turnitin unesa1

212 Production

Project 10

Document Details

Submission ID
trn:oid:::3618:142100657

Submission Date
Jun 8, 2026, 12:09 PM GMT+7

Download Date
Jun 8, 2026, 12:18 PM GMT+7

File Name
similarity erta_10+Layout+Article+176.pdf

File Size
1.8 MB

20 Pages

9,984 Words

59,034 Characters

*% detected as AI

AI detection includes the possibility of false positives. Although some text in this submission is likely AI generated, scores below the 20% threshold are not surfaced because they have a higher likelihood of false positives.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Journal of Current Studies in SDGs
Homepage : https://journal.i-ros.org/index.php/JOCSIS
Email : jocsis@i-ros.org

e-ISSN 3089-7564
Vol.2, No.1, March 2026

Towards Sustainable and Trustworthy Digital Infrastructure: Benchmarking RSA and ECDSA Digital Signature Algorithms in Support of SDGs 9 and 16

Yuliani Puji Astuti*, Ulfa Siti Nuraini

Universitas Negeri Surabaya, Surabaya, Indonesia

DOI : https://doi.org/10.63230/jocsis.2.1.212

Sections Info

Article history:
Submitted: May 21, 2026
Final Revised: June 8, 2026
Accepted: June 8, 2026
First Available Online: June 16, 2026
Publication Date: June 27, 2026

Keywords:
Cryptography;
Digital Signature;
ECDSA;
RSA;
Security.

ABSTRACT

Objective: *This study aims to evaluate and compare the performance of the Rivest–Shamir–Adleman (RSA) and Elliptic Curve Digital Signature Algorithm (ECDSA) digital signature schemes in terms of key generation, signing, verification, and storage efficiency. The research supports the advancement of secure digital communication systems aligned with Sustainable Development Goals (SDGs) 9 and 16, which emphasize innovation, resilient digital infrastructure, and trustworthy institutions.* ***Method:*** *A quantitative experimental approach was employed on a Windows AMD64 platform using Python. Five cryptographic configurations were evaluated: RSA-2048, RSA-4096, ECDSA P-256, ECDSA P-384, and ECDSA P-521. Performance tests were conducted on payload sizes of 1 KB, 10 KB, and 100 KB. Each cryptographic operation, including key generation, signing, and verification, was repeated 100 times to ensure measurement consistency and reliability.* ***Results:*** *The findings indicate that ECDSA significantly outperforms RSA in several performance aspects. ECDSA P-256 reduced signature storage requirements by 72.3%, generated keys nearly 13,000 times faster than RSA-2048, and signed 10 KB payloads approximately 48 times faster. ECDSA P-384 also demonstrated strong performance while providing a higher security level. Although RSA-2048 remains suitable for legacy systems, its efficiency is lower than ECDSA-based alternatives.* ***Novelty:*** *This study provides a comprehensive comparative evaluation of multiple RSA and ECDSA variants across different payload sizes and operational metrics, offering practical recommendations for selecting digital signature algorithms. The results highlight ECDSA P-256 as the optimal choice for 128-bit security requirements and ECDSA P-384 for applications requiring stronger 192-bit security.*

INTRODUCTION

Digital signatures are fundamental components of contemporary cryptographic systems, providing three essential security properties: authenticity, which verifies the identity of the signer; integrity, which ensures that signed data cannot be altered without detection; and non-repudiation, which prevents the signer from denying authorship of the signed document (Gjøsteen & Jager, 2018; Penubadi et al., 2023; Shukla et al., 2022). These properties form the basis of trust in modern digital ecosystems, including electronic commerce, secure communications, software distribution, financial systems, and e-government services (NIST, 2024; Mohammed, 2024).

Among public-key cryptographic schemes, RSA remains one of the most widely recognized and deployed digital signature algorithms (Shah & Gor, 2025; Tanwar & Kumar, 2019). Its security relies on the computational difficulty of factoring large semiprime integers, which continues to be computationally infeasible for sufficiently large key sizes under classical computing assumptions. However, increasing security requirements have resulted in larger modulus sizes, leading to higher computational and storage overheads compared to elliptic-curve-based approaches (Schemitt et al., 2025; Tesoro et al., 2024).

At equivalent security levels, Elliptic Curve Digital Signature Algorithm (ECDSA) schemes generally achieve smaller key sizes and faster cryptographic operations than RSA, making them more suitable for resource-constrained and large-scale digital environments. Recent studies also indicate that ECDSA continues to play a significant role in secure communication infrastructures while research communities simultaneously prepare for the transition toward post-quantum cryptography (Banerjee & Saha, 2025; Mohammed, 2024).

The Elliptic Curve Digital Signature Algorithm (ECDSA) was introduced in 1992 and standardized by NIST in successive versions of the Digital Signature Standard (Johnson et al., 2001). The security of ECDSA is based on the elliptic curve discrete logarithm problem, which is thought to be much more difficult than integer factorization, per bit of key material. That means ECDSA can reach the same 128-bit security level as RSA-3072 with a 256-bit key, therefore the keys and signatures are dramatically less (Hankerson et al., 2004). This efficiency advantage is of direct practical importance in environments where bandwidth, storage, or processing resources are constrained.

ECDSA has a theoretical efficiency advantage, but RSA is still the most common key type in many deployed systems, especially TLS certificates, document signing infrastructure, and enterprise middleware (Cooper et al., 2008). When system architects and security engineers are deciding on a deployment there are several things to consider: signing speed, verification speed, time to generate a key, output byte sizes, compatibility with existing client software, etc. All these parameters are not in the same direction, and their proportional weight depends on the unique application situation.

