Real-world performance analysis of signcryption and sign-then-encrypt schemes for resource-constrained IoT devices

KOEN MOLENAAR, University of Twente, The Netherlands

For resource-constrained devices, the traditional cryptographic schemes are often too heavy. Therefore, signcryption was invented. Signcryption performs signing and encryption at the same time and is theoretically more efficient. However, no research has been done comparing the performance of signcryption schemes with traditional cryptographic schemes in a practical manner. This research will implement multiple signcryption and traditional cryptographic schemes on a resource-constrained device and measure computational costs, power draw & communication overhead. These measurements will be analysed and used to create a comprehensive comparison of real-world performance. The research will add to the scientific body of knowledge of comparisons between signcryption and traditional cryptographic schemes.

Additional Key Words and Phrases: Signcryption, Performance, Comparison, Real-world, Resource-constrained

1 INTRODUCTION

Over the past decades, billions of devices have been connected with each other via the Internet [15]. While a portion of these devices are still traditional computers, servers, phones, etc, there has been a rise in the number of everyday objects connected to the Internet via embedded sensors [15]. These types of devices connected via the Internet are collectively known as the Internet of Things (IoT). IoT devices are most commonly used to gather large amounts of data. This data is generally transmitted over a network to an application server or cloud, which handles the processing and makes the data available for the end-user [16]. Since these IoT devices use low-power embedded sensors and processors, they are resource-constrained.

The ability to provide security for data during transit has been said to be a key factor for increased adoption of IoT devices [10]. To transmit data securely, cryptographic schemes are used for data confidentiality, while digital signature algorithms are used for data integrity [14]. However, since IoT devices are resource-constrained regarding their storage, memory usage, and processing capabilities, they cannot use the traditional heavy-weighted cryptographic schemes to securely transmit data [6]. Because of this, there is a need for a lightweight cryptographic scheme designed for resource-constrained devices [4].

A proposed solution is signcryption. The traditional workflow to transmit data is to first sign the data, then encrypt it and finally transmit it. This is known as sign-then-encrypt. With signcryption, data is signed and encrypted at the same time. As proposed by Zheng in [19], this could lead to 58% less computational costs and 85% less overhead with communication.

Although research has been done into the theoretical performance gains of signcryption schemes, the studying of the real-world impact on performance has not been done. Next to this, no research has been performed looking into how these computational performance gains transform into power-draw gains.

This problem statement leads to this research question:

How do signcryption and sign-then-encrypt schemes differ in performance when compared in a practical scenario?

This research question can be answered with these sub-questions:

(1) **RQ1: How do signcryption and sign-then-encrypt schemes compare when computational cost is measured practically?**

(2) **RQ2: How do signcryption and sign-then-encrypt schemes compare when power draw is measured practically?**

(3) **RQ3: How do signcryption and sign-then-encrypt schemes compare when communication overhead is measured practically?**

The approach to this research paper involves several steps. First, related work will be researched. Then, the background behind signcryption is explained. After this, we will implement signcryption and sign-then-encrypt schemes and practically compare their performance. This paper will measure the computational performance, power draw and communication overhead of signcryption and sign-then-encrypt schemes on a resource-constrained device. These measurements will be analysed and compared with a comparative analysis based on literature. Finally, a conclusion will be made together with a recommendation for choosing between signcryption and sign-then-encrypt schemes.

The paper is organised as follows: Section 2 discusses work related to this paper. Section 3 explains signcryption in detail and selects several schemes to which signcryption will be compared. Section 4 explains the research methodology, measurement environment and measurement tools. Section 5 shows the results of the research. Section 6 contains an analysis of the results. Section 7 concludes this paper and recommends a choice of cryptographic scheme.

2 RELATED WORK

Signcryption as a concept was first introduced in 1997 by Zheng in [19]. The motivation for his research was the fact that while secure message delivery is a significant aim of communication security research, no alternatives to sign-then-encrypt had been created since the start of public-key cryptography. Zheng's signcryption

TScIT 37, July 8, 2022, Enschede, The Netherlands

scheme promised a 50% reduction in computational costs and an 85% reduction in communication overhead.

The work by Elshobaky et al. [5] from 2014 implements the Schnorr Signcryption and the RSA Encrypt-Sign-Encrypt schemes. Next to this, they compare the computational costs of both schemes in a practical manner, by measuring the time it takes to securely transmit a message over LTE using the respective cryptographic schemes. Lastly, they analyse the impact of parallelisation techniques on the computational costs of both schemes.

In their 2017 paper [12], Singh and Patro compare multiple signcryption schemes. The signcryption schemes are compared on both their respective security attributes and theoretical computational costs. Additionally, they propose a generic approach for designing lightweight signcryption schemes.

Rezaeibagha et al. in their work [11] propose a signcryption scheme specifically designed for privacy preservation in IoT wireless sensor networks by using deniable authentication and homomorphic encryption. Additionally, they compared their scheme with other schemes based on deniable authentication of security attributes and performance. The schemes were implemented on a computer and their performance was tested practically.

