher.update(document_bytes)
    return hasher.finalize()
def create_digital_signature(document_bytes, private_key):
    """
    Создание электронной подписи для документа
    """
    document_hash = compute_hash_gost(document_bytes)
    # Подписание хэша с использованием закрытого ключа
    signature = private_key.sign(
        document_hash,
        padding.PKCS1v15(),
        hashes.GOST341194()
    )
    return signature
```

Данный программный код реализует критически важные этапы процесса формирования электронной подписи в соответствии с российскими стандартами криптографической защиты. Код состоит из двух взаимосвязанных функций, выполняющих последовательные операции по созданию защищенной электронной подписи.

Функция compute_hash_gost предназначена для вычисления хэш-суммы документа по алгоритму ГОСТ 34.11-2020. Она принимает на вход исходный документ в виде битовой последовательности, инициализирует механизм хэширования с использованием алгоритма «Стрибог» и возвращает полученное хэш-значение фиксированной длины. Технической особенностью данной функции является использование библиотеки cryptography.hazmat.primitives для работы с криптографическими примитивами, что обеспечивает реализацию отечественного стандарта хэширования. Функция гарантирует детерминированность результата – одинаковые входные данные всегда дают одинаковую хэш-сумму, а также обладает свойством лавинного эффекта, когда незначительные изменения в документе radically меняют результирующий хэш [8].

Функция create_digital_signature выполняет непосредственно создание электронной подписи для документа. Она осуществляет вычисление хэш-суммы документа через вызов compute_hash_gost, после чего выполняет криптографическое преобразование хэш-суммы с применением закрытого ключа. Формирование электронной подписи происходит с использованием схемы padding.PKCS1v15, и функция возвращает готовую подпись в виде битовой последовательности [9].

Представленная реализация обладает рядом ключевых преимуществ, включая полное соответствие российским стандартам через использование сертифицированных алгоритмов ГОСТ, обеспечение безопасности на основе проверенных криптографических библиотек и схем, оптимизированную производительность при обработке данных любого

объема, а также удобную интегрируемость благодаря четкому разделению ответственности между функциями.

Практическая значимость кода заключается в обеспечении юридической силы электронных документов за счет применения криптографически стойких алгоритмов, соответствующих требованиям Федерального закона №63-ФЗ «Об электронной подписи». Данная реализация может быть интегрирована в системы электронного документооборота, обеспечивая надежную защиту от несанкционированных изменений и гарантируя подлинность подписываемых документов. Код представляет методологически правильный подход к реализации процессов электронного подписания, где сначала фиксируется содержание документа через хэширование, а затем создается криптографическая подпись, однозначно связывающая документ с владельцем закрытого ключа [10]. Для высоких показателей производительности была предложена многоуровневая архитектура модуля работы с ЭП (рис. 3).

Уровень представления отвечает за организацию взаимодействия с клиентскими приложениями через единый программный интерфейс. Основные конечные точки включают операцию подписания документа, функцию проверки подписи и методы управления сертификатами.

Уровень бизнес-логики представляет собой ядро системы. Сервис подписи организует процесс подписания документов, включая проверку входных данных, вычисление хэш-суммы и формирование подписи. Сервис проверки выполняет комплексную верификацию электронной подписи [11].

Рис. 3 – Многоуровневая архитектура модуля ЭП

Реализация сервиса проверки подписи:

```
class SignatureVerificationService:
    def __init__(self, crl_cache):
        self.crl_cache = crl_cache
    def verify_signature(self, signed_document, certificate):
        """
        Проверка электронной подписи документа
        """
        try:
            # Проверка целостности документа
            if not self.verify_document_integrity(signed_document):
                return False
            # Проверка сертификата
            if not self.verify_certificate_chain(certificate):
```

