ELLIPIC-CURVE CRYPTOGRAPHY (ECC) AND ARGON2 ALGORITHM IN PHP USING OPENSSL AND SODIUM LIBRARIES

Mariusz Duka
Silesian University of Technology, Faculty of Automatic Control, Electronic and Computer Science, Gliwice, Poland

Abstract. This paper presents the elliptic-curve cryptography (ECC) and Argon2 algorithm in PHP using OpenSSL and Sodium cryptographic libraries. The vital part of this thesis presents an analysis of the efficiency of elliptic-curve cryptography (ECC) and the Argon2 hashing algorithm in the Sodium library, depending on the variation of initiation parameters.

Keywords: elliptic-curve cryptography, ECC, OpenSSL, Argon2, Sodium

KRYPTOGRAFIA KRZYWYCH ELLIPTYCZNYCH (ECC) I ALGORYTM ARGON2 W JEZYKU PHP Z WYKORZYSTANIEM BIBLIOTEKI OPENSSL I SODIUM

Streszczenie. Celem niniejszej pracy jest analiza wydajności kryptografii krzywych eliptycznych (ECC) i wskazanie optymalnej krzywej dla kryptosystemu wykorzystującego język PHP wraz z biblioteką OpenSSL, a także analiza wydajności algorytmu haszącego Argon2, wchodzącego w skład biblioteki Sodium, w zależności od zmienności parametrów inicjujących.

Słowa kluczowe: kryptografia krzywych eliptycznych, ECC, OpenSSL, Argon2, Sodium

Introduction

The information security, its storage and dissemination is a crucial element of any IT system, regardless of what data the system processes and on what technology it is based on. The Internet users of these IT systems expect the information they provide, especially sensitive information, to be adequately protected.

Currently available programming languages, such as Java, C, Python, Ruby, or PHP are used to build modern IT systems. The choice for a particular programming language depends on many factors, such as the requirements set for programmers, available programming libraries, technical support or its popularity at a time.

The scripting programming language, PHP, was designed to create web applications and generate websites on the server side. For the last 20 years PHP has been extensively developed and is currently the leader (almost 80% [4]) amongst the programming languages, on basis of which all websites are built. The popularity of the PHP language among the programmers around the world is enormous, as evidenced by the ninth position in the TIOBE index [5] (October 2019) and the sixth position in the SPECTRUM [6] ranking (2018). The PHP potential has also been recognised by such Internet giants as Google, Facebook, Yahoo!, Wikipedia or WordPress.

When designing a PHP based IT system to process information, a web developer is equipped with a number of modules and features to facilitate the information security, e.g. the OpenSSL library, that also uses, cryptography of elliptic curves or currently, the RSA algorithm. With PHP 7.2 version, the Sodium cross – platform library and the new Argon2 hashing algorithm are available to programmers. It makes PHP language a safe environment that implements the latest cryptography solutions.

This paper presents the findings related to the efficiency of generating cryptographic key using a cryptography of elliptic curves and efficiency of the Argon2 algorithm, depending on the variability of initiating parameters.

The conducted experiments related to the elliptic curve cryptography in IT systems based on PHP 7.2 were aimed at analysing which elliptic curves were the most efficient and could be recommended for using in the production environment.

1. Elliptic – Curve Cryptography (ECC)

Since 1985, the theory of elliptic curves over finite bodies has been applied to several cryptographic issues, such as the distribution of integers into prime factors, primality tests, and the construction of cryptosystems. One of the main reasons for the interest in the cryptology in elliptic curves is that they are the source of a vast number of finite groups equipped with a rich additional algebraic structure. The groups of elliptic curve points are in many aspects similar to multiplicative finite body groups. However, they have two advantages over them – there are more of them and it seems that they provide the same degree of security with a short key length [2, 3].

The cryptosystems using elliptic curves were first suggested in 1985 by Victor Miller and Neal Koblitz. Initially, they did not expect that their idea would be used in practical terms, in any case, not earlier than in the distant future. Today, a few years after they had submitted their project, many useful implementations have been in use [2].

