Цифровые подписи в ограниченных средах: Сравнительный анализ безопасности RSA, ECDSA и EdDSA

Наурас Х. Сабри[1]

[1]Санкт-Петербругский государственный электротехнический университет «ЛЭТИ», Россия, 197022, г. Санкт-Петербург

Аннотация. Данное исследование направлено на выявление оптимального алгоритма цифровой подписи, сочетающего высокий уровень безопасности с эффективным использованием вычислительных ресурсов, что важно для таких устройств как микроконтроллеры, которые работают с ограниченными объёмами вычислительных ресурсов и памяти. В работе приведён сравнительный анализ трёх алгоритмов цифровой подписи – RSA, ECDSA и EdDSA – с точки зрения их применимости во встраиваемых системах и устройствах IoT. Методология исследования включала математический анализ и практическую реализацию рассматриваемых алгоритмов на языке программирования Python с последующим тестированием производительности по ключевым параметрам: времени генерации ключей, скорости подписания и верификации, а также объёму потребляемой памяти. Полученные результаты демонстрируют, что алгоритм EdDSA обеспечивает наилучший баланс между производительностью и безопасностью, что делает его наиболее предпочтительным выбором для систем на базе микроконтроллеров. ECDSA, несмотря на высокую производительность, является менее эффективным и обладает известными уязвимостями, связанными с генерацией случайных чисел (nonce). RSA, несмотря на высокую скорость верификации, требует значительных вычислительных ресурсов, что ограничивает его применение в устройствах с жёсткими ограничениями памяти и производительности. Полученные результаты предоставляют разработчикам встраиваемых систем чёткие критерии выбора алгоритмов цифровой подписи, учитывающие как требования безопасности, так и специфику работы в условиях ограниченных ресурсов. Практическая значимость исследования заключается в оптимизации архитектуры защищённых IoT-решений, где эффективное использование вычислительных мощностей является критически важным фактором.

Ключевые слова: цифровая подпись, микроконтроллер, устройства с ограниченными ресурсами, RSA, ECDSA, EdDSA.

Abstract. This study aims to identify the optimal digital signature algorithm that combines a high level of security with efficient use of computational resources, which is crucial for devices such as microcontrollers operating with limited processing power and memory. The paper presents a comparative analysis of three digital signature algorithms – RSA, ECDSA, and EdDSA – regarding their applicability in embedded systems and IoT devices. The research methodology included mathematical analysis and practical implementation of the algorithms using Python programming language, followed by performance testing of key parameters: key generation time, signing and verification speed, and memory consumption. The results demonstrate that the EdDSA algorithm provides the best balance between performance and security, making it the preferred choice for microcontroller-based systems. While ECDSA shows strong performance, it’s slightly

less efficient and has known vulnerabilities related to nonce generation. Despite its fast verification speed, RSA requires substantial computational resources, limiting its use in devices with strict memory and performance constraints. The findings provide embedded systems developers with clear criteria for selecting digital signature algorithms that consider both security requirements and the specifics of operating in resource-constrained environments. The practical significance of this research lies in optimizing the architecture of secure IoT solutions where efficient use of computational resources is a critical factor.

Keywords: digiral signature, microcontroller, constrained resources devices, RSA, ECDSA, EdDSA.

Введение

С момента появления в 1970-х годах криптографии с открытым ключом, цифровые подписи стали неотъемлемым элементом безопасной коммуникации [1]. Основы этого направления были заложены в 1976 году работами Диффи и Хеллмана по асимметричному шифрованию, а уже в 1977 году Ривест, Шамир и Адлеман представили миру алгоритм RSA [2]. Данные фундаментальные разработки создали основу для современных систем цифровой подписи, ставших обязательным компонентом защиты информации, передаваемой в электронном виде. С распространением компактных вычислительных устройств особую актуальность приобрела задача адаптации криптографических методов к условиям ограниченных ресурсов, характерных для микроконтроллерных систем. Жёсткие ограничения по вычислительной мощности, объёму памяти и энергопотреблению, присущие микроконтроллерам, широко используемых во встраиваемых системах и IoT-устройствах, создают значительные сложности при реализации криптографических протоколов, требуя особого подхода к выбору и оптимизации алгоритмов цифровой подписи.