ECDSA has been shown to be superior to RSA in most of the operational metrics in previous comparative studies, however many of these studies were performed using older hardware and library versions, did not include standard deviations across repeated trials, or did not examine performance across various message sizes. The significant development in cryptographic library implementations in the past decade, together with the proliferation of new platforms such as IoT devices and cloud-native designs, encourages a new empirical comparison utilizing current tooling (Arif & Park, 2025).

This study aims to: Compare and measure the key generation time, signing time, verification time, size of signature, size of public key and size of private key of RSA-2048, RSA-4096, ECDSA P-256, ECDSA P-384 and ECDSA P-521 under the same experimental settings; Evaluate the effect of message size on signing and verification times using 1 KB, 10 KB and 100 KB inputs; and Reporting the best algorithm configurations for the different application scenarios. The selection of a digital signature algorithm affects not only the security posture of a system but also its performance at scale, its bandwidth footprint, and its suitability for resource-constrained deployment environments (Vidaković & Miličević, 2023). As the volume of digitally signed content grows across e-government, e-commerce, IoT, and blockchain applications, the difference between an efficient and an inefficient signature scheme compounds into measurable differences in infrastructure cost and user-facing latency (Opiłka et al., 2024). This study presents up-to-date and reproducible benchmark data based on the real-world performance of modern cryptographic libraries, which allows practitioners and researchers to make evidence-based judgments with respect to algorithm selection.

DIGITAL SIGNATURES

Contemporary asymmetric cryptography is based on one-way mathematical functions, which are computationally straightforward to evaluate in one direction, however computationally infeasible to reverse without the additional information of special rights or auxiliary parameters (García-Cid et al., 2025). In this infrastructure, digital signature schemes play a crucial role in guaranteeing the security of information exchanged over public networks by providing three assurances: integrity to detect changes to data, authentication to prove the identity of the signatory, and non-repudiation to prove responsibility (Xu, 2025).

Rivest-Shamir-Adleman (RSA) algorithm

The Rivest-Shamir-Adleman (RSA) cryptosystem represents one of the earliest standardized implementations of asymmetric infrastructure utilized for message signing (Rivest et al., 1978). The underlying security architecture of RSA rests on the classical Integer Factorization Problem (IFP), exploiting the computational asymmetry between prime multiplication and composite factorization (Xu, 2025).

The initialization of an RSA key pair requires the selection of two distinct, large secret prime numbers p and q (Imam et al., 2021; Lone & Khalique, 2016; Overmas & Venkatraman, 2021). From these primes, the cryptographic modulus n is calculated$n = pq$. The Euler totient function $\phi(n)$ is subsequently evaluated to determine the coprimality boundaries within the finite field $\phi(n) = (p-1)(q-1)$.

An encryption exponent e is selected such that $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$. Using extended Euclidean algorithms, the private decryption exponent d is derived as the multiplicative inverse of e modulo $\phi(n)$ in eq. (1).

$$ed \equiv 1 \pmod{\phi(n)} \quad (1)$$

The public verification key is comprised of the pair (e,n), while the private signing key contains the tuple (d, n, p, q).

To execute a digital signature on a message payload, a cryptographically secure hash function $H(\cdot)$ is applied to the payload to derive a fixed-length digest m (Ginting et al., 2023). The signer computes the digital signature s using their private exponent d in eq. (2).

$$s = m^d \pmod{n} \quad (2)$$

Upon receiving the payload and signature string, the verifier validates authenticity by evaluating the signature against the signatory's public exponent e in eq. (3).

$$m' = s^e \pmod{n} \quad (3)$$

The verification criteria require that $m' \equiv H(\cdot)$, confirming that the document has not been altered post-signature generation.

Elliptic Curve Digital Signature Algorithm (ECDSA)

As resource constraints and throughput demand escalated, traditional IFP systems encountered scaling bottlenecks due to the large key sizes (e.g., 2048-bit to 3072-bit keys) required to sustain contemporary security levels (Xu, 2025). ECDSA derives its cryptographic resilience from the Elliptic Curve Discrete Logarithm Problem (ECDLP), which lacks known sub-exponential time solution algorithms on classical systems (Lyu, 2025).

ECDSA operates over a finite field, where curves are commonly specified using the short Weierstrass equation:

$$y^2 = x^3 + ax + b \pmod{p} \quad (4)$$

While guaranteeing algebraic structural security, the curve constants must fulfill the non-singularity condition where $4a^3 + 27b^2 \neq 0 \pmod{p}$. The operational domain is bounded by a base point *G* situated on the curve, possessing a large prime order *n*.

The private key *d* is defined as a randomly chosen scalar integer within the group range $d \in [1, n-1]$. The public verification key *Q* represents a computed coordinate point on the elliptic curve generated via group scalar multiplication with $Q = d \cdot G$. To sign a message digest $e = H(M)$, the signer isolates a cryptographically secure, unpredictable random nonce k within the field range $[1, n-1]$. The signer computes the ephemeral curve point as $k \cdot G = (x_1, y_1)$. The initial signature component *r* is derived from the *x*-coordinate projection, as $r = x_1 \pmod{n}$ (where $r \neq 0$). The secondary signature component *s* acts as an algebraic link uniting the message digest, the private key, and the ephemeral nonce inverse:$s = k^{-1}(e + r \cdot d) \pmod{n}$ (where $s \neq 0$). The generated ECDSA digital signature is packaged as the numerical coordinate pair. During the verification phase, the verifier computes the inverse parameter $w = s^{-1} \pmod{n}$ to isolate the scalar components like in eq. (5).

$$u_1 = e \cdot w \pmod{n} \text{ and } u_2 = r \cdot w \pmod{n} \quad (5)$$

The verifier reconstructs the target validation point *X* on the elliptic curve using the public key *Q*, as in eq. (6).

$$X = u_1 \cdot G + u_2 \cdot Q \quad (6)$$

The signature is mathematically accepted if and only if the *x*-coordinate projection of point *X* satisfies $x_X \pmod{n} = r$.