The Table 1 shows the recommendations of the National Institute for Standards and Technology (NIST) regarding the public key length. As we can see, the conventional keys of 1024-byte and 2048-byte size (value often found in the case of RSA) corresponds to cryptosystems based on elliptic curves with 160-223-byte and 224-255-byte key sizes, respectively. In a fair comparison, the complexity of implementing the cryptosystem should also be taken into account. In practice, however, shorter keys can translate into faster implementations, lower energy consumption, smaller surfaces of silicon, etc. [1].

1.1. The optimal elliptic curve for the cryptosystem

The use of elliptic curves increases the efficiency of cryptographic systems compared to other public key systems, such as RSA or Diffie – Hellman. It takes into account the difference in the key length (Table 1), while maintaining the same level of security. The selection of optimal elliptic curves requires measuring the generation time, taking into account the key length for the individual types of curves recommended by the international organisations, i.e. Brainpool curves (according to Brainpool), Prime, C2pnb, C2nb (according to ANSI), Scop and Sect curves (according to SECG). As a result, Prime elliptic curves, which achieved a ratio of 2.03%, surpassed all other curves in terms of efficiency.
The experiment was carried out on a Dell PowerEdge 1950 server – with a typical workday load (Web server), with the following parameters:
- processor – Intel Xeon E5430 @ 2.66 GHz, 8CPU; 8GB RAM;
- operating system – FreeBSD 11.1 – RELEASE-p10;
- Software:
  - PHP 7.2; php72-openssl-7.2.5;
  - OpenSSL 1.0.2.

Cryptography based on elliptic curves, i.e. Brainpool, Prime C2pnb, Secp and Sect, is available in PHP since its 7.1 version, and the OpenSSL library. There are many types of elliptical curves recommended by the international standardisation organisations or communities. The American National Standard Institute (ANSI) recommends Prime, C2pnb, C2nb elliptic curves, Standard for Efficient Cryptography Group (SECG) recommends Secp and Sect elliptic curves or Brainpool Group, which recommends Brainpool elliptic curves.

1.2. The course of the experiment

In order to measure the efficiency of the key generation by using elliptic curves, a PHP 7.2 programming language, using the OpenSSL cryptographic library, has been developed. The experiment involved the measurement of time to generate 100 keys for each elliptic curve separately, during a typical operation of the production server.

The efficiency, marked as a performance ratio and expressed as a percentage for each elliptic curve, was calculated by dividing the average generation time by the key length of the elliptic curve. At the end of the experiment, the key performance for each group of elliptic curves was determined by dividing the performance ratio by the number of elliptic curves for that group.

1.3. The outcome of the experiment

We can conclude, from the data in Tables 3-8, that the larger the key size, the more time it takes to generate it. Specifically for curves with the key length of 384-511 bytes, the time needed to generate it increases significantly, e.g. for the elliptic curve sect571r1 the time needed was almost 56 milliseconds. PHP Group, which manages the implementation and development of the PHP programming language, based on the NIST recommendation, suggests the use of elliptic curves prime256v1 (NIST P-256) and secp384rl (NIST P-384).