In [8], Kumar et al. propose a lightweight signcryption scheme specifically for perception layer devices in IoT. The proposed scheme uses a lightweight hashing function. The research validates the security of the proposed scheme and analyses possible attacks against it. Moreover, the performance in terms of communication overhead, computational costs and energy consumption are analysed and compared to existing cryptographic schemes. The analyses were performed practically, using a network of smartphones as clients, laptops as servers and a gateway to connect them.

Lastly, in [13] a hybrid lightweight signcryption scheme for IoT is proposed. The scheme uses both asymmetric and symmetric cryptographic techniques in combination with an ultra-lightweight block cipher. The security of the proposed scheme is analysed and proven. Additionally, the performance is compared with two different signcryption schemes by implementing them and measuring their computational costs in practice.

3 BACKGROUND

In order to reason about the performance of signcryption vs sign-then-encrypt schemes, signcryption must first be explained in detail. First, the underlying mathematical problems behind signcryption are explained. Then, two versions of signcryption schemes proposed by Zheng in [19] and [20] are explained in detail. Lastly, a number of sign-then-encrypt schemes to compare signcryption against are chosen.

3.1 Underlying mathematical problems

The signcryption schemes explained below rely upon certain mathematical problems for their security. These mathematical problems are all easy to compute in one direction, but near impossible to compute inversely without knowing a certain fact. Therefore, these mathematical problems are often called 'trapdoor functions'. For the signcryption schemes chosen in this paper, these mathematical problems are used:

(1) Discrete logarithm problem (DLP)

(2) Elliptic curve cryptography (ECC)

3.1.1 Discrete logarithm problem. The discrete logarithm problem is based on the fact that it is very hard to find the exponent x used in the following equation:

$$a = g^x \bmod n \tag{1}$$

Where n is a large prime number and g is a prime root of n. Due to g being a prime root of n, calculating $a = g^x \bmod n$ with any exponent x results in all solutions a comprising the elements of a cyclic group G of order $n - 1$. This means that solution a is equally likely to be any integer between 1 and $n - 1$.

To calculate x, the following equation is used:

$$x = \log_g a \bmod n \tag{2}$$

However, this is difficult due to the fact that for any given a, multiple exponents x could have resulted in a. Moreover, the amount of guesses needed to find x grows exponentially with the size of n. Therefore, when a sufficiently large n is chosen, it is computationally impossible to find x in reasonable time. However, computing a with x is computationally easy, making it a good trapdoor function.

3.1.2 Elliptic curve cryptography. Elliptic curve cryptography was independently invented in 1985 by Miller [9] and Koblitz [7]. An elliptic curve is defined as a plane curve of a finite field, in which all points satisfy the following equation:

$$y^2 = x^3 + ax + b \tag{3}$$

A property of elliptic curves is that when adding two points, or doubling a single point, this results in a new point. Due to the properties of elliptic curves, the position of this new point seemingly has no relation to the positions of the points used to create it.

This means that when a single point P on the curve is multiplied with an integer n (addition to itself n times), the result is a point Q with no identifiable relation with P. Herein lies the mathematical problem. With a known P and Q, it is near impossible to determine n if n is sufficiently large. The advantage of ECC is that the size of n can be relatively much smaller than the key sizes needed for the DLP while offering an equally hard problem [17].

3.2 Signcryption

3.2.1 Traditional signcryption. Signcryption is a combination of public-key cryptography with digital signatures in a single logical step. Signcryption is based on the discrete logarithm problem (DLP). Signcryption in general has three different basic functions: Key Generation (KG), Signcryption (SC) and Unsigncryption (USC). The KG function is responsible for providing the appropriate keys to the user. SC is responsible for signing and encrypting the message, while USC does the opposite.

We will use the SDSS1-based signcryption scheme from [19]. x_a and y_a will refer to Alice the sender's key pair, while x_b and y_b will refer to Bob the receiver's key pair. Both key pairs come from the KG function. Encryption and decryption are done using a private key cipher, such as DES in CBC mode [19]. Encryption with a key is denoted as $encrypt(message, key)$ and decryption is denoted as $decrypt(message, key)$

Key generation

First, choose (p, q, g) so that they define a multiplicative subgroup of $\mathbb{Z}_p$.

$$(p, q, g) = \mathbb{Z}_p \tag{4}$$

p, q must be large primes, with q being of factor $p - 1$ and $1 < g < p - 1$ with order q. Using minimal sizes for p and q of 1024 & 160 bits ensures that the DLP is sufficiently hard enough. (p, q, g) must be public for all users of this signcryption scheme. Then, pick a random integer x

$$x = [1, ..., q - 1] \tag{5}$$

Lastly, calculate public key y

$$y = g^x \bmod p \tag{6}$$

SC by Alice

Pick a random integer x

$$x = [1, ..., q - 1] \tag{7}$$

Calculate k

$$k = y_b^x \bmod p \tag{8}$$

This means that

$$k = g^{x_b * x} \bmod p \tag{9}$$

k must then be split into two sub-keys, k_1 and k_2. k_1 is used as the encryption key and k_2 as the hash key. After this, calculate hash r

$$r = hash(m, k_2) \tag{10}$$

Then, calculate s

$$s = x/(x_a + r) \bmod q \tag{11}$$

Lastly, calculate ciphertext c

$$c = encrypt(m, k_1) \tag{12}$$

Then, (c, r, s) is transmitted to Bob by Alice.