EXPERIMENTAL METHODOLOGY

The present study employs a quantitative experimental methodology to assess and compare the performance of RSA and ECDSA digital signature algorithms. The comparison is performed over five cryptographic dimensions: time to generate the key, time to sign, time to verify, output byte sizes, and key byte sizes. All experiments are conducted in the same hardware and software environment to verify the validity and repeatability of the results.

Selecting and configuring algorithms

Five algorithm configurations are selected to reflect a wide security spectrum from the 112-bit security level suggested for near-term use to the 256-bit level recommended for long-term protection. We evaluate RSA at two key sizes: 2048 bits and 4096 bits, which are the most typically deployed sizes in contemporary systems like TLS certificates and document signing infrastructures. For ECDSA, we consider three NIST-standardized curves; such as P-256, P-384, and P-521, corresponding to 128-bit, 192-bit, and 256-bit security levels respectively. All settings use SHA-256 as hash function to ensure that the performance variations are caused by the signature algorithm itself and not the hash calculation.

The choice of these configurations is based on the current guidelines from the NIST as defined in FIPS 186-5 and Special Publication 800-57. RSA-2048 and ECDSA P-256 are the minimum acceptable configurations for modern secure applications, with RSA-4096 and ECDSA P-521 being used for high-security installations. A detailed overview of the tested combinations is shown in Table 1.

Table 1. Algorithm used in the benchmark experiment.

Algorithm	Type	Key Size (bits)	Security Level (bits)
RSA-2048	RSA	2048	112
RSA-4096	RSA	4096	140
ECDSA P-256	ECDSA	256	128
ECDSA P-384	ECDSA	384	192
ECDSA P-521	ECDSA	521	256

Performance metrics

Six performance metrics are measured for each algorithm configuration. These metrics are chosen to reflect the practical requirements of systems that depend on digital signatures, including server-side authentication, mobile applications, and resource-constrained IoT devices. The metrics are defined as follows.

Key generation time measures the total duration required to produce a mathematically valid public and private key pair. This statistic is important for applications that create ephemeral keys every session, e.g. for forward-secrecy protocols. *Signing time* is the time needed to create a digital signature for a given message, while *verification time* is the time needed to verify the authenticity and integrity of that signature. Both metrics are tested on three message sizes to test the sensitivity to input size. *Signature size*, the *public key size* and *the private key size* in bytes, by exporting the respective objects in PEM format and measuring their byte length. These size measures have a direct impact on bandwidth usage, storage needs, and processing overhead in network protocols and file systems. A complete definition of each metric is provided in Table 2.

Table 2. Performance metrics measured in this study

Metric	Description	Measurement	Unit
Key generation time	Time to generate a public/private key pair	1000 iterations, compute mean	Ms
Signing time	Time to produce a digital signature	100 iterations per message size	Ms
Verification time	Time to verify an existing signature	100 iterations per message size	Ms
Signature size	Byte length of the generated signature	Measure output of sign()	Bytes
Public key size	Byte length of the exported public key	Export PEM, measure length	Bytes
Private key size	Byte length of the exported private key	Export PEM, measure length	Bytes

Experimental design

Each signing and verification operation is repeated 100 times per message size, and each key generation operation is repeated 10 times due to its substantially longer execution duration. Running many iterations decreases the influence of the operating system scheduling noise, CPU cache state and transitory background operations on the recorded observations. Within each set of iterations, the arithmetic means are computed and saved.

The benchmark input contains three message sizes: 1 kilobyte for a typical short document or authentication token, 10 kilobytes for a regular contract or form, and 100 kilobytes for a bigger document such as a PDF report. A cryptographically secure random byte generator is used to produce messages before the experiment, so that there is no

chance of caching effects affecting the timing results. The messages are hashed using SHA-256 before being given to the signing function.

The timing mechanism used is Python `time.perf_counter()`, which provides sub-microsecond resolution on modern operating systems and is not subject to adjustments from the system clock. Timing measurements are taken immediately before and after the cryptographic call to minimize overhead from surrounding code.

The RSA implementation uses the PyCryptodome library with the PKCS 1 v1.5 signature scheme. Private key objects are created using `RSA.generate()`, and signatures are produced using `pkcs1_15.new().sign()` with a SHA-256 hash object. Verification is performed using `pkcs1_15.new().verify()`, which raises a ValueError if the signature is invalid.

The ECDSA implementation uses the cryptography library, specifically the hazardous materials (hazmat) interface. Private keys are generated using `ec.generate_private_key()` with the selected NIST curve and the default backend. Signatures are produced using `private_key.sign()` with the `ec.ECDSA(hashes.SHA256())` algorithm. Verification is performed using `public_key.verify()`, and a failed verification raises an InvalidSignature exception.

Key sizes are measured by exporting private keys in PEM format using TraditionalOpenSSL encoding without encryption, and public keys in SubjectPublicKeyInfo PEM format. These export formats are standard and interoperable across programming languages and operating environments.

To remove confounding variables caused by hardware heterogeneity, all studies are carried out on a single system with a uniform software environment. During the benchmarking period, no more computationally demanding operations are executed concurrently.