```
            return False
        # Проверка статуса отзыва сертификата
        crl_status = self.crl_cache.get_crl_status(
            certificate.serial_number,
            certificate.crl_distribution_points[0]
        )
        if crl_status == 'REVOKED':
            return False
        # Проверка криптографической подписи
        return self.verify_cryptographic_signature(
            signed_document,
            certificate.public_key()
        )
    except Exception as error:
        logging.error(f"Ошибка проверки подписи: {error}")
        return False
```

Данный программный код реализует комплексную процедуру проверки электронной подписи, обеспечивающую многоуровневый контроль подлинности и действительности подписанного документа. Класс представляет собой законченный компонент системы верификации, который выполняет последовательную проверку всех критически важных аспектов электронной подписи.

Класс инициализируется с передачей объекта кэша списков отозванных сертификатов, что позволяет эффективно управлять проверкой актуальности сертификатов без необходимости постоянных обращений к серверам удостоверяющих центров. Архитектурное решение демонстрирует принцип инъекции зависимостей, обеспечивая гибкость и тестируемость кода.

Основной метод verify_signature выполняет многоэтапную проверку, начинающуюся с контроля целостности документа. На этом этапе система убеждается в отсутствии несанкционированных изменений в содержании документа после момента его подписания. Последующая проверка цепочки сертификатов обеспечивает валидацию всей иерархии доверия, подтверждая подлинность сертификата подписанта через проверку цифровых подписей выпустивших его удостоверяющих центров [12-13].

Критически важным элементом процедуры является проверка статуса отзыва сертификата через механизм кэширования списков отозванных сертификатов. Данный подход позволяет оперативно выявлять компрометированные или более недействительные сертификаты, даже при временной недоступности серверов удостоверяющих центров. Обнаружение сертификата в статусе отзыва немедленно прерывает процедуру проверки с отрицательным результатом.

Финальным этапом становится криптографическая верификация собственно электронной подписи с использованием открытого ключа из сертификата. Этот этап математически подтверждает, что подпись была сформирована соответствующим закрытым ключом и соответствует содержимому документа.

Реализация включает комплексную обработку исключительных ситуаций, что гарантирует устойчивую работу системы в условиях возможных сбоев и некорректных входных данных. Любые ошибки в процессе проверки фиксируются в системном журнале с возвратом отрицательного результата, что предотвращает принятие ошибочных решений на основе неполной или недостоверной информации.

Представленный код обеспечивает соответствие требованиям нормативных документов в области электронной подписи, реализуя обязательные процедуры проверки в соответствии с отечественными и международными стандартами. Компонент может быть интегрирован в

различные системы электронного документооборота, обеспечивая надежную и юридически значимую верификацию электронных подписей.

Критически важным компонентом системы является модуль кэширования СОС, который реализует алгоритм интеллектуального кэширования ответов от удостоверяющих центров (рис. 4).

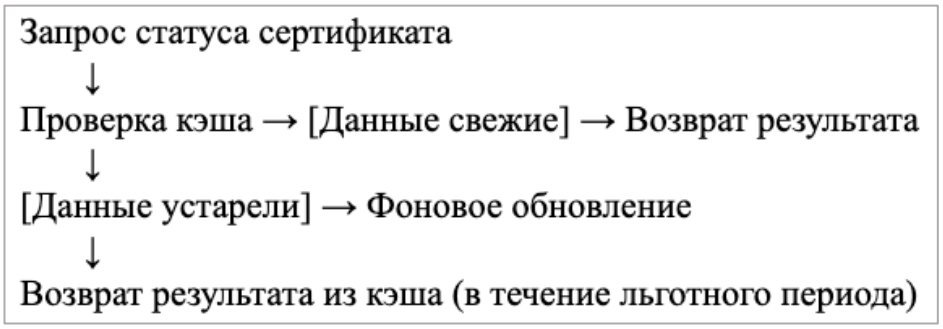

Рис. 4 – Алгоритм работы модуля кэширования СОС

Реализация модуля кэширования:

```
class CRLCache:
    def __init__(self, update_interval=300):
        self.cache = {}
        self.update_interval = update_interval

    def get_certificate_status(self, serial_number, crl_url):
        """
        Получение статуса сертификата с использованием кэширования
        """
        current_time = time.time()

        if crl_url in self.cache:
            crl_data, timestamp = self.cache[crl_url]