USC by Bob

Derive k

$$k = (y_a * g^r)^{s * x_b} \bmod p \tag{13}$$

This follows because

$$y_a = g^x x_a$$
$$g^{x_a} * g^r = g^{x_a + r}$$
$$s = x/(x_a + r) \bmod q$$

Therefore

$$(g^{x_a + r})^{s * x_b} \bmod p = (g^{x_b})^x \bmod p = y_b^x \bmod p = k \tag{14}$$

Then, k must be split into k_1 and k_1 in the same fashion as the SC step. After this, the message m can be retrieved as follows:

$$m = decrypt(c, k_1) \tag{15}$$

After which, m can only be considered valid if

$$hash(m, k_2) = r \tag{16}$$

3.2.2 ECC Signcryption. In 1998, Zheng proposed two new signcryption schemes based on elliptic curve cryptography (ECC) [20]. The ECC signcryption process is largely the same as the traditional signcryption process, with the key-generation step being replaced by the step of determining the elliptic curve parameters. To determine the required parameters, the following steps have to be taken:

(1) Choose an elliptic curve C with the form $y^2 = x^3 + ax + b$ over a finite field $GF(p)$ where p is a large prime at least 160 bits in size.
(2) Choose q, a large prime of size $|p|$
(3) Choose a random point G on C, with G being of order q. This is the base point used in the multiplications needed with ECC.
(4) Choose a one-way hash function $hash$ with output size at least 128 bits. This will be used to compute the keys for encryption & signing.
(5) Choose a keyed one-way hash function KH. This will be used for the signature.
(6) Choose appropriate encryption ($encrypt$) & decryption algorithms ($decrypt$). These can be any private-key cipher.

In this explanation, a sender Alice wants to send a message m to Bob. Alice's key pair consists of private key v_a and public key P_a, as defined by the key generation procedure.

Key generation

Private key v_a chosen randomly

$$v_a = [1, ..., q - 1] \tag{17}$$

Public key P_a

$$P_a = v_a * G \tag{18}$$

Bob's key pair is (v_b, P_b), derived similarly. We assume both parties have exchanged their public keys. The following steps are based on the ECSC1 signcryption scheme proposed in [20].

SC by Alice

Choose a random number v

$$v = [1, ..., q - 1] \tag{19}$$

Calculate keys (k_1, k_2). k_1 and k_2 are derived by splitting the result of the following equation.

$$(k_1, k_2) = hash(v * P_b) \tag{20}$$

This links k_1 and k_2 to Bob's public key. Then, the ciphertext is calculated

$$c = encrypt(k_1, m) \tag{21}$$

After which, hash r is calculated

$$r = KH(k_2, m) \tag{22}$$

Then, signature s is calculated. This can be used by Bob to derive k_1 and k_2. Finally, Alice sends (c, r, s) to Bob.

USC by Bob

Derive temporary variable u

$$u = s * v_b \bmod q \tag{23}$$

Then, (k_1, k_2) is derived like this

$$(k_1, k_2) = hash(uP_a + urG) \tag{24}$$

This follows because

$$uP_a + urG = \tag{25}$$

$$u(P_a + rG) =$$

$$u(v_aG + rG) =$$

$$uG(v_a + r) =$$

$$s * v_b * G(v_a + r) =$$

$$s * P_b(v_a + r) =$$

$$(v/(r + v_a) * P_b * (v_a + r)) =$$

$$vP_b$$

Then, m can be calculated

$$m = decrypt(k_1, c) \tag{26}$$

After which, the validity of m is tested using this equation

$$KH(k_2, m) = r \tag{27}$$

3.3 Selected schemes

The traditional schemes chosen to compare with traditional signcryption are:

(1) RSA

(2) ElGamal encryption + DSA signature

(3) ElGamal encryption + Schnorr signature

These schemes were chosen because in Zheng's original paper [19], signcryption was compared in a theoretical manner with RSA, ElGamal encryption + DSA and ElGamal encryption + Schnorr. Therefore, they are good candidates to be compared practically with traditional signcryption. Moreover, all are well-known cryptographic schemes with existing implementations.

For the ECC-based schemes to compare with ECC signcryption, the chosen schemes are:

(1) ECC ElGamal encryption + ECDSA signature

(2) ECC ElGamal encryption + ECC Schnorr + signature

These schemes were chosen because they are the ECC-based counterparts of the selected traditional cryptographic schemes. Therefore, the performance of ECC and traditional cryptographic schemes can be compared without having to introduce completely new schemes. RSA is omitted in this category, as the underlying mathematical problem of RSA cannot be transformed to work with elliptic curves.

4 TEST ENVIRONMENT

4.1 Measurement tools

The implementation of the chosen schemes will be done using the charm-crypto library [1] and the Python programming language. The charm-crypto library is a Python library which internally uses PBC, OpenSSL and GMP to provide the arithmetic operations and security parameters used in cryptographic schemes.