RESULTS AND DISCUSSION

In this section, we show the numerical results of the benchmark experiment on a Windows AMD64 platform with Python. We show all timing metrics as arithmetic which means over 100 repetitions of each operation. Results are presented in four metrics: key generation, signing and verification, output byte sizes, and practical deployment advice. All figures and tables cited in this section are directly taken from the benchmark data acquired throughout the experiment.

Results

Key generation performance

Key generation time was the dimension that produced the largest absolute difference between the two algorithm families (Ferretti et al., 2011; Tsai & Cho, 2021; Xu et al., 2016). RSA key generation relies on finding two large random prime numbers through repeated probabilistic primality testing, a process whose computational cost scales super-linearly with key length (Ozpinar & Serengil, 2026). RSA-2048 required a mean of 943.231 ms to produce a key pair, while RSA-4096 required 19,012.405 ms, which is approximately 20.1 times longer. The standard deviation for RSA-4096 was 14,113.74 ms, which is 74.2 percent of the mean, indicating that the duration of a single RSA-4096 key generation can vary by an order of magnitude from one invocation to the next.

Elliptic curve key generation does not require prime search (Kumar & Sharma, 2023). A private key is simply a uniformly random integer in the curve's scalar field, and the corresponding public key is a scalar multiplication of that integer with the curve's base

point. ECDSA P-256 averaged 0.073 ms, ECDSA P-384 averaged 2.074 ms, and ECDSA P-521 averaged 4.797 ms. The ratio between RSA-2048 and ECDSA P-256 exceeds 12,900 to 1. Even ECDSA P-521 completed key generation nearly 200 times faster than RSA-2048. The standard deviations for ECDSA key generation remained below 2 ms, confirming that elliptic curve key generation is both faster and considerably more predictable than RSA. Table 3 presents the complete key generation results.

Table 3. Key generation mean and standard deviation each algorithm

Algorithm	Security Level (bits)	Key Gen Mean (ms)	Key Gen Std Dev (ms)
RSA-2048	112	943.231	501.46
RSA-4096	140	19,012.405	14,113.74
ECDSA P-256	128	0.073	0.06
ECDSA P-384	192	2.074	0.61
ECDSA P-521	256	4.797	1.61

The high variability of RSA-4096 key generation makes it unsuitable for latency-sensitive contexts. Protocols that implement perfect forward secrecy by generating fresh key material per session would experience unpredictable delays if RSA-4096 were used. ECDSA P-256 and P-384 exhibit sub-millisecond and low-millisecond durations with low variance, making them far more appropriate for real-time or high-throughput scenarios such as mutual TLS authentication, ephemeral key exchange in session protocols, and certificate issuance at scale.

Signing and verification time

Signing time was measured at three message sizes. At the 10 KB reference size, RSA-2048 signing averaged 5.0767 ms with a standard deviation of 1.8106 ms. RSA-4096 signing averaged 26.3675 ms with a standard deviation of 31.5704 ms. The high standard deviation for RSA-4096 signing reflects the behavior of modular exponentiation on the Windows platform, where thread scheduling interruptions are more frequent during long CPU-bound operations than on Linux server environments. ECDSA P-256 signing averaged 0.1057 ms at 10 KB, making it approximately 48 times faster than RSA-2048 and approximately 249 times faster than RSA-4096. Figure 1 illustrates how signing and verification times scale with message size.

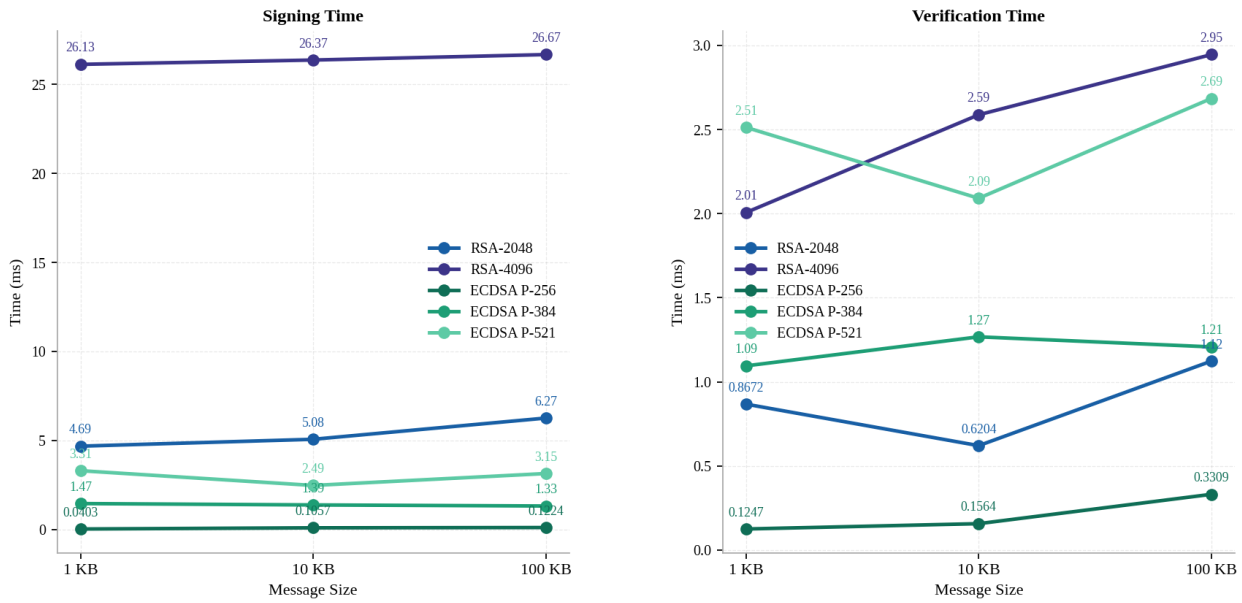

Figure 1. Signing time (left) and verification time (right) as a function of message size for all five algorithm configurations.