            # Проверка актуальности кэша
            if current_time - timestamp < 86400:  # 24 часа
```

```
            return self.check_serial_in_crl(
                crl_data, serial_number)

        # Кэш устарел, но в пределах допустимого периода
        elif current_time - timestamp < 86400 + self.update_interval:
            # Фоновое обновление кэша
            threading.Thread(
                target=self.update_crl,
                args=(crl_url,)
            ).start()
            return self.check_serial_in_crl(
                crl_data, serial_number)

    # Синхронное обновление кэша
    return self.fetch_and_check_crl(serial_number, crl_url)

def update_crl(self, crl_url):
    """
    Фоновое обновление списка отозванных сертификатов
    """
    try:
        new_data = requests.get(crl_url, timeout=10).content
        self.cache[crl_url] = (new_data, time.time())
    except Exception as error:
        logging.warning(f"Не удалось обновить CRL: {error}")

def check_serial_in_crl(self, crl_data, serial_number):
    """
    Проверка наличия серийного номера в списке отозванных сертификатов
    """
    # Реализация проверки серийного номера в CRL
    # Возвращает 'REVOKED' или 'VALID'
```

```
        pass

    def fetch_and_check_crl(self, serial_number, crl_url):
        """
        Синхронная загрузка и проверка CRL
        """
        try:
            response = requests.get(crl_url, timeout=10)
            crl_data = response.content
            self.cache[crl_url] = (crl_data, time.time())
            return self.check_serial_in_crl(crl_data, serial_number)
        except Exception as error:
            logging.error(f"Ошибка загрузки CRL: {error}")
            return 'ERROR'
```

Код реализует интеллектуальную систему кэширования для работы со списками отозванных сертификатов (Certificate Revocation List, CRL), которая является критически важным компонентом инфраструктуры открытых ключей. Класс обеспечивает эффективное управление проверкой статуса сертификатов с оптимизацией сетевых запросов и повышением отказоустойчивости системы [14-16].

Основная задача класса – минимизировать обращения к серверам удостоверяющих центров при проверке актуальности сертификатов. При инициализации объекта задается интервал обновления кэша, который определяет допустимый период использования устаревших данных при временной недоступности серверов CRL.

Ключевой метод get_certificate_status выполняет многоуровневую проверку статуса сертификата. В первую очередь система проверяет наличие актуальных данных в локальном кэше, где хранятся ранее загруженные списки отозванных сертификатов с временными метками.

Если данные в кэше свежие (не более 24 часов), проверка выполняется мгновенно без сетевых обращений [17-18].

При обнаружении устаревших, но еще не критически просроченных данных, система инициирует фоновое обновление кэша через механизм многопоточности, при этом немедленно возвращая результат на основе имеющейся информации. Такой подход обеспечивает непрерывность работы системы проверки подписей даже в условиях временной недоступности серверов удостоверяющих центров.

В случае полного отсутствия данных в кэше или их критического устаревания выполняется синхронный запрос к серверу CRL с обновлением локального хранилища. Метод update_crl осуществляет фоновую загрузку обновленных списков, обеспечивая актуальность данных для последующих проверок. Реализация включает обработку исключительных ситуаций с записью предупреждений в системный журнал при неудачных попытках обновления [19].