The charm-crypto library was chosen because it contains numerous cryptographic primitives and helper functions which makes the implementation significantly easier. Moreover, a significant number of cryptographic schemes have already been implemented and are included in the library. The charm-crypto library also has included benchmarking functionality. This benchmarking functionality features the possibility to record all cryptographic operations (addition, subtraction etc) and records the duration of the chosen sequence. The Python programming language was chosen because it is a very well-known language which is fully compatible with the charm-crypto library. The used version of the charm-crypto library is 0.50 and the Python version is 3.7, as that is the latest version which works with the charm-crypto library.

4.2 Measurement environment

The device upon which the schemes will be implemented is a Raspberry PI 4B 2GB. The Raspberry PI 4B 2GB is a 1.5GHZ 4-core ARM-based small computer with 2GB of RAM. The Raspberry PI 4B uses the 04-04-2022 version of 32-bit Raspberry Pi OS. A Raspberry PI 4B was chosen because it provides complete computer functionality while remaining a relatively resource-constrained device. Moreover, the charm-crypto library and the Python programming language are compatible with the ARM architecture of the Raspberry Pi 4B and the Linux-based Raspberry Pi OS.

4.3 Methodology

Firstly, each selected cryptographic scheme is either manually implemented or uses an existing implementation. Then, for each scheme, a random 28-byte message is signed + encrypted/signcrypted. Then, the size of the resulting ciphertext is measured. After this, the ciphertext is unencrypted/unsigncrypted and it is verified that the derived plaintext is correct. This process is repeated for several iterations to get accurate measurements.

For the traditional cryptographic schemes, the signing algorithms use 256-bit keys and the encryption/signcryption algorithms use 2048-bit keys. The ECC-based schemes use the 256-bit P-256 curve. This curve was chosen as it is a very popular NIST-approved curve and is available in charm-crypto library.

The power draw of the Raspberry PI will be measured by using a USB power meter, situated between the Raspberry PI and its power supply. The USB power meter measures voltage, current and power draw in W and Wh. The computational cost will be measured by measuring how long a scheme takes to process the message. This is calculated as the total processing time divided by the number of iterations. Communication overhead is measured by comparing the original message size to the size of the ciphertext.

The code used to benchmark the selected schemes and generate the results in the following section can be found at https://github.com/K0enM/ResearchProject. A guide to setting up the charm-crypto library can be found at https://lrusso96.github.io/blog/cryptography/2021/03/04/charm-setup.html.

5 RESULTS

5.1 Comparative analysis

Table 1 shows the results of the comparative analysis. The comparative analysis consists of comparing the chosen cryptographic schemes on computational costs, communication overhead and power draw. This data is retrieved from relevant literature. For the traditional signcryption and sign-then-encrypt schemes, the data was retrieved from [19]. For the ECC-based schemes, the data was retrieved from [20], [2], [18] and [3] for signcryption, ECC ElGamal encryption, ECDSA and ECC Schnorr respectively.

Computational cost is measured in the number of modular mathematical operations for the traditional schemes, with point multiplications added for the ECC-based schemes. Communication overhead is represented as an equation, where the variables are the relevant parameters of the cryptographic scheme. No power draw data could be found in existing literature for the combinations of signature and encryption schemes and the signcryption schemes chosen.

5.2 Test results

Table 2 contains the collected data for the traditional signcryption and sign-then-encrypt schemes. Table 3 contains the data for the ECC-based signcryption and sign-then-encrypt schemes. In these tables, for each cryptographic scheme, the computational cost, power total, power average, # of iterations, total running time, and communication overhead is displayed.

Figure 1 shows the computational cost for the traditional schemes, while figure 2 shows it for the ECC-based schemes. Figures 3 and 4 show the average power draw of the traditional and ECC-based schemes respectively. Next, figures 5 and 6 show the total power draw of the traditional and ECC-based schemes. Lastly, figures 7 and 8 visualize the communication overhead.

6 ANALYSIS

It can be seen that both forms of signcryption are faster than the sign-then-encrypt schemes while simultaneously drawing less power. This seems to confirm the notion that signcryption is both faster and more efficient. However, these performance differences do not match up exactly with the theoretical performance gains. For traditional signcryption, the performance difference with RSA is only 12.7%, but for the ElGamal-based sign-then-encrypt schemes the differences are 73% and 83% respectively as can be seen in figure 1. This is different from the literature, which suggests a 58% difference in computational costs. ECC-based signcryption also is not as fast as theoretically proposed, with the differences being 34% and 35% respectively, seen in figure 2.