---

Table 2. Elliptic curves available in PHP

| Name of the organization | Elliptic Curve Name | The length of key in bytes |
|--------------------------|--------------------|---------------------------|
| Brainpool Group          | brainpoolP160r1    | 160                       |
|                          | brainpoolP192r1    | 192                       |
|                          | brainpoolP224r1    | 224                       |
|                          | brainpoolP239r1    | 239                       |
|                          | brainpoolP256r1    | 256                       |
|                          | brainpoolP384r1    | 384                       |
|                          | brainpoolP384r2    | 384                       |
|                          | brainpoolP512r1    | 512                       |
| ANSI                     | prime12r1          | 12                        |
|                          | prime12r2          | 12                        |
|                          | prime192r1         | 192                       |
|                          | prime192r2         | 192                       |
|                          | prime239r1         | 239                       |
|                          | prime239r2         | 239                       |
|                          | prime256r1         | 256                       |
|                          | prime384r1         | 384                       |
|                          | prime384r2         | 384                       |
|                          | prime384r3         | 384                       |
|                          | prime512r1         | 512                       |
| SECG                     | secp128r1          | 12                        |
|                          | secp128r2          | 12                        |
|                          | secp160r1          | 160                       |
|                          | secp160r2          | 160                       |
|                          | secp224r1          | 224                       |
|                          | secp224r2          | 224                       |
|                          | secp256r1          | 256                       |
|                          | secp384r1          | 384                       |
|                          | secp384r2          | 384                       |
|                          | secp384r3          | 384                       |
|                          | secp512r1          | 512                       |
|                          | secp113r1          | 113                       |
|                          | secp113r2          | 113                       |
|                          | secp113r3          | 113                       |
|                          | secp163k1          | 163                       |
|                          | sec233k1           | 233                       |
|                          | sec233k2           | 233                       |
|                          | sec239k1           | 239                       |
|                          | sec239k2           | 239                       |
|                          | sec283k1           | 283                       |
|                          | sec409k1           | 409                       |
|                          | sec409k2           | 409                       |
|                          | sec511r1           | 511                       |
|                          | sec571r1           | 571                       |

The experiment (ANSI) recommends Prime, C2pnb, C2nb elliptic curves, Standard for Efficient Cryptography Group (SECG) recommends Secp and Sect elliptic curves or Brainpool Group, which recommends Brainpool elliptic curves.

---

Table 3. Average performance ratio for Brainpool elliptic curves

| Elliptic Curve Name | The length of key in bytes (A) | Average time to generate keys (in milliseconds) (B) | Ratio in % (B/A) |
|---------------------|--------------------------------|---------------------------------------------------|-----------------|
| BrainpoolP160r1     | 160                            | 4.03                                              | 2.52            |
| BrainpoolP192r1     | 192                            | 4.38                                              | 2.28            |
| BrainpoolP224r1     | 224                            | 5.26                                              | 2.35            |
| BrainpoolP256r1     | 256                            | 5.68                                              | 2.22            |
| BrainpoolP384r1     | 384                            | 5.50                                              | 2.15            |
| BrainpoolP512r1     | 512                            | 13.46                                             | 2.63            |

Average ratio in %: 2.38

Table 4. Average performance ratio for Prime elliptic curves

| Elliptic Curve Name | The length of key in bytes (A) | Average time to generate keys (in milliseconds) (B) | Ratio in % (B/A) |
|---------------------|--------------------------------|---------------------------------------------------|-----------------|
| prime12r1           | 12                             | 4.20                                              | 2.19            |
| prime192r2          | 192                            | 4.21                                              | 2.19            |
| prime192r3          | 192                            | 4.21                                              | 2.19            |
| prime239r2          | 239                            | 5.19                                              | 2.17            |
| prime239r3          | 239                            | 5.19                                              | 2.17            |
| prime256v1          | 256                            | 2.85                                              | 1.11            |

Average ratio in %: 2.03

Table 5. Average performance ratio for C2pnb elliptic curves performance

| Elliptic Curve Name | The length of key in bytes (A) | Average time to generate keys (in milliseconds) (B) | Ratio in % (B/A) |
|---------------------|--------------------------------|---------------------------------------------------|-----------------|
| c2pnb163v1          | 163                            | 5.85                                              | 3.59            |
| c2pnb163v3          | 163                            | 5.84                                              | 3.58            |
| c2pnb163v5          | 163                            | 5.83                                              | 3.58            |
| c2pnb176v1          | 176                            | 5.79                                              | 3.29            |
| c2pnb208w1          | 208                            | 6.74                                              | 3.24            |
| c2pnb232v1          | 232                            | 12.15                                             | 4.47            |
| c2pnb304w1          | 304                            | 13.61                                             | 4.48            |
| c2pnb368w1          | 368                            | 16.93                                             | 4.60            |