Реализация алгоритмов цифровой подписи в микроконтроллерах сталкивается со значительными трудностями из-за их ограниченных вычислительных возможностей и малого объёма памяти [3]. Традиционные схемы, такие как RSA, как правило демонстрируют низкую эффективность в таких средах из-за высокой вычислительной сложности и требовательности к ресурсам. В частности, RSA требует использования длинных ключей и выполнения ресурсоёмких операций модульного возведения в степень, что часто превышает возможности микроконтроллеров. Таким образом, поиск и рассмотрение альтернативных криптографических алгоритмов, которые сохраняют необходимый уровень безопасности при оптимизированном потреблении ресурсов, является крайне актуальным и создаёт сложную задачу нахождения баланса между защитой данных и техническими требованиями устройств с ограниченным энергопотреблением.

Несмотря на существующие проблемы, цифровые подписи остаются критически важным инструментом обеспечения безопасности и целостности коммуникаций в микроконтроллерных системах. В различных приложениях – от медицинского оборудования до автомобильных систем и интеллектуальных счётчиков – важно подтверждать подлинность команд и верифицировать данные. Механизмы цифровой подписи позволяют микроконтроллерам достоверно устанавливать неизменность сообщений и подтверждать их поступление из доверенных источников, что является фундаментальным требованием для обеспечения надёжности систем, где даже незначительные ошибки могут привести к серьёзным последствиям.

В данной работе особый исследовательский интерес представляют три алгоритма цифровой подписи, которые актуальны для реализации в микроконтроллерах: RSA [4], ECDSA [5] и EdDSA [6]. Важно отметить, что хотя алгоритмы хэширования эффективно обеспечивают целостность данных, они не решают задачи аутентификации источника информации – если в данные были внесены изменения, то хэш можно пересчитать, однако без секретного ключа невозможно связать конкретный хэш с конкретным отправителем. Цифровые подписи, в свою очередь, сочетая хэширование с асимметричным шифрованием, гарантируют как целостность, так и подлинность данных. Использование закрытого ключа в данном случае позволяет подтвердить, что конкретную подпись мог создать только

уполномоченный пользователь, обеспечивая тем самым безопасность, которую не может гарантировать только хэширование.

Основная цель настоящего исследования заключается в выявлении оптимального алгоритма цифровой подписи для сред с ограниченными ресурсами, в частости, для микроконтроллеров, который бы обеспечивал наилучший баланс между эффективностью и безопасностью. Исследование сосредоточено на сравнительном анализе производительности алгоритмов RSA, ECDSA, EdDSA с точки зрения скорости выполнения операций, потребления ресурсов и уровня предоставленной защиты, что позволяет определить наиболее подходящее решение для конкретных условий применения.

Актуальность настоящего исследования определяется растущей потребностью в обеспечении безопасности микроконтроллерных систем без ущерба для их производительности. Широкое внедрение микроконтроллеров в критически важные приложения делает принципиально важным глубокое понимание компромиссных решений при выборе используемых алгоритмов. Методологическая основа исследования включает комплексный математический анализ эффективности алгоритмов цифровой подписи в условиях ограниченных ресурсов с последующей практической реализацией на языке Python и детальной оценкой полученных результатов. Весь исходный код и результаты экспериментов доступны в открытом репозитории GitHub по адресу https://github.com/NawrasHusseinSabbry/Digital_Signature.

Данное исследование разделено на четыре смысловых блока: вводная часть, раскрывающая проблематику исследования; теоретический раздел, посвящённый фундаментальным принципам анализируемых алгоритмов цифровой подписи; аналитический раздел с результатами математического моделирования и программной реализации; заключительная часть, содержащая основные выводы и практические рекомендации по применению полученных результатов.

1. Основы анализа цифровых подписей

1.1. Цифровая подпись RSA

Криптосистема Ривеста-Шамира-Адлемана (RSA) обеспечивает аутентификацию и целостность сообщений через механизм ассиметричного шифрования с помощью системы открытых ключей. Безопасность RSA основана на вычислительной сложности решения задачи модульного возведения в степень. В этом процессе отправитель создаёт цифровую подпись, зашифровывая хэш сообщения своим закрытым ключом. Получатель может проверить подлинность подписи с помощью открытого ключа отправителя. Размеры ключей криптосистемы RSA обычно варьируются от 1014 до 4096 бит, при этом 2048-битные ключи считаются безопасными в настоящее время – такой размер ключа обеспечивает экспоненциально более высокую стойкость к атакам факторизации по сравнению с 1024-битным ключом. Считается, что взлом 2048-битного ключа потребует в четыре миллиарда раз больше времени, что обеспечивает его стойкость, благодаря которой, согласно прогнозам, данные ключи сохранят свою криптостойкость до 2040-2050 годов, что делает их предпочтительным выбором для безопасной передачи данных [7].