Secondly, the power draw of signcryption compared to the sign-then-encrypt schemes is interesting. Both forms of signcryption seem to

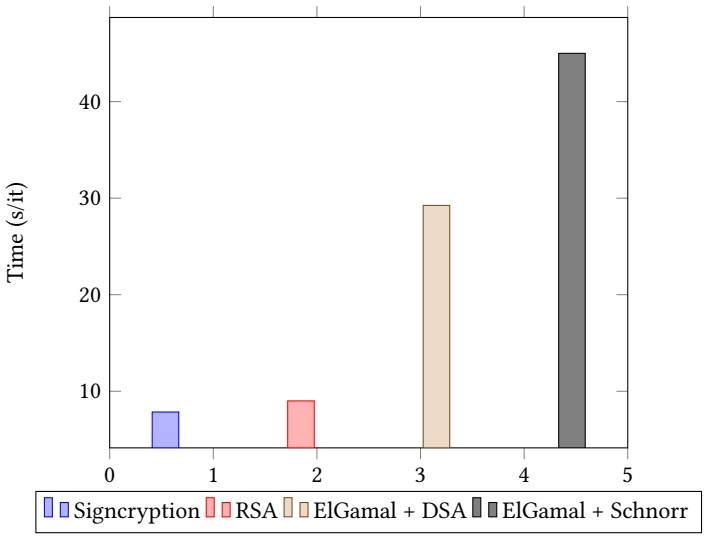

Fig. 1. Computational cost of traditional schemes

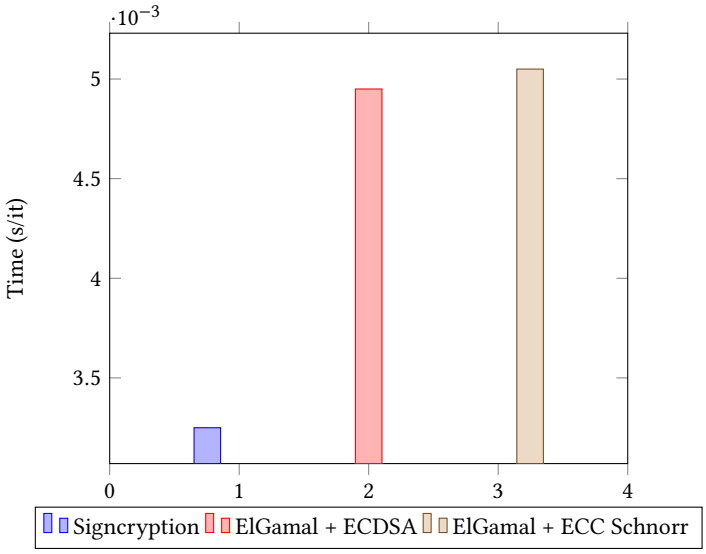

Fig. 2. Computational cost of ECC-based schemes

use the same or slightly more power on average when compared with the other schemes, as can be seen in figures 3 and 4. Moreover, ECC-based signcryption draws the highest average power of all ECC-based schemes. However, as showed in figures 5 and 6, signcryption draws significantly less total power than the sign-then-encrypt schemes. This correlates with the theory behind signcryption of one single computationally intensive step (responsible for peak power), instead of two, which results in less total power being drawn.

Thirdly, in figures it is easily noticed that both ECC-based signcryption and sign-then-encrypt schemes are significantly faster than their traditional counterparts, while still offering the same level of

Table 1. Comparative analysis of signcryption & sign-then-encrypt schemes.

Algorithm	Power Draw	Computational cost	Communication overhead
Signcryption	-	EXP=3, MUL=2, DIV=1, ADD=1, SUB=0	$hash(m) + \|q\|$
RSA - RSA	-	EXP=3, MUL=0, DIV=0, ADD=0, SUB=0	$\|n_a\| + \|n_b\|$
ElGamal - DSA	-	EXP=6, MUL=2, DIV=3, ADD=1, SUB=0	$2\|q\| + \|p\|$
ElGamal - Schnorr	-	EXP=6, MUL=2, DIV=0, ADD=1, SUB=0	$hash(m) + \|q\| + \|p\|$
Signcryption (ECC)	-	PMUL=3, MUL=2, DIV=1, ADD=1, SUB=0	$KH(m) + \|q\|$
ElGamal (ECC) - ECDSA	-	PMUL=6, MUL=3, DIV=1, ADD=1, SUB=0	$hash(m) + 2\|q\|$
ElGamal (ECC) - Schnorr (ECC)	-	PMUL=6, MUL=1, DIV=0, ADD=1, SUB=1	$hash(m) + 2\|q\|$

EXP = number of modular exponentiations, MUL = number of modular multiplications, DIV = number of modular divisions, ADD = number of modular additions, SUB = number of modular subtractions, PMUL = number of point multiplications

Table 2. Performance comparison of traditional signcryption & sign-then-encrypt schemes

Scheme	Comp. cost (s/it)	Power total (Wh)	Power average (W)	Iterations	Total time (mm:ss)	Comm. overhead
Signcryption	7.85	0.116	2.729	20	2:37	1957%
RSA	9	0.145	2.899	20	3:01	1728%
ElGamal + DSA	29.25	0.441	2.714	20	9:45	1728%
ElGamal + Schnorr	45	0.676	2.704	20	14:59	1728%

Table 3. Performance comparison of ECC-based signcryption & sign-then-encrypt schemes