Verification results reveal a structural asymmetry within each algorithm. RSA verification uses the public exponent, which in standard PKCS#1 deployments is the smallest fixed value 65537. This makes RSA verification far cheaper than RSA signing: RSA-2048 verification averaged 0.6204 ms compared to 5.0767 ms for signing at 10 KB, a ratio of approximately 8 to 1. ECDSA verification requires two scalar point multiplications on the curve rather than one, making it inherently slower than ECDSA signing, which requires only one. ECDSA P-256 verification averaged 0.1564 ms at 10 KB, compared to 0.1057 ms for signing. Table 4 presents the timing of complete signing and verification data.

Table 4. Signing and verification time for all algorithms across three message sizes

Algorithm	Sign 1KB (ms)	Sign 10KB (ms)	Sign 100KB (ms)	Verify 1KB (ms)	Verify 10KB (ms)	Verify 100KB (ms)
RSA-2048	4.6900	5.0767	6.2700	0.8672	0.6204	1.1200
RSA-4096	26.1300	26.3675	26.6700	2.0100	2.5880	2.9500
ECDSA P-256	0.0403	0.1057	0.1124	0.1100	0.1564	0.1700
ECDSA P-384	1.4700	1.3912	1.3300	1.0900	1.2675	1.2100
ECDSA P-521	3.5100	2.4852	3.1500	2.5100	2.0914	2.6900

The sensitivity of ECDSA to message size at large inputs is attributable to the SHA-256 hashing step applied to the full message before the elliptic curve operation is invoked. The curve arithmetic itself operates on a fixed-size hash digest regardless of the original message length. ECDSA P-256 signing rose from 0.034 ms at 1 KB to 0.111 ms at 100 KB, a relative increase of 226 percent, though the absolute values remained well below all RSA configurations. RSA-2048, in the same range, showed a relatively moderate 35 percent relative gain.

Among the ECDSA configurations, the performance difference between curves reflects the increase in field arithmetic operations required as the curve order grows. P-384 and P-521 use larger prime fields, which increases the cost of each field multiplication in the

point addition steps. ECDSA P-384 signing at 10 KB averaged 1.3912 ms, which is 13.2 times slower than P-256 at the same size. ECDSA P-521 averaged 2.4852 ms, which is 23.5 times slower than P-256. Despite these intra-ECDSA differences, all three ECDSA curves signed faster than RSA-2048 at 10 KB, and all three verified faster than RSA-4096.

Signature and key sizes

Signature length in RSA is determined by the modulus size: a 2048-bit modulus produces a 256-byte signature, and a 4096-bit modulus produces a 512-byte signature. ECDSA signatures are encoded as a pair of integers (r, s) in ASN.1 DER format, where each integer is approximately half the curve order in bytes. ECDSA P-256 produced 71-byte signatures, which is 27.7 percent of the RSA-2048 output. ECDSA P-384 produced 103-byte signatures and ECDSA P-521 produced 139-byte signatures. Even the largest ECDSA configuration generated signatures smaller than the smallest RSA configuration tested.

Public key size follows a similar pattern. RSA public keys must encode the full modulus, whereas ECDSA public keys encode a single point on the curve, which can be stored in compressed form. RSA-2048 public keys occupied 450 bytes in PEM format, while ECDSA P-256 public keys occupied 178 bytes, a reduction of 60.4 percent. Private key sizes ranged from 227 bytes for ECDSA P-256 to 3,242 bytes for RSA-4096. Table 5 presents the complete size comparison with ratios relative to RSA-2048.

Table 5. Signature and key sizes in bytes with relative comparison to RSA-2048

Algorithm	Signature (bytes)	Public Key (bytes)	Private Key (bytes)
RSA-2048	256	450	1674
RSA-4096	512	799	3242
ECDSA P-256	71	178	227
ECDSA P-384	103	215	288
ECDSA P-521	139	268	365

In TLS 1.3, the server transmits its certificate chain during the handshake, and the public key size directly affects the volume of data exchanged before a secure session is established. Smaller keys reduce the number of TCP segments required, which decreases sensitivity to round-trip latency, particularly on high-latency links. Replacing an RSA-2048 public key with an ECDSA P-256 key reduces the public key contribution to each certificate in the chain by approximately 272 bytes. In blockchain systems, each transaction permanently stores a signature on the ledger. A network processing one million transactions per day using ECDSA P-256 instead of RSA-2048 would store approximately 185 gigabytes less signature data per year.

Platform observations

Figure 2 provides a visual summary of all six benchmark dimensions. It shows that the best performance from six benchmark metrics is ECDSA P-256. The key generation, signing, and verification time give the lowest time. The signature, public key, and private key also give the lowest value. All measurements were taken on a single Windows AMD64 machine. Performance characteristics can differ substantially from hardware with native ECC acceleration instructions, on ARM processors commonly used in mobile and embedded devices, or on microcontrollers without a floating-point unit.