Average ratio in %: 3.85
Table 6. Average performance ratio for C2nb elliptic curves

| Elliptic Curve Name | The length of key in bytes(A) | Average time to generate keys (in milliseconds) (B) | Ratio in % (B/A) |
|---------------------|-------------------------------|---------------------------------------------------|-----------------|
| c2nb191v1           | 191                           | 6.23                                              | 3.26            |
| c2nb191v2           | 191                           | 6.19                                              | 3.24            |
| c2nb191v3           | 191                           | 6.18                                              | 3.23            |
| c2nb239v1           | 239                           | 7.51                                              | 3.14            |
| c2nb239v2           | 239                           | 7.49                                              | 3.13            |
| c2nb239v3           | 239                           | 7.45                                              | 3.12            |
| c2nb150v1           | 159                           | 16.02                                             | 4.46            |
| c2nb431r1           | 431                           | 26.38                                             | 6.12            |

Average ratio in %: 3.71

Table 7. Average performance ratio for Secp elliptic curves

| Elliptic Curve Name | The length of key in bytes(A) | Average time to generate keys (in milliseconds) (B) | Ratio in % (B/A) |
|---------------------|-------------------------------|---------------------------------------------------|-----------------|
| secp112r1           | 112                           | 3.29                                              | 2.94            |
| secp112r2           | 112                           | 3.30                                              | 2.95            |
| secp128r1           | 128                           | 3.36                                              | 2.62            |
| secp160r1           | 160                           | 3.38                                              | 2.64            |
| secp160r2           | 160                           | 4.02                                              | 2.51            |
| secp192r1           | 192                           | 3.92                                              | 2.45            |
| secp224r1           | 224                           | 4.15                                              | 2.27            |
| secp283r1           | 283                           | 4.99                                              | 2.23            |
| secp384r1           | 384                           | 8.94                                              | 2.33            |
| secp521r1           | 521                           | 16.92                                             | 2.55            |

Average ratio in %: 3.71

Table 8. Average performance ratio for Sect elliptic curves

| Elliptic Curve Name | The length of key in bytes(A) | Average time to generate keys (in milliseconds) (B) | Ratio in % (B/A) |
|---------------------|-------------------------------|---------------------------------------------------|-----------------|
| secp113r1           | 113                           | 3.48                                              | 3.08            |
| secp113r2           | 113                           | 3.47                                              | 3.01            |
| secp128r1           | 128                           | 4.81                                              | 3.62            |
| secp137r2           | 131                           | 4.88                                              | 3.72            |
| secp163v1           | 163                           | 5.66                                              | 3.47            |
| secp163v2           | 163                           | 5.81                                              | 3.57            |
| secp163v3           | 163                           | 5.83                                              | 3.58            |
| secp193r1           | 193                           | 6.56                                              | 3.40            |
| secp193r2           | 193                           | 6.55                                              | 3.39            |
| secp223r1           | 223                           | 7.34                                              | 3.01            |
| secp223r2           | 223                           | 7.36                                              | 3.16            |
| secp259r1           | 259                           | 7.09                                              | 2.97            |
| secp283r1           | 283                           | 12.24                                             | 4.33            |
| secp384r1           | 384                           | 24.22                                             | 9.52            |
| secp409r1           | 409                           | 26.57                                             | 6.50            |
| secp511r1           | 511                           | 50.14                                             | 8.78            |
| secp521r1           | 521                           | 55.96                                             | 9.80            |

Average ratio in %: 4.45

Fig. 1. Average performance ratio for all analysed groups of elliptic curves

The interesting conclusion of the experiment was the demonstration that Brainpool curves do not differ in efficiency from those recommended by the PHP Group as the most efficient elliptic curves Prime and Secp, and are in the second position between Prime and Secp in the ranking of average performance.

2. Argon2 algorithm

The Argon2 algorithm became the winner in the contest organised by the Password Hashing Competition [7], lasting from 2013 to 2015. The idea of the contest was to select a hashing algorithm that would be in line with the requirements of the National Institute for Standards and Technology (NIST) in the United States, for AES and SHA-3 algorithm, but more effective and sophisticated. Of the 24 projects submitted, it was the project by Alex Biryukov, Daniel Dinu and Dmitry Khovratovich from the University of Luxembourg in Luxembourg that won the most recognition. The algorithm has been made available on the Internet [7] in the form of the source code in C language.