Scheme	Comp. cost (s/it)	Power total (Wh)	Power average (W)	Iterations	Total time (mm:ss)	Comm. overhead
Signcryption (ECC)	0.00325	0.062	3.434	20000	01:05	357%
ElGamal (ECC) + ECDSA	0.00495	0.093	3.452	20000	1:37	228%
ElGamal (ECC) + Schnorr (ECC)	0.00505	0.095	3.386	20000	01:41	228%

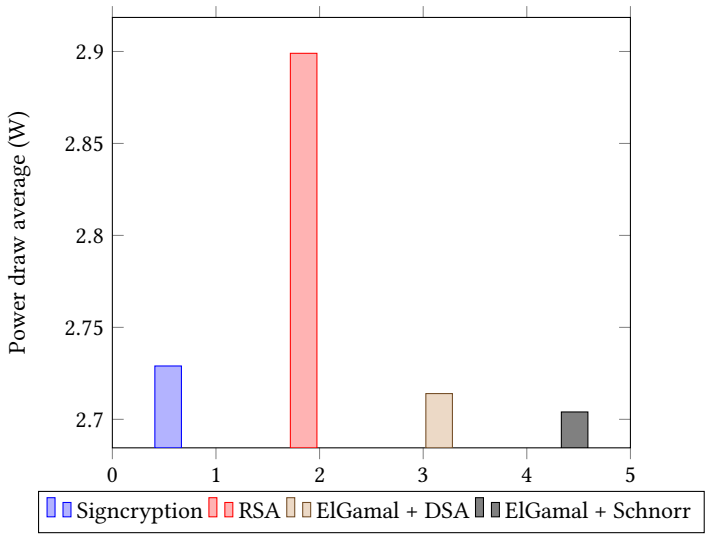

Fig. 3. Average power draw of traditional schemes

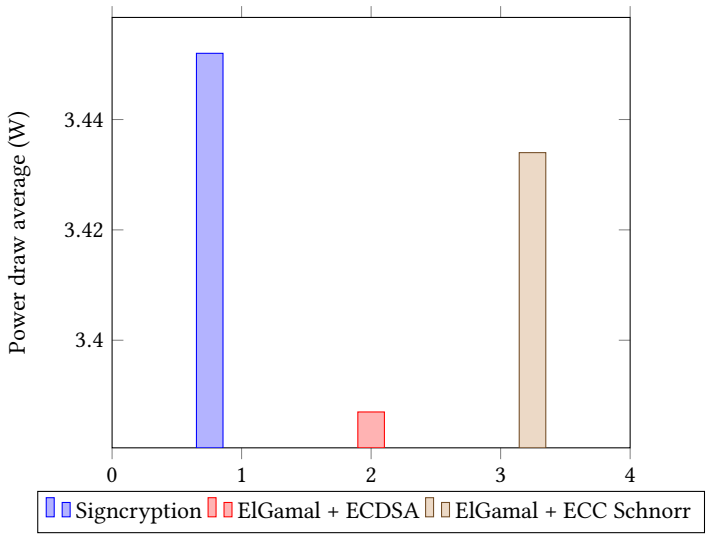

Fig. 4. Average power draw of ECC-based schemes

security. Moreover, their communication overhead is also significantly lower because of their use of smaller key-sizes, as can be seen in figures 7 and 8.

7 CONCLUSION

Based on figures 1 and 2, it can be seen that the computational costs of signcryption schemes are lower than those of equivalent sign-then-encrypt schemes. Therefore, RQ1 can be answered by