Методика реализации RSA для цифровых подписей включает несколько этапов [8]:

1) Генерация ключевой пары открытого и закрытого ключей с помощью следующей процедуры:
   а) Выбор двух больших простых чисел p и q;

b) Вычисление модуля n как произведения p и q, то есть $n = p \times q$;
c) Определение функции $\varphi(n)$, где $\varphi(n) = (p-1)(q-1)$;
d) Выбор открытой экспоненты e такой, что $1 < e < \varphi(n)$ и наибольший общий делитель $\gcd(e, \varphi(n)) = 1$;
e) Вычисление закрытой экспоненты d, обратной e по модулю $\varphi(n)$, гарантируя, что $(d \times e) = 1 \left(mod\ \varphi(n)\right)$.

Полученный открытый ключ состоит из модуля n и открытой экспоненты e, а закрытый ключ – из модуля n и закрытой экспоненты d.

2) Подписание: чтобы подписать сообщение M, отправитель вычисляет хэш сообщения $h = hash(M)$ с помощью криптографической хэш-функции, после чего генерирует подпись s как $s = h^d\ (mod\ n)$, где d – закрытая экспонента из пары ключей отправителя.

3) Проверка: чтобы верифицировать подпись s, получатель вычисляет значение исходного хэша h с использованием подписи s и открытого ключа отправителя $\tilde{h} = s^e\ (mod\ n)$. Если h эквивалентно $\tilde{h}$, то подпись считается действительной, в противном случае – недействительной.

Безопасность механизма цифровой подписи RSA напрямую зависит от надёжности процесса генерации ключевой пары и криптографической стойкости используемой хэш-функции. Критически важными параметрами являются размеры простых чисел p и q, которые должны быть достаточно большими для противодействия атакам факторизации, а также правильный выбор открытой экспоненты e – случайной величины достаточной величины. Применяемая хэш-функция должна обладать устойчивостью к коллизиям, исключая возможность нахождения двух различных сообщений с идентичным хэш-значением. Кроме того, схема подписи должна быть защищена от атак повторного использования и других потенциальных уязвимостей.

В рамках исследования для анализа схемы цифровой подписи RSA была выбрана библиотека PyCryptodome, реализованная на языке программирования Python. Данный выбор обусловлен сочетанием простоты использования, гибкости функционала и минимальными требованиями к зависимостям, что особенно важно для ограниченных в ресурсах сред, включая микроконтроллерные системы. PyCryptodome представляет собой автономную криптографическую библиотеку, предлагающую широкий спектр примитивов, включая реализацию RSA, алгоритм хэширования SHA-256 и схему дополнения PKCS#1 V1.5. Ключевыми преимуществами библиотеки являются отсутствие внешних зависимостей и оптимизированное потребление ресурсов, что соответствует строгим требованиям встраиваемых систем.

Экспериментальная реализация включает основные операции: генерацию 2048-битной ключевой пары RSA, хэширование сообщения с использованием SHA-256, создание подписи с применением закрытого ключа и последующую верификацию через открытый ключ. В процессе выполнения фиксируются ключевые метрики производительности – время выполнения операций и объём потребляемой памяти. Эти данные важны для оценки применимости алгоритма RSA в условиях ограниченных ресурсов, характерных для микроконтроллерных платформ. Данная реализация предоставляет ценные практические сведения о реальных требованиях к вычислительным ресурсам при использовании цифровых подписей RSA, что позволяет делать обоснованные выводы о возможности их интеграции в различные встраиваемые системы.

1.2. Алгоритм цифровой подписи на эллиптической кривой (ECDSA)

ECDSA – это криптографический метод, используемый для создания безопасных цифровых подписей, в основе которого лежит криптография эллиптических кривых (ECC) и являющийся одной из версии алгоритма цифровой подписи (DSA). ECDSA демонстрирует существенные преимущества по сравнению с традиционными решениями, такими как RSA, предлагая сопоставимый уровень безопасности при значительно меньших размерах ключей и подписей – данная особенность делает этот алгоритм особенно привлекательным для применения в ситуациях, когда вычислительная мощность и пространство ограничены как, например, во встроенных системах [9].