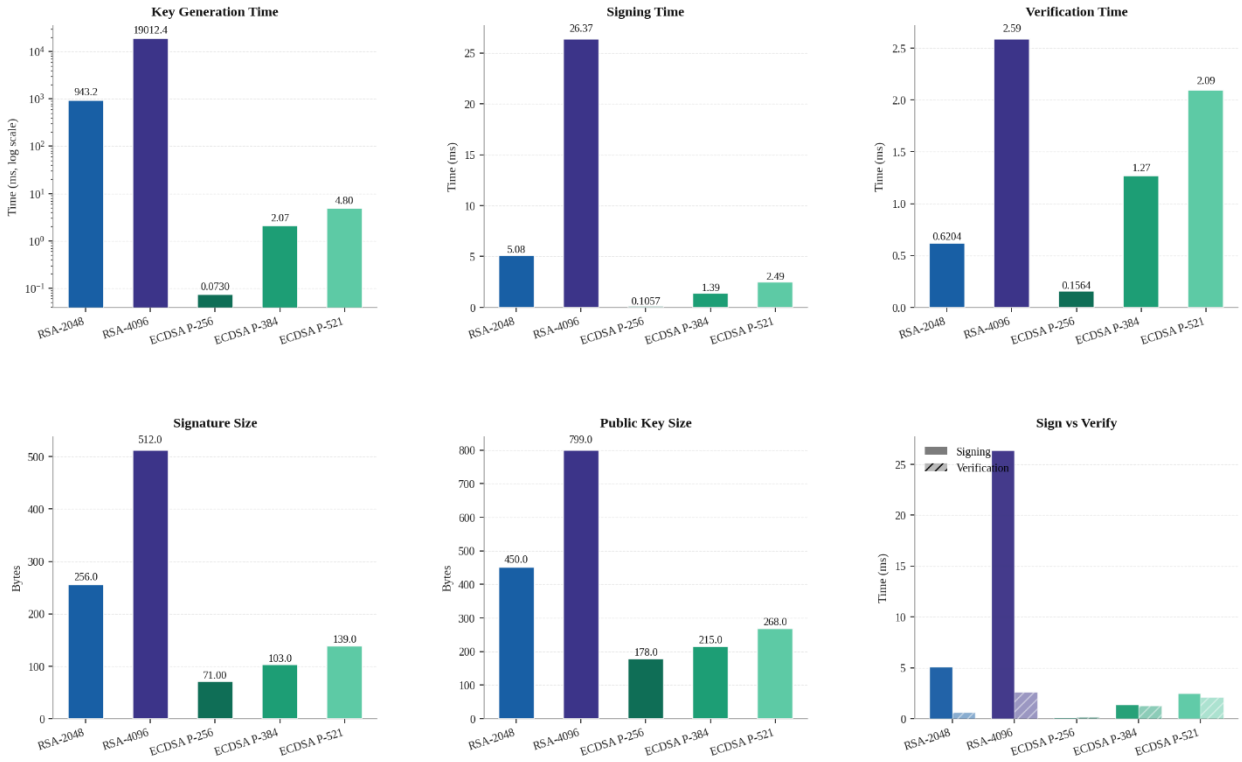

Figure 2. Six-panel benchmark dashboard: key generation time (log scale), signing time, verification time, signature size, public key size, and sign-versus-verify grouped comparison at 10 KB

Discussion

The results obtained from the experiment were compatible with the theoretical basis of both cryptographic families. RSA security depends on integer factorization. To be secure enough, RSA keys must be large, but the cost of an RSA operation grows badly with key size. The security of ECDSA is based on the difficulty of the elliptic curve discrete logarithm problem. The advantage of this problem is that it offers the same level of security as much smaller keys. This reduces all the expenses related to computation and storage. ECDSA P-256 was almost 12,900 times faster than RSA-2048 in key generation, 48 times faster in signing, and generated signatures 72.3 percent smaller at an equal or greater security level as defined in NIST SP 800-57.

The functional correctness test demonstrated that both families of algorithms correctly implement signature integrity, i.e., a valid signature verifies successfully. This experiment defines quality of a secure digital signature technique which gives both authenticity and non-repudiation to the signed content. The findings indicate a migration from RSA to ECDSA for new deployments, and ECDSA P-256 as the default configuration where 128-bit security level is acceptable. ECDSA P-384 should be reserved for use cases that demand a higher security margin.

Beyond the technical implications, these findings also contribute to the achievement of Sustainable Development Goals (SDGs), particularly SDG 9 (Industry, Innovation and Infrastructure) and SDG 16 (Peace, Justice and Strong Institutions). SDG 9 emphasizes the development of resilient infrastructure and the promotion of innovation. The superior efficiency demonstrated by ECDSA, especially ECDSA P-256, enables the deployment of secure digital services with lower computational and storage requirements, making robust cryptographic protection more accessible for cloud platforms, mobile applications,

Internet of Things (IoT) devices, and large-scale digital infrastructures. Such efficiency supports the development of sustainable and scalable digital ecosystems while reducing operational overhead.

Furthermore, SDG 16 highlights the importance of building effective, accountable, and trustworthy institutions. Digital signatures serve as a fundamental mechanism for ensuring authenticity, integrity, and non-repudiation in electronic transactions and public digital services. By identifying efficient digital signature schemes capable of maintaining strong security guarantees, this study provides practical evidence that may assist governments, organizations, and system developers in strengthening trust in e-government systems, digital identity services, electronic document management, and other institutional processes that rely on secure digital interactions. Therefore, the selection of efficient cryptographic mechanisms should be viewed not only as a technical decision but also as a strategic contribution toward sustainable digital transformation and the realization of SDGs 9 and 16.