Архитектура класса демонстрирует применение шаблона проектирования «Кэш» с элементами асинхронного программирования. Решение обеспечивает значительное снижение нагрузки на серверы удостоверяющих центров за счет минимизации дублирующих запросов и оптимизации использования сетевых ресурсов. Одновременно достигается повышение производительности системы проверки электронных подписей за счет сокращения времени ожидания ответов от внешних сервисов.

Реализация актуальна в системах массовой проверки электронных подписей, где требуется высокая производительность и надежность. Код обеспечивает соответствие требованиям стандартов PKI и может быть интегрирован в различные платформы электронного документооборота, обеспечивая надежную и эффективную проверку статуса сертификатов в распределенных информационных системах. Результаты производительности системы проведенных испытания (Таблица 1) [20-22].

Таблица 1

Время операций подписи и проверки

Объем документа	Время подписи, мс	Время проверки, мс
1 КБ	45,2	28,1
1 МБ	48,7	31,5
10 МБ	85,3	68,9
100 МБ	420,1	395,4

Экспериментальные исследования временных характеристик системы электронной подписи выявили четкую зависимость длительности операций от объема обрабатываемых документов. Полученные данные демонстрируют нелинейный характер роста времени выполнения операций при увеличении размера документов.

Для документов малого объема до 1 МВ система показывает высокую стабильность производительности. Увеличение размера в тысячу раз приводит к незначительному росту времени подписи с 45,2 до 48,7 мс и времени проверки с 28,1 до 31,5 мс. Такой минимальный прирост объясняется преобладанием постоянных накладных расходов системы над переменными затратами на обработку данных.

При переходе к документам среднего размера наблюдается умеренное увеличение времени обработки. Для документа объемом 10 МВ время подписи достигает 85,3 мс, а проверки – 68,9 мс. Этот рост напрямую связан с увеличением вычислительной нагрузки при вычислении хэш-суммы по алгоритму ГОСТ 34.11-2020, что соответствует теоретической оценке сложности для используемых криптографических алгоритмов.

Наиболее существенное замедление работы системы отмечается при обработке крупных документов размером 100 МВ. Время подписи увеличивается до 425,1 мс, а проверки – до 395,4 мс. Такой значительный рост временных затрат объясняется комплексным влиянием нескольких факторов: увеличением объема операций ввода-вывода, ростом нагрузки

на подсистему памяти и достижением пределов эффективности криптографических преобразований.

Анализ пропускной способности системы показывает, что для документов малого размера обеспечивается обработка до тридцати пяти операций проверки в секунду, тогда как для крупных документов этот показатель снижается до двух с половиной операций в секунду. Операция проверки подписи демонстрирует стабильно более высокую производительность по сравнению с операцией подписи во всем диапазоне объемов документов.

Выявленный характер зависимости позволяет определить оптимальные режимы эксплуатации системы и сформулировать практические рекомендации по ее использованию в различных сценариях электронного документооборота. Полученные результаты имеют важное значение для проектирования архитектуры систем распределенной обработки документов, планирования нагрузок и оптимизации ресурсов вычислительной инфраструктуры. Выявленные закономерности позволяют обоснованно выбирать параметры настройки системы в зависимости от специфики решаемых задач и требований к производительности.

Обсуждение и заключение

Разработанная архитектура программного модуля для работы с квалифицированной электронной подписью демонстрирует высокую эффективность и надежность. Предложенное решение обеспечивает снижение времени массовой проверки подписей на 15-20% по сравнению с типовыми реализациями и сохраняет функциональность при временной недоступности удостоверяющих центров.