Алгоритмы генерации и проверки подписи в ECDSA описывается следующим образом [10]. Процесс работы начинается с согласования параметров эллиптической кривой (далее ЭК), а именно (E, G, n), между отправителем и получателем, где E – уравнение ЭК, G – базовая точка ЭК (точка на кривой, порождающая подгруппу простого порядка, n – целочисленный порядок G, где $nG = O$ (O – нейтральный элемент). Отправитель (далее обозначается как «Алиса» или буквой «A») использует закрытый ключ d_A – случайное число в диапазоне $[1, n-1]$, и соответствующий открытый ключ $Q_A = d_A G$ для подписания сообщения m. Чтобы подписать сообщение, Алиса выполняет следующие действия:

1) Вычисление $e = H(m)$;
2) Выбор случайного целого числа k из диапазона $[1, n-1]$;
3) Вычисление точки кривой $(x_1, y_1) = k \times G$;
4) Определение $r \equiv x_1 \ (mod\ n)$;
5) Формирование $s = k^{-1}(e + rd_A)$ по модулю n.

В результате, сформированной подписью является пара (r, s), которую Алиса, вместе с сообщением m и открытым ключом Q_A, отправляет получателю (как правило обозначается как «Боб»). Важным аспектом является необходимость использования уникального значения k для каждой подписи во избежание компрометации закрытого ключа.

Процедура верификации подписи получателем включает следующие шаги:

1) Вычисление $e = H(m)$;
2) Определение величин $u_1 \equiv es^{-1} \ (mod\ n)$ и $u_2 \equiv rs^{-1} \ (mod\ n)$;
3) Вычисление точки $(x_1, y_1) = u_1 G + u_2 Q_A$.

Подпись Алисы считается действительной тогда и только тогда, когда $(x_1, y_1) \equiv kG$.

В рамках исследования для анализа реализации схемы ECDSA использовался код на языке Python, представляющий собой алгоритм, использующий эллиптическую кривую secp256k1 в проективных координатах и включающий процессы генерации ключа, подписания сообщения и проверки подписи.

На этапе формирования ключа случайным образом генерируется 256-битный закрытый ключ, а соответствующий открытый ключ получается путём скалярного умножения с применением лестничного метода Монтгомери – данный метод, за счёт дополнительных шагов сложения и удвоения в операции умножения точек, обеспечивает не только вычислительную эффективность за счёт последовательной реализации, но и устойчивость к атакам по побочным каналам [11].

При реализации ECDSA хэш сообщения вычисляется с использованием алгоритма SHA-256, при этом дополнительно генерируется случайный параметр (nonce). Процесс верификации подписи включает пересчёт точки эллиптической кривой с использованием

метода double-and-add («сложение-и-удвоение), который был выбран вместо более сложного лестничного метода Монтгомери, имеющего большую вычислительную сложность [11]. Такой выбор оправдан тем, что на этапе верификации работа происходит с открытыми параметрами, что снижает требования к защите от атак по побочным каналам.

Использование проективных координат в коде повышает производительность за счёт исключения операции инверсии при умножении точек ЭК, что особенно важно в средах с ограниченными вычислительными ресурсами и памятью, где эффективность вычислений и оптимизация использования памяти являются критическими факторами. Однако следует отметить, что ECDSA сохраняет уязвимость к атакам с повторным использованием nonce, как подробно описано в [10].

1.3. Алгоритм цифровой подписи на кривой Эдвардса (EdDSA)

EdDSA – это быстрая и безопасная схема цифровой подписи, основанная на модификации схемы Шнорра для витых кривых Эдвардса, таких как Ed25519. Разработанный группой специалистов под руководством Дэниела Дж. Бернштейн, данный алгоритм сочетает высокую производительность с надёжной защитой, будучи при этом детерминированным – для одного и того же сообщения и ключа существует последовательность подписей [12]. Процессы генерации и проверки подписи в EdDSA определяются следующим образом [10].

Для подписания сообщения m, Алисе необходимо:

1) Выбрать закрытый ключ (секретный ключ S_k) и вычислить открытый ключ P_k=$S_k G$;
2) Вычислить $h = H(S_k)$;
3) Рассчитать nonce r путём конкатенации (операции ‖) h с сообщением m и вычислить для них хэш-значение $r = H(h \parallel m)$;
4) Вычислить $R = x$-координату (rG);
5) Вычислить $s = r + H(r \parallel P_k \parallel m)S_k$.

Подписью является пара (R, s), которую Алиса отправляет Бобу вместе с m и P_k.