CONCLUSION

Fundamental Finding: This study presents a controlled quantitative benchmark of RSA-2048, RSA-4096, ECDSA P-256, ECDSA P-384, and ECDSA P-521 across six performance dimensions on a Windows AMD64 platform. The findings demonstrate that ECDSA consistently outperforms RSA in key generation, signing efficiency, storage requirements, and overall computational performance while maintaining equivalent or higher security levels. Among the evaluated schemes, ECDSA P-256 provides the most efficient balance between security and performance for modern digital signature applications, while ECDSA P-384 offers stronger security with competitive efficiency. **Implication:** The results suggest that ECDSA-based digital signature schemes are better suited for contemporary secure communication systems, particularly in environments that require high performance, reduced storage overhead, and scalable security. Organizations deploying new digital infrastructures can benefit from adopting ECDSA P-256 for 128-bit security requirements and ECDSA P-384 when a 192-bit security margin is required, while RSA-2048 remains relevant primarily for legacy interoperability and regulatory environments that mandate RSA usage. From an SDGs perspective, these findings support SDG 9 through the promotion of efficient and resilient digital infrastructure, and SDG 16 by enhancing trust, integrity, and accountability in digital services and institutional processes that depend on secure communication technologies. **Limitation:** This study is limited to benchmarking on a Windows AMD64 platform and focuses solely on classical public-key signature algorithms. The evaluation does not consider performance variations across different hardware architectures, embedded systems, energy consumption metrics, or post-quantum cryptographic schemes. **Future Research:** Future studies should extend the benchmark to post-quantum digital signature algorithms and evaluate performance on ARM-based devices, microcontrollers, and resource-constrained environments. Additional investigations may also include energy efficiency, scalability, and real-world deployment scenarios to provide a more comprehensive assessment of digital signature technologies.

ACKNOWLEDGEMENTS

This research was funded by the Directorate of Research and Community Service (LPPM) of Universitas Negeri Surabaya under the research contract number 343/UN38/HK/PP/2024. The authors also express their gratitude to the Faculty of Mathematics and Natural Science and the Laboratory of Big Data Analysis for providing the facilities to conduct this study.

AUTHOR CONTRIBUTIONS

Yuliani Puji Astuti contributed to the conceptualization of the study, research design, methodology development, validation, supervision, and review and editing of the manuscript. **Ulfa Siti Nuraini** contributed to data collection, software implementation, experimental testing, formal analysis, visualization, literature review, and preparation of the original manuscript draft. Both authors participated in the interpretation of results, reviewed the final manuscript, and approved the submitted version.

CONFLICT OF INTEREST STATEMENT

The authors confirm that there are no conflicts of interest, either financial or personal, that may have influenced the content or outcome of this study.

ETHICAL COMPLIANCE STATEMENT

This manuscript complies with research and publication ethics. The authors affirm that the work is original, conducted with academic integrity, and free from any unethical practices, including plagiarism.

STATEMENT ON THE USE OF AI OR DIGITAL TOOLS IN WRITING

The authors acknowledge the use of ChatGPT (OpenAI) as an AI-assisted writing tool during the preparation of this manuscript. ChatGPT was utilized to support language refinement, grammar checking, text restructuring, and improvement of academic writing clarity. The tool was also used to assist in drafting and revising selected sections of the manuscript. All AI-generated outputs were carefully reviewed, verified, and edited by the authors to ensure accuracy, originality, academic integrity, and compliance with ethical research standards. The authors take full responsibility for the content of this article and the conclusions presented herein.

REFERENCES

Arif, T., Jo, B., & Park, J. H. (2025). A comprehensive survey of privacy-enhancing and trust-centric cloud-native security techniques against cyber threats. *Sensors*, *25*(8), 2350. https://doi.org/10.3390/s25082350

Banerjee, K., & Saha, S. (2025). *Blockchain signatures to ensure information integrity and non-repudiation in the digital era*. arXiv. https://doi.org/10.48550/arXiv.2510.22561

Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., & Polk, W. (2008). *Internet X.509 public key infrastructure certificate and Certificate Revocation List (CRL) profile*. RFC Editor.

Ferretti, A., Fumagalli, A., Novali, F., Prati, C., Rocca, F., & Rucci, A. (2011). A new algorithm for processing interferometric data-stacks: SqueeSAR. *IEEE Transactions on Geoscience and Remote Sensing*, *49*(9), 3460–3470. https://doi.org/10.1109/TGRS.2011.2124465

García-Cid, M. I., Martín, R., Domingo, D., Martín, V., & Ortiz, L. (2025). Design and implementation of a quantum-assisted digital signature. *Cryptography*, *9*(1), 11. https://doi.org/10.3390/cryptography9010011

Gjøsteen, K., & Jager, T. (2018). Practical and tightly-secure digital signatures and authenticated key exchange. *Annual International Cryptology Conference*, *3*(319), 95–125. https://doi.org/10.1007/978-3-319-96881-0_4

Hankerson, Menezes, A. J., & Vanstone, S. A. (2004). *Guide to elliptic curve cryptography* (2004th ed.). Springer.

Kumar, S., & Sharma, D. (2023). Key generation in cryptography using elliptic-curve cryptography and genetic algorithm. *Engineering Proceedings*, *59*(1), 59. https://www.mdpi.com/2673-4591/59/1/59#

Imam, R., Areeb, Q. M., Alturki, A., & Anwer, F. (2021). Systematic and critical review of rsa based public key cryptographic schemes: Past and present status. *IEEE Access*, *9*, 155949–155976. https://doi.org/10.1109/ACCESS.2021.3129224

Johnson, D., Menezes, A., & Vanstone, S. (2001). The elliptic curve digital signature algorithm (ECDSA). *Int. J. Inf. Secur.*, *1*(1), 36–63. https://doi.org/10.1007/s102070100002

Lenstra, A. K., & Verheul, E. R. (2001). Selecting cryptographic key sizes. *J. Cryptol.*, *14*(4), 255–293. https://doi.org/10.1007/s00145-001-0009-4