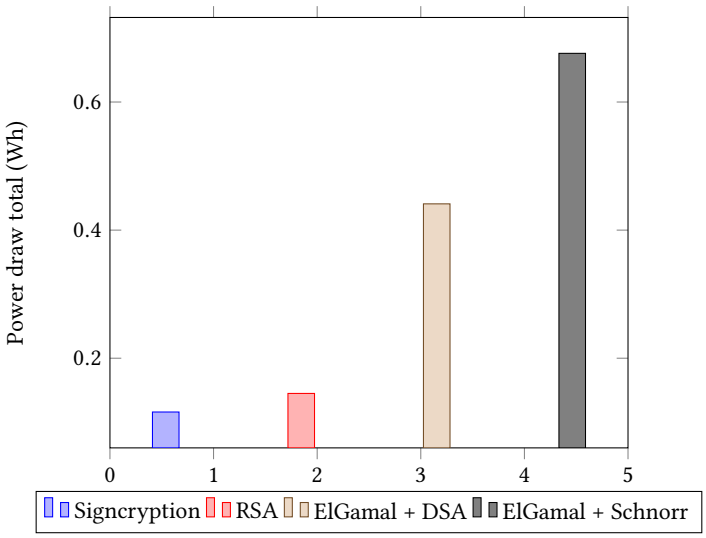

Fig. 5. Total power draw of traditional schemes

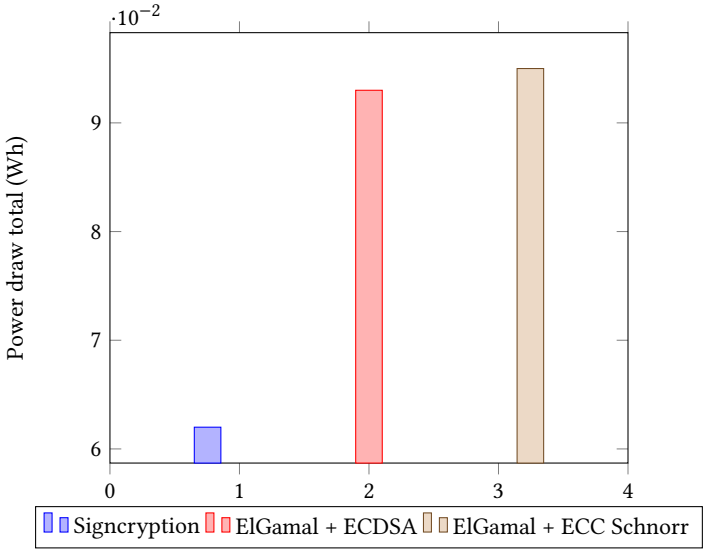

Fig. 6. Total power draw of ECC-based schemes

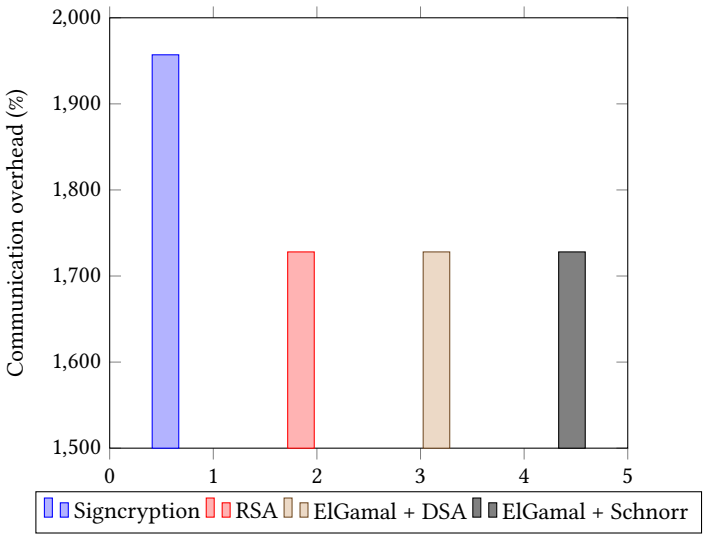

Fig. 7. Communication overhead of traditional schemes

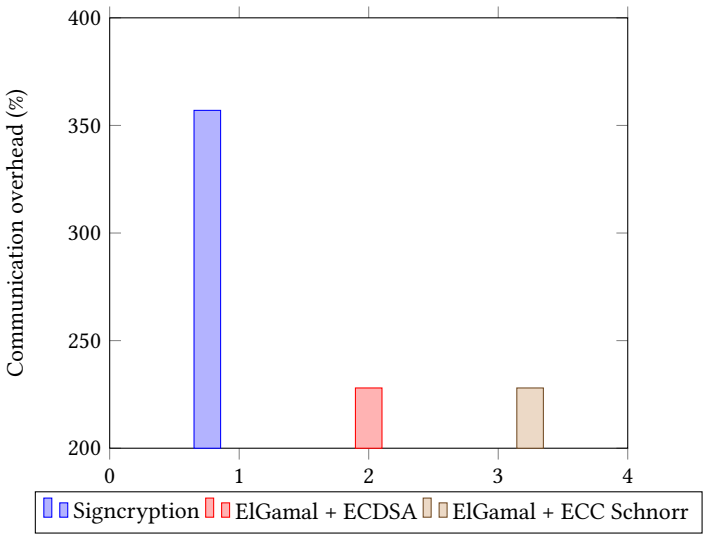

Fig. 8. Communication overhead of ECC-based schemes

concluding that signcryption schemes do indeed have better computational costs when measured practically.

Answering RQ2, it can be concluded from figures 3, 4, 5 and 6 that signcryption schemes consume less power in total compared to sign-then-encrypt schemes, but their average power draw is equivalent.

Finally, based on figures 7 and 8, signcryption, it can be seen that with this implementation, signcryption in practice has higher communication overhead than sign-then-encrypt schemes. This is because signcryption transmits both the symmetrically encrypted message + 2 large integers and the other sign-then-encrypt schemes only need to transmit 2 integers. Therefore, RQ3 can be answered by concluding that signcryption does not necessarily provide better communication overhead when measured practically.

7.1 Recommendation

Based on the results and analysis of this research, we recommend that anyone needing to transmit confidential data on resource-constrained devices use ECC-based signcryption or sign-then-encrypt schemes. ECC-based schemes are significantly faster while offering a high enough level of security. Additionally, we recommend if performance is of the utmost importance, to use a signcryption scheme, as based on this research it is even faster while using fewer resources. However, there are no implementations of signcryption

available in widely-used libraries or standards, so for most use cases, ECC-based sign-then-encrypt schemes will be the best option.