Для проверки подписи Алисы Боб выполняет следующие шаги:

1) Вычисляет $\bar{s} = H(R \parallel P_k \parallel m)$;
2) Вычисляет $V_1 = sG$ и $V_2 = R + P_k\bar{s}$.

Подпись является действительной тогда и только тогда, когда $V_1 = V_2$.

В рамках данного исследования для анализа схемы EdDSA используется реализация на языке Python через витую кривую Эдвардса Ed25519. Благодаря ECC, рассматриваемый код работает как при подписании, так и проверке сообщений с учётом оптимизации безопасности. Реализация алгоритма EdDSA включает три ключевых этапа: генерацию ключей, процедуру подписания сообщений и механизм верификации подписей, с дополнительными функциями для оценки потребления вычислительных ресурсов – времени и памяти. Для оптимизации производительности при работе с точками ЭК используются проективные координаты, что исключает необходимость в ресурсоёмких операциях модульной инверсии, характерных для операций сложения и удвоения точек. Применение лестничного метода Монтгомери для скалярного умножения обеспечивает выполнение операций за постоянное время, что существенно повышает устойчивость к атакам по времени.

Процесс начинается с генерации 256-битного закрытого ключа с использованием криптографически стойкого генератора случайных чисел, после чего соответствующий открытый ключ вычисляется на кривой Ed25519 посредством метода Монтгомери, гарантируя как безопасность, так и вычислительную эффективность. На этапе подписания закрытый ключ хэшируется алгоритмом SHA-256 для создания детерминированного nonce, который используется при формирования публичного параметра подписи. Окончательная подпись создаётся путём комбинирования закрытого nonce, хэша сообщения (также с использованием SHA-256) и закрытого ключа.

В процессе верификации применяется бинарный метод (удвоение-и-сложение) для скалярного умножения, аналогичный используемому в ECDSA, что обеспечивает оптимальный баланс между производительностью и безопасностью при работе с открытыми параметрами. Реализация включает инструменты для точного измерения ключевых показателей производительности: времени выполнения операций генерации ключей, подписания и верификации, а также максимального потребления памяти. Данные метрики имеют принципиальное значение для оценки применимости алгоритма в устройствах с ограниченными ресурсами, таких как микроконтроллеры, где требования к эффективности использования вычислительных ресурсов и энергопотреблению являются критически важными факторами при выборе криптографических решений.

2. Сравнительный анализ RSA, ECDSA и EdDSA

Проведённое исследование представляет результаты сравнительного анализа трёх алгоритмов цифровой подписи с целью оценки их применимости для использования в средах с ограниченными ресурсами. Анализ основан на метриках времени выполнения и потребления памяти, полученных при реализации этих алгоритмов на языке Python. Тестирование каждого алгоритма включало операции подписания и верификации сообщений определённого размера.

Процесс генерации ключей, являющийся критически важным этапом для устройств с ограниченными ресурсами, показал существенные различия между алгоритмами. Наиболее ресурсоёмким оказался алгоритм RSA, требующий 2,4435 секунд и 608,654 КБ памяти для генерации ключевой пары (рисунок 1).

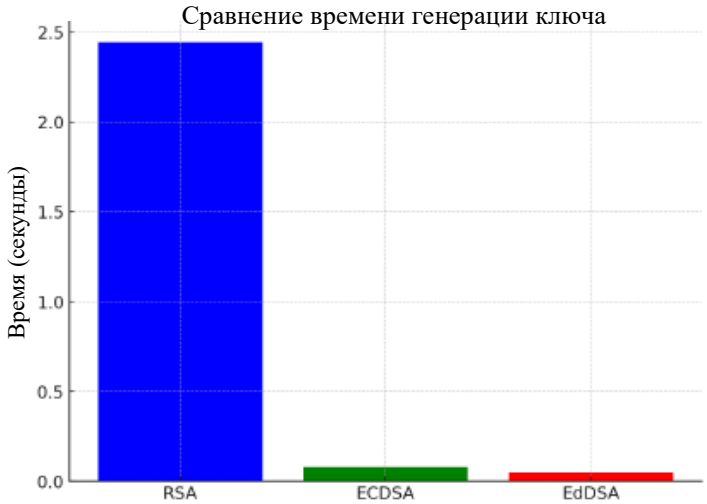

Рисунок 1 – Время, затраченное на генерацию ключа для RSA, ECDSA, EdDSA

В сравнении, ECDSA продемонстрировал значительно лучшую эффективность – всего 0,078162 секунды и 3,78 КБ памяти (рисунок 2).