Lone, A. H., & Khalique, A. (2016). Generalized RSA using 2k prime numbers with secure key generation. *Security and Communication Networks*, *9*(17), 4443–4450. https://doi.org/10.1002/sec.1619

Lyu, S. (2025). Advances in Digital Signature Algorithms: Performance, Security and Future Prospects. *ITM Web of Conferences*, *73*, 03010. https://doi.org/10.1051/itmconf/20257303010

Mohammed, Q. A. A. S., Joudah, M., & Mohammed, H. (2024). A survey on digital signature schemes. In *AIP Conference Proceedings* (Vol. 3232, No. 1). AIP Publishing. https://doi.org/10.1063/5.0236576

National Institute of Standards and Technology. (2023). *Digital Signature Standard (DSS) (FIPS PUB 186-5)*. U.S. Department of Commerce. https://doi.org/10.6028/NIST.FIPS.186-5

Octora Ginting, F. S., Veithzal Rivai Zainal, & Aziz Hakim. (2023). Digital signature standard implementation strategy by optimizing hash functions through performance optimization. *Journal of Accounting and Finance Management*, *3*(6), 362–371. https://doi.org/10.38035/jafm.v3i6.175

Overmars, A., & Venkatraman, S. (2021). New semi-prime factorization and application in large RSA key attacks. *Journal of Cybersecurity and Privacy*, *1*(4), 660-674. https://doi.org/10.3390/jcp1040033

Opiłka, F., Niemiec, M., Gagliardi, M., & Kourtis, M. A. (2024). Performance analysis of post-quantum cryptography algorithms for digital signature. *Applied Sciences*, *14*(12), 4994. https://doi.org/10.3390/app14124994

Ozpinar, A., & Serengil, S. I. (2026). Sustainable cryptography: Carbon asymmetry in partially homomorphic encryption in the cloud. *Symmetry*, *18*(5), 832. https://doi.org/10.3390/sym18050832

Paar, C., & Pelzl, J. (2010). *Understanding cryptography*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-04101-3

Penubadi, H. R., Shah, P., Sekhar, R., Alrasheedy, M. N., Niu, Y., Radhi, A. D., Tharwat, M., Tawfeq, J. F., Gheni, H. M., & Abdulbaqi, A. S. (2023). Sustainable electronic

document security: A comprehensive framework integrating encryption, digital signature and watermarking algorithms. *Heritage and Sustainable Development*, *5*(2), 391–404. https://doi.org/10.37868/hsd.v4i1.359

Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, *21*(2), 120–126. https://doi.org/10.1145/359340.359342

Schemitt, A. G., Silva, H. F. da, Lunardi, R. C., Kreutz, D., Mansilha, R. B., & Zorzo, A. F. (2025). Assessing the impact of post-quantum digital signature algorithms on blockchains. In *2025 IEEE 24th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)* (pp. 2373–2380). IEEE. https://doi.org/10.1109/TrustCom66490.2025.00276

Shah, A. M., & Gor, A. (2025). Comprehensive survey of symmetric and public-key cryptographic algorithms: Foundations, attacks, and applications. *International Journal Of Informative and Futuristic Research*, *12*(10), 20–38. Retrieved from: https://ijifr.org/pdfsave/18-06-2025917IJIFR-V12-E10-005.pdf

Shukla, P. K., Aljaedi, A., Pareek, P. K., Alharbi, A. R., & Jamal, S. S. (2022). AES-based white-box cryptography in digital signature verification. *Sensors, 22*(23), 9444. https://doi.org/10.3390/s22239444

Tanwar, S., & Kumar, A. (2019). An efficient and secure identity based multiple signatures scheme based on RSA. *Journal of Discrete Mathematical Sciences and Cryptography*, *22*(6), 953–971. https://doi.org/10.1080/09720529.2019.1632024

Tesoro, M., Siloi, I., Jaschke, D., Magnifico, G., & Montangero, S. (2024). *Quantum inspired factorization up to 100-bit RSA number in polynomial time*. arXiv. https://doi.org/10.48550/arXiv.2410.16355

Tsai, M.-Y., & Cho, H.-H. (2021). A high security symmetric key generation by using genetic algorithm based on a novel similarity model. *Mobile Networks and Applications*, *26*(3), 1386–1396. https://doi.org/10.1007/s11036-021-01753-1

Vidaković, M., & Miličević, K. (2023). Performance and Applicability of Post-Quantum Digital Signature Algorithms in Resource-Constrained Environments. *Algorithms*, *16*(11), 518. https://doi.org/10.3390/a16110518

Xu, J. (2025). A Comprehensive study of digital signatures: Algorithms, challenges and future prospects. *ITM Web of Conferences*, *73*, 03009. https://doi.org/10.1051/itmconf/20257303009

Xu, P., Cumanan, K., Ding, Z., Dai, X., & Leung, K. K. (2016). Group secret key generation in wireless networks: Algorithms and rate optimization. *IEEE Transactions on Information Forensics and Security*, *11*(8), 1831–1846. https://doi.org/10.1109/TIFS.2016.2553643

*Yuliani Puji Astuti (Corresponding Author)
Department of Data Science, Faculty of Mathematics and Natural Sciences,
Universitas Negeri Surabaya, Surabaya, Indonesia.
Ketintang Campus, Jl. Ketintang, Surabaya, East Java, Indonesia, 60231
Email: yulianipuji@unesa.ac.id

Ulfa Siti Nuraini
Department of Data Science, Faculty of Mathematics and Natural Sciences,
Universitas Negeri Surabaya, Surabaya, Indonesia.
Ketintang Campus, Jl. Ketintang, Surabaya, East Java, Indonesia, 60231
Email: ulfanuraini@unesa.ac.id