REFERENCES

[1] Joseph A. Akinyele, Christina Garman, Ian Miers, Matthew W. Pagano, Michael Rushanan, Matthew Green, and Aviel D. Rubin. 2013. Charm: A framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering* 3, 2 (6 2013), 111–128. https://doi.org/10.1007/S13389-013-0057-3

[2] J.J. Botes and W.T. Penzhorn. 1994. An implementation of an elliptic curve cryptosystem. *Proceedings of COMSIG '94 - 1994 South African Symposium on Communications and Signal Processing* (10 1994), 85–90. https://doi.org/10.1109/COMSIG.1994.512441

[3] Bundesamt für Sicherheit in der Informationstechnik. 2018. Elliptic Curve Cryptography. *Technical Guideline BSI TR-0311* 2.10 (2018), 33–33. http://www.bsi.bund.de

[4] Sumit Singh Dhanda, Brahmjit Singh, and Poonam Jindal. 2020. Lightweight Cryptography: A Solution to Secure IoT. *Wireless Personal Communications* 112, 3 (6 2020), 1947–1980. https://doi.org/10.1007/S11277-020-07134-3

[5] Aya Elshobaky, Ghada Elkabbany, Mohamed Rasslan, and Shawkat Gurguis. 2014. Implementation, Comparison, and Enhancement of Secure Communication Designs. *Procedia Computer Science* 37 (1 2014), 363–369. https://doi.org/10.1016/J.PROCS.2014.08.054

[6] Bei Gong, Yong Wu, Qian Wang, Yu heng Ren, and Chong Guo. 2022. A secure and lightweight certificateless hybrid signcryption scheme for Internet of Things. *Future Generation Computer Systems* 127 (2 2022), 23–30. https://doi.org/10.1016/J.FUTURE.2021.08.027

[7] Neal Koblitz. 1987. Elliptic curve cryptosystems. *Math. Comp.* 48, 177 (1987), 203–209. https://doi.org/10.1090/S0025-5718-1987-0866109-5

[8] Ashish Kumar, Rahul Saha, Mamoun Alazab, and Gulshan Kumar. 2020. A Lightweight Signcryption Method for Perception Layer in Internet-of-Things. *Journal of Information Security and Applications* 55 (12 2020), 102662. https://doi.org/10.1016/J.JISA.2020.102662

[9] Victor S. Miller. 1986. Use of Elliptic Curves in Cryptography. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 218 LNCS (1986), 417–426. https://doi.org/10.1007/3-540-39799-X_31

[10] Mardiana binti Mohamad Noor and Wan Haslina Hassan. 2019. Current research on Internet of Things (IoT) security: A survey. *Computer Networks* 148 (1 2019), 283–294. https://doi.org/10.1016/J.COMNET.2018.11.025

[11] Fatemeh Rezaeibagha, Yi Mu, Ke Huang, Leyou Zhang, and Xinyi Huang. 2021. Secure and Privacy-Preserved Data Collection for IoT Wireless Sensors. *IEEE Internet of Things Journal* 8, 24 (12 2021), 17669–17677. https://doi.org/10.1109/JIOT.2021.3082150

[12] Anuj Kumar Singh and Patro B.D.K. 2017. Performance Comparison of Signcryption Schemes – A Step towards Designing Lightweight Cryptographic Mechanism. *International Journal of Engineering and Technology* 9, 2 (4 2017), 1163–1170. https://doi.org/10.21817/IJET/2017/V9I2/170902173

[13] M. Sruthi and Rajkumar Rajasekaran. 2021. Hybrid lightweight Signcryption scheme for IoT. *Open Computer Science* 11, 1 (1 2021), 391–398. https://doi.org/10.1515/comp-2020-0105

[14] S. R. Subramanya and Byung K. Yi. 2006. Digital signatures. *IEEE Potentials* 25, 2 (2006), 5–8. https://doi.org/10.1109/MP.2006.1649003

[15] Melanie Swan. 2012. Sensor mania! the internet of things, wearable computing, objective metrics, and the quantified self 2.0. *Journal of Sensor and Actuator Networks* 1, 3 (12 2012), 217–253. https://doi.org/10.3390/JSAN1030217

[16] Nipun Balan Thekkummal, Devki Nandan Jha, Deepak Puthal, Philip James, and Rajiv Ranjan. 2020. Coordinated Data Flow Control in IoT Networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 12041 LNCS (2020), 25–41. https://doi.org/10.1007/978-3-030-58628-7_3

[17] Naoya Torii and Kazuhiro Yokoyama. 1987. Elliptic curve cryptosystems. *Math. Comp.* 48, 177 (1987), 203–209. https://doi.org/10.1090/S0025-5718-1987-0866109-5

[18] S. Vanstone. 1992. Responses to NIST's Proposal. *Commun. ACM* 35, 7 (1992), 50–52.

[19] Yuliang Zheng. 1997. Digital signcryption or how to achieve cost(signature & encryption) « cost(signature) + cost(encryption). *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 1294 (1997), 165–179. https://doi.org/10.1007/BFB0052234

[20] Yuliang Zheng and Hideki Imai. 1998. How to construct efficient signcryption schemes on elliptic curves. *Inform. Process. Lett.* 68, 5 (12 1998), 227–233. https://doi.org/10.1016/S0020-0190(98)00167-7