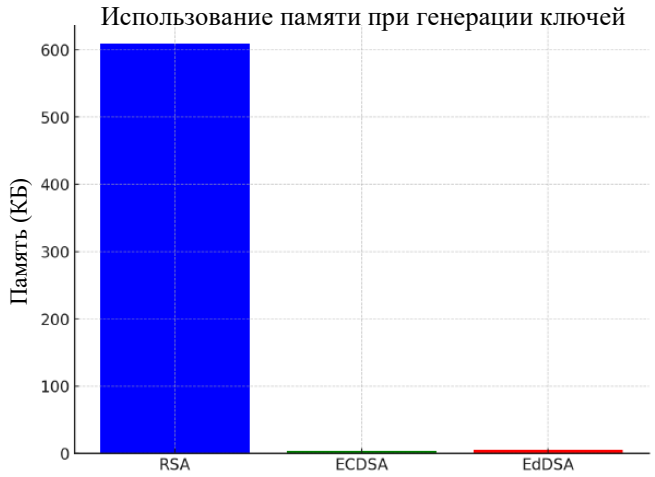

Рисунок 2 – Использование памяти при генерации ключей для RSA, ECDSA, EdDSA

Наилучшие результаты показал алгоритм EdDSA, для которого время генерации ключей составило 0,047711 секунды при потреблении 5,30 КБ памяти, что делает его оптимальным выбором для устройств с ограниченными ресурсами за счёт эффективности как с точки зрения скорости, так и потребления памяти при генерации ключей. Рассмотренные количественные показатели наглядно демонстрируют преимущества алгоритмов на эллиптических кривых (ECDSA и EdDSA) перед традиционным RSA-подходом, который является крайне требовательным к ресурсам, что делает его менее подходящим для подобных сред.

В процессе подписания время и объём памяти являются решающими факторами для определения практичности алгоритма в приложениях реального времени. В результате эксперимента алгоритм RSA подтвердил свою ресурсоёмкость, требуя 0,2306 секунды времени и 608,654 КБ памяти при подписании (рисунок 3).

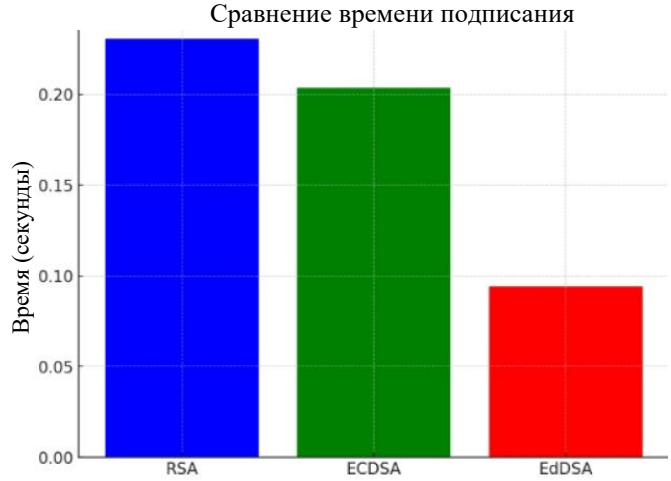

Рисунок 3 – Время, затраченное на процесс подписания для RSA, ECDSA, EdDSA

ECDSA потребовалось 0,203737 секунды и 4,48 КБ памяти. Наилучшие показатели продемонстрировал EdDSA, завершая операцию подписания всего за 0,094148 секунды и использовав 5,30 КБ памяти (рисунок 4).

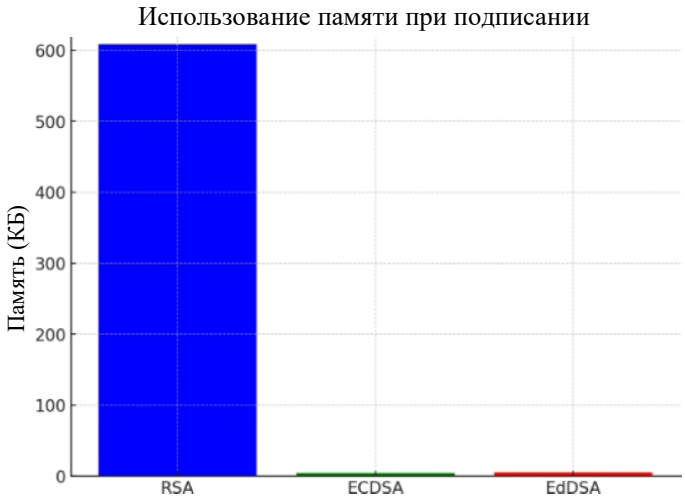

Рисунок 4 – Использование памяти при подписании для RSA, ECDSA, EdDSA

Результаты показывают, что EdDSA хорошо подходит для сред, в которых важны быстрые операции подписания и низкое потребление памяти. ECDSA также предлагает хорошую производительность, однако менее эффективную по сравнению с другими схемами из-за меньшей скорости и более высоких требований к памяти, что может оказаться не лучшим выбором для чувствительных ко времени приложений в условии ограниченных ресурсов.