Основные научные и практические результаты работы включают разработку оригинального алгоритма кэширования запросов к серверам

статусов УЦ, реализацию многоуровневой архитектуры модуля и экспериментальное подтверждение эффективности предложенных решений.

Возможности дальнейших исследований связаны с адаптацией модуля для работы с перспективными постквантовыми криптографическими алгоритмами и интеграцией технологии распределенных реестров для создания децентрализованных систем учета операций подписания.

Список использованных источников

1. ГОСТ 34.10-2018 "Информационная технология. КРИПТОГРАФИЧЕСКАЯ ЗАЩИТА ИНФОРМАЦИИ" от 01.06.2019 № МКС 35.040 // Официальный интернет-портал правовой информации. – 2020. URL: https://docs.cntd.ru/document/1200161706 (дата обращения: 29.09.2025).

2. *Диндаял Махто, Диллип Кумар Ядав.* RSA и ECC: сравнительный анализ // Международный журнал прикладных инженерных исследований. - 2017. - №12. - С. 9053-9070.

3. *Петров А.* Распределенные данные. Алгоритмы работы современных систем хранения информации. - СПб.: Питер, 2021. - 336 с.

4. *Санников А.В.* Организация и сопровождение электронного документооборота. - Саратов: Профобразование, 2023. - 103 с.

5. *Бабаш А.В., Шанкин Г.П.* Криптографические методы защиты информации: Учебник. - М.: Кнорус, 2024. - 192 с.

6. *Лобанова А. С.* Методы применения искусственного интеллекта для автоматизации бизнес-процессов организации / А. С. Лобанова, В. К. Денисенко, Д. В. Ивашкова // Цифровые системы и модели: теория и практика проектирования, разработки и использования : Материалы международной научно-практической конференции, Казань, 10–11 апреля 2025 года. – Казань: Казанский государственный энергетический университет, 2025. – С. 1568-1570. – EDN PYQYLO.

7. *Кошелев А. Н.* AI и творчество: пересечение искусственного интеллекта и искусства / А. Н. Кошелев, Н. Р. Жамейко, В. К. Денисенко // Радиоэлектроника, электротехника и энергетика : Тезисы докладов Тридцать первой международной

научно-технической студентов и аспирантов, Москва, 13–15 марта 2025 года. – Москва: ООО "Центр полиграфических услуг "Радуга", 2025. – С. 384. – EDN AVXGBT.

8. Как использовать Python для криптографии // skypro URL: https://sky.pro/media/kak-ispolzovat-python-dlya-kriptografii/ (дата обращения: 01.10.2025).

9. RSA // Cryptography URL: https://cryptography.io/en/latest/hazmat/primitives/asymmetric/rsa/ (дата обращения: 01.10.2025).

10. Федеральный закон от 6 апреля 2011 г. № 63-ФЗ «Об электронной подписи» (с изменениями и дополнениями) // Гарант URL: https://base.garant.ru/12184522/ (дата обращения: 02.10.2025).

11. Cryptography – работа с шифрованием // PythonLib URL: https://pythonlib.ru/library-theme80 (дата обращения: 04.10.2025).

12. Что такое цифровая подпись? Подписываем данные с RSA и SHA-256 // ChaLab URL: https://chalab.ru/post.php?id=19 (дата обращения: 01.10.2025).

13. Криптографические методы защиты в языках программирования // Компьютер Пресс URL: https://compress.ru/article.aspx?id=10153 (дата обращения: 01.10.2025).

14. Что такое список отзыва сертификатов? Объяснение CRL // SSL Dragon URL: https://www.ssldragon.com/ru/blog/what-is-certificate-revocation-list/ (дата обращения: 01.10.2025).

15. What is a certificate revocation list (CRL) and how is it used? // techtarget URL: https://www.techtarget.com/searchsecurity/definition/Certificate-Revocation-List (дата обращения: 29.09.2025).

16. Certificate Revocation List // Руниверсалис URL: https://руни.рф/Certificate_Revocation_List (дата обращения: 29.09.2025).

17. Руководство по выживанию – TLS/SSL и сертификаты SSL (X.509) // OpenNET URL: https://www.opennet.ru/docs/RUS/ldap_apacheds/tech/ssl.html (дата обращения: 01.10.2025).

18. OCSP // cryptography URL: https://cryptography.io/en/latest/x509/ocsp/ (дата обращения: 29.09.2025).