Процесс верификации является заключительным этапом работы с цифровыми подписями и требует особого внимания к показателям скорости и эффективности использования памяти, что особенно критично для устройств с ограниченными ресурсами. Результаты тестирования выявили парадоксальную ситуацию: алгоритм RSA демонстрирует быстрое время проверки в 0,0043 секунды, однако достигает этого ценой чрезмерного потребления памяти в 608,654 КБ (рисунок 5).

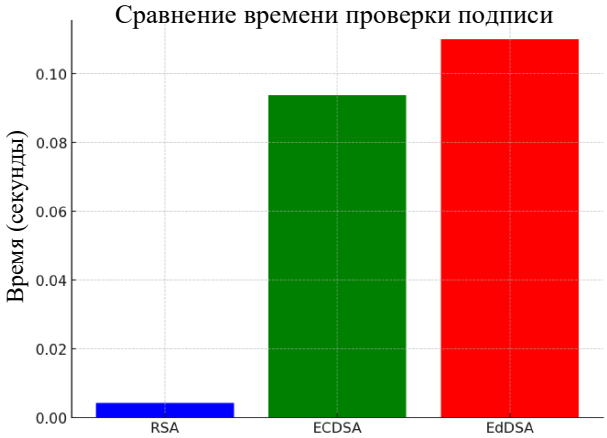

Рисунок 5 – Время, затраченное на проверку для RSA, ECDSA, EdDSA

В отличие от RSA, схемы ECDSA и EdDSA демонстрируют более сбалансированную производительность: ECDSA выполняет верификацию за 0,093747 секунды и потребляет 5,04 КБ памяти, а EdDSA – за 0,109980 секунды при 7,66 КБ памяти (рисунок 6).

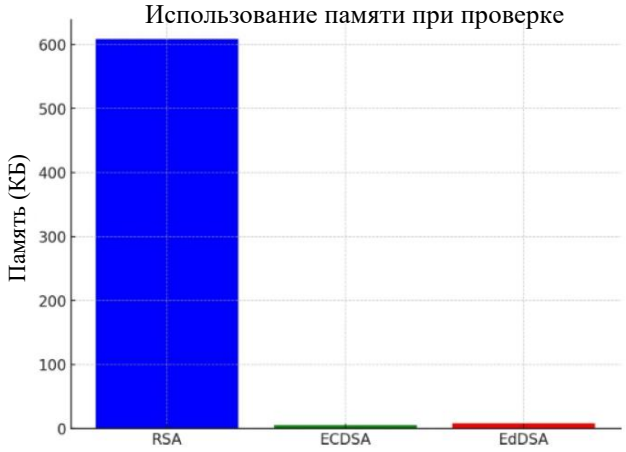

Рисунок 6 – Использование памяти при проверке для RSA, ECDSA, EdDSA

Хотя RSA лидирует по скорости, он использует значительно больше памяти, что делает его менее практичным для сред с ограниченными ресурсами, в то время как алгоритмы на эллиптических кривых обеспечивают сбалансированный компромисс между скоростью и эффективностью использования памяти, при этом EdDSA требует незначительно больше памяти, чем ECDSA.

Заключение

Проведённое исследование представляет тщательный сравнительный анализ трёх алгоритмов цифровой подписи RSA, ECDSA и EdDSA с точки зрения их применимости в ресурсоограниченных средах микроконтроллеров. Экспериментальная оценка охватила такие ключевые аспекты, как генерацию ключа, процесс подписания и верификации, с особым вниманием к временным характеристикам и потреблению памяти.

Результаты демонстрируют, что EdDSA является наиболее подходящим алгоритмом для ограниченных сред, в частности, благодаря эффективному балансу между скоростью выполнения операций и использованием ресурсов во всех протестированных процессах. Производительность EdDSA делает его оптимальным выбором для приложений, где важны как безопасность, так и эффективность. ECDSA, хотя и показывает несколько меньшую эффективность и имеет известные уязвимости, связанные с повторным использованием nonce, остаётся достойной альтернативой для определённых классов задач за счёт высокой производительности. Алгоритм RSA, несмотря на впечатляющую скорость верификации (0,0043 секунды), является наименее подходящим для сред с ограниченными ресурсами из-за высоких требований к памяти и низкой производительности при генерации ключей и подписании сообщений. Тем не менее, RSA всё ещё является жизнеспособным вариантом, особенно для сценариев, в которых микроконтроллер проверяет подписи, а не генерирует их, благодаря более быстрому процессу верификации.

Полученные результаты имеют важное практическое значение для разработчиков встраиваемых систем, предоставляя чёткие критерии выбора наиболее подходящего алгоритма цифровой подписи для безопасной и эффективной работы в средах с ограниченными ресурсами. Более того, результаты исследования также подчёркивают необходимость в дальнейшей оптимизации алгоритмов цифровой подписи для специфических требований микроконтроллерных платформ, что открывает перспективные направления для будущих исследований в области безопасности и эффективности данных алгоритмов и систем.

Благодарности

Работа выполнена в рамках государственного задания Министерства науки и высшего образования Российской Федерации № 075-00003-24-01 от 08.02.2024 (проект FSEE-2024-0003).

Список литературы

1. New directions in cryptography / Diffie, W., Hellman, M. //IEEE Transactions o Information Theory 22(6), 644–654 (1976). URL: https://doi.org/10.1109/TIT.1976.1055638;
2. A method for obtaining digital signatures and public-key cryptosystems / Rivest, R.L., Shamir, A., Adleman, L. // Communications of the ACM 21(2), 120–126 (1978). URL: https://doi.org/10.1145/359340.359342;
3. Curve25519 signature microcontrollers / Ullah, S., Zahilah, N. // Cybersecurity 4(11) (2021). URL: https://doi.org/10.1186/s42400-021-00078-6;
4. RSA Signature Algorithm for Microcontroller Implementation / Qiao, G., Lam, K.Y. // In: Quisquater, J.J., Schneier, B. (eds.) Smart Card Research and Applications. CARDIS 1998. LNCS, vol. 1820, pp. 323–334. Springer, Berlin, Heidelberg (2000). URL: https://doi.org/10.1007/10721064_32;
5. Efficient and secure ECDSA algorithm and its applications: A survey / Ibrahim, A., Dalkılıç, A. // International Journal of Communication Networks and Information Security 11(1), 1–30 (2019);
6. Efficient and secure elliptic curve cryptography for 8-bit AVR microcontrollers / Nascimento, E., López, J., Dahab, R. // Advances in Information and Communication Technology, pp. 17–30. Springer (2015). URL: https://doi.org/10.1007/978-3-319-24126-5_17;
7. An overview of cryptanalysis of RSA public key system / Kulandei, B., Dhenakaran, S. S. // International Journal of Engineering and Technology, vol. 9, no. 5, pp. 3575–3580 (2017). URL: https://doi.org/10.21817/ijet/2017/v9i5/170905312;
8. RSA digital signature scheme / Kaliski, B. // In: van Tilborg, H.C.A. (ed.) Encyclopedia of Cryptography and Security, pp. 527–531. Springer, US (2005). URL: https://doi.org/10.1007/0-387-23483-7_361;
9. The elliptic curve digital signature algorithm (ECDSA) / Johnson, D.B., Menezes, A., Vanstone, S.A. // International Journal of Information Security 1(1), 36–63 (2001);
10. Nonce generation techniques in Schnorr multi-signatures: Exploring EdDSA-inspired approaches / Sabbry, N.H., Levina, A. // AIMS Mathematics 9(8), 20304–20325 (2024). URL: https://doi.org/10.3934/math.2024988;
11. Elliptic curve cryptography on constrained devices: A comparative study of point multiplication methods / Sabbry, N.H., Levina, A. // 13th Mediterranean Conference on Embedded Computing (MECO), pp. 1–5 (2024). URL: https://doi.org/10.1109/MECO62516.2024.10577953;
12. High-speed highsecurity signatures / Bernstein, D., Duif, N., Lange, T., Schwabe, P., Yang, B.-Y. // Journal of Cryptographic Engineering, vol. 2, pp. 124–142 (2011). URL: https://doi.org/10.1007/978-3-642-23951-9_9.