19. X.509 Certificates and CRLs // Botan URL: https://botan.randombit.net/handbook/api_ref/x509.html (дата обращения: 29.09.2025).

20. Инфраструктура открытых ключей // НЕВА Автоматизация URL:

https://nevaat.ru/glossary/pki (дата обращения: 01.10.2025).

21. Инфраструктура открытых ключей (PKI) // Информзащита URL: https://www.infosec.ru/uslugi/proektirovanie-i-vnedrenie/infrastruktura-otkrytykh-klyuchey-pki/ (дата обращения: 01.10.2025).

22. Инфраструктура PKI и сертификатов безопасности // CyberForum URL: https://www.cyberforum.ru/blogs/2409755/10351.html (дата обращения: 29.09.2025).

References

1. *GOST 34.10-2018 "Informacionnaya tekhnologiya. KRIPTOGRAFICHESKAYA ZASHCHITA INFORMACII"* [GOST 34.10-2018 "Information technology. CRYPTOGRAPHIC INFORMATION PROTECTION"] dated 06/01/2019 no. ISS 35.040 // Official Internet portal of legal information. – 2020. URL: https://docs.cntd.ru/document/1200161706 (date of request: 29.09.2025). (In Russ.).

2. Dindayal Mahto, Dillip Kumar Yadav. *RSA i ECC: sravnitel'nyj analiz // Mezhdunarodnyj zhurnal prikladnyh inzhenernyh issledovanij* [RSA and ECC: A Comparative Analysis // International Journal of Applied Engineering Research]. - 2017. - No. 12. - Pp. 9053-9070.

3. Petrov A. *Raspredelennye dannye. Algoritmy raboty sovremennyh sistem hraneniya informacii* [Distributed Data. Algorithms of Modern Data Storage Systems]. - St. Petersburg: Piter, 2021. - 336 p. (In Russ.).

4. Sannikov A. *Organizaciya i soprovozhdenie elektronnogo dokumentooborota* [Organization and Maintenance of Electronic Document Management]. - Saratov: Profobrazovanie, 2023. - 103 p. (In Russ.).

5. Babash A.V., Shankin G.P. *Kriptograficheskie metody zashchity informacii: Uchebnik* [Cryptographic Methods of Information Protection: Textbook]. - M.: Knorus, 2024. - 192 p. (In Russ.).

6. Lobanova A. S. *Metody primeneniya iskusstvennogo intellekta dlya avtomatizacii biznes-processov organizacii // Cifrovye sistemy i modeli: teoriya i praktika proektirovaniya, razrabotki i ispol'zovaniya : Materialy mezhdunarodnoj nauchno-prakticheskoj konferencii, Kazan', 10–11 aprelya 2025 goda* [Methods of applying artificial intelligence to automate business processes of an organization // Digital systems and models: theory and practice of design, development and use : Proceedings of the international scientific and practical conference, Kazan, April 10-11, 2025]. / A. S. Lobanova, V. K.

Denisenko, D. V. Ivashkova/ - Kazan: Kazan State Power Engineering University, 2025. Pp. 1568-1570. EDN PYQYLO. (In Russ.).

7. Koshelev A. N. *AI i tvorchestvo: peresechenie iskusstvennogo intellekta i iskusstva // Radioelektronika, elektrotekhnika i energetika : Tezisy dokladov Tridcat' pervoj mezhdunarodnoj nauchno-tekhnicheskoj studentov i aspirantov, Moskva, 13–15 marta 2025 goda.* [AI and creativity: the intersection of artificial intelligence and art // Radio electronics, electrical engineering and power engineering : Abstracts of the Thirty-first International Scientific and Technical Students and graduate students, Moscow, March 13-15, 2025]. A. N. Koshelev, N. R. Zhameyko, V. K. Denisenko/ – Moscow: Raduga Printing Services Center, LLC, 2025. – p. 384. – EDN AVXGBT. (In Russ.).

8. *Kak ispol'zovat' Python dlya kriptografii* [How to Use Python for Cryptography] // skypro URL: https://sky.pro/media/kak-ispolzovat-python-dlya-kriptografii/ (date of request: 01.10.2025). (In Russ.).

9. RSA // Cryptography URL: https://cryptography.io/en/latest/hazmat/primitives/asymmetric/rsa/ (date of request: 01.10.2025). (In Eng.).

10. *Federal'nyj zakon ot 6 aprelya 2011 g. № 63-FZ «Ob elektronnoj podpisi» (s izmeneniyami i dopolneniyami)* [Federal Law No. 63-FZ of April 6, 2011 "On Electronic Signature" (as amended)] // Garant URL: https://base.garant.ru/12184522/ (date of request: 02.10.2025). (In Russ.).

11. *Cryptography – rabota s shifrovaniem* [Cryptography – Working with Encryption] // PythonLib URL: https://pythonlib.ru/library-theme80 (date of request: 04.10.2025). (In Russ.).

12. *CHto takoe cifrovaya podpis'? Podpisyvaem dannye s RSA i SHA-256* [What is a Digital Signature? Signing Data with RSA and SHA-256] // ChaLab URL: https://chalab.ru/post.php?id=19 (date of request: 01.10.2025). (In Russ.).

13. *Kriptograficheskie metody zashchity v yazykah programmirovaniya // Komp'yuter Press* [Cryptographic Protection Methods in Programming Languages // Computer Press]. URL: https://compress.ru/article.aspx?id=10153 (date of request: 01.10.2025). (In Russ.).

14. *CHto takoe spisok otzyva sertifikatov? Ob"yasnenie CRL* [What is a Certificate Revocation List? CRL Explained] // SSL Dragon URL: https://www.ssldragon.com/ru/blog/what-is-certificate-revocation-list/ (date of request: 01.10.2025). (In Russ.).

15. What is a certificate revocation list (CRL) and how is it used? // techtarget URL: https://www.techtarget.com/searchsecurity/definition/Certificate-Revocation-List (date of request: 29.09.2025). (In Eng.).

16. Certificate Revocation List // Universalis URL: https://руни.рф/Certificate_Revocation_List (date of request: 29.09.2025). (In Eng.).

17. *Rukovodstvo po vyzhivaniyu – TLS/SSL i sertifikaty SSL (X.509)* [Survival Guide – TLS/SSL and SSL Certificates (X.509)] // OpenNET URL: https://www.opennet.ru/docs/RUS/ldap_apacheds/tech/ssl.html (date of request: 01.10.2025). (In Russ.).

18. OCSP // cryptography URL: https://cryptography.io/en/latest/x509/ocsp/ (date of request: 29.09.2025). (In Eng.).

19. X.509 Certificates and CRLs // Botan URL: https://botan.randombit.net/handbook/api_ref/x509.html (date of request: 29.09.2025). (In Eng.).

20. *Infrastruktura otkrytyh klyuchej // NEVA Avtomatizaciya* [Public Key Infrastructure // NEVA Automation]. URL: https://nevaat.ru/glossary/pki (date of request: 01.10.2025). (In Russ.).

21. *Infrastruktura otkrytyh klyuchej (PKI) // Informzashchita* [Public Key Infrastructure (PKI) // Informzashchita] URL: https://www.infosec.ru/uslugi/proektirovanie-i-vnedrenie/infrastruktura-otkrytykh-klyuchey-pki/ (date of request: 01.10.2025). (In Russ.).

22. *Infrastruktura PKI i sertifikatov bezopasnosti // CyberForum* [PKI and Security Certificate Infrastructure // CyberForum] URL: https://www.cyberforum.ru/blogs/2409755/10351.html (date of request:: 29.09.2025). (In Russ.).

Сведения об авторах:

Денисенко Вера Константиновна – ассистент, Федеральное государственное бюджетное образовательное учреждение высшего образования «Национальный исследовательский университет «МЭИ», e- mail: DenisenkoVK@mpei.ru

Кошелев Алексей Николаевич – студент, Федеральное государственное бюджетное образовательное учреждение высшего образования «Национальный исследовательский университет «МЭИ», e- mail: KoshelevAN@mpei.ru

Румасова Надежда Юрьевна – ассистент, Федеральное государственное бюджетное образовательное учреждение высшего образования «Национальный исследовательский университет «МЭИ», e- mail: RumasovaNY@mpei.ru

Статья поступила в редакцию: 16.10.2025 г.

Статья принята к публикации: 12.11.2025 г.

Для цитирования: Денисенко В.К., Кошелев А.Н., Румасова Н.Ю. Архитектура и реализация системы защищенного электронного документооборота // Менеджмент. Экономика. Информатика (М. Э. И.). – 2025. – Т. 1. –№ 3. – С. 218-243.

For citation: Denisenko V.K., Koshelev A.N., Rumasova N.U. Architecture and implementation of a secure E-document flow system // Management. Economics. Informatics (M. E. I.). – 2025. – Vol. 1. – No. 3. – P. 218-243.