Argon2 protects against brute-force attacks by using a predefined memory size, GPU usage time and an appropriate degree of Instruction Level Parallelism. It uses three parameters to control: the memory required, execution time and the number of threads used. There are two variants of this algorithm: Argon2i and Argon2d. Argon2i is effective against side-channel attacks because it uses data independent access to memory, which is why it is well suited for password hashing. Argon2d provides better protection against GPU-based attacks.

The Argon2i hashing algorithm has been implemented in PHP, starting from the 7.2 version, as part of the Sodium library. A range of Argon2 algorithms is used in tasks requiring massive storage capacity. It is optimised for x86 architecture and works efficiently on older processors. The key feature of Argon2 is the simultaneous use of multiple processor cores, which, consequently, is an additional protection that prevents a time/memory/date/trade off attack type of intrusion.

Argon2 uses optimally all available memory, Argon2i uses the memory at a speed of two processor cycles per byte, while Argon2d is three times faster. These features, among others, are included in the software used for cryptocurrencies. Argon2 is exceptionally scalable, both in terms of memory and CPU usage, because it can use up to 224 threads simultaneously.

2.1. The efficiency of Argon2 algorithm

The efficiency of Argon2 algorithm depends on the technical conditions in which it functions. Primarily, on the speed of the processor and the memory acquired. The performance evaluation will be basically to measure the algorithm’s workload time dependent on the initialised parameters. The experiment is carried out on a Dell PowerEdge 1950 server – with a typical workload load (Web server), with the following parameters:

- processor – Intel Xeon E5430 @2.66 GHz, 8CPU; 8GB RAM;
- operating system – FreeBSD 11.1 – RELEASE-p10;
- Software:
  - PHP 7.2; php72-sodium-7.2.5;
  - Sodium library – libsodium – 1.0.16.

2.2. The course of the experiment

In the tcsh shell, the Argon2 program was cyclically called with parameters determining the memory acquired, the number of threads and the number of iterations, with the output key length set for all tests to 32 bytes.

The testing procedure had the following pattern:
```
echo -n "PSAEI_w_Gliwicach" | argon2 MySaltPSAEI -v 13 -m 14 -l 32 -i -x 1 -p 1
```
where MySaltPSAEI – the first parameter specifying grain (salt), v – algorithm version number (default 13), m – memory usage in kilobytes, calculated as 2^N, l – length of the shortcut at the output, i or d – Argon2i algorithm used or Argon2d, t – number of iteration, p – number of threads.

Figure 2 shows the results of a typical application of the Argon2 algorithm. As we can see, the algorithm’s running time in a single iteration was 0.055 seconds, using 16MB of memory, with the number of threads being one.
The next stage of the experiment was the cyclical calling of the hash function from the PHP programming language, using the Argon2i algorithm. The number of iterations for each type of measurement was 10, with the results divided by the number of threads and the amount of the memory used.

| Thread | Memory in MB | Argon2i (1 iteration) in seconds | Argon2i (1 iterations) in seconds |
|--------|--------------|---------------------------------|----------------------------------|
| 1      | 16           | 0.031                           | 0.094                            |
| 2      | 16           | 0.078                           | 0.172                            |
| 4      | 128          | 0.172                           | 0.383                            |
| 1      | 128          | 0.359                           | 0.723                            |
| 3      | 256          | 0.719                           | 1.594                            |
| 5      | 512          | 1.445                           | 3.227                            |
| 2      | 1024         | 2.914                           | 6.489                            |
| 3      | 256          | 0.039                           | 0.125                            |
| 1      | 64           | 0.109                           | 0.219                            |
| 2      | 64           | 0.211                           | 0.453                            |
| 2      | 128          | 0.445                           | 0.914                            |
| 2      | 256          | 0.914                           | 1.828                            |
| 2      | 512          | 1.828                           | 3.688                            |
| 2      | 1024         | 3.594                           | 7.492                            |
| 3      | 64           | 0.180                           | 0.328                            |
| 3      | 64           | 0.375                           | 0.664                            |
| 3      | 128          | 0.773                           | 1.328                            |
| 2      | 256          | 1.664                           | 2.773                            |
| 2      | 512          | 3.544                           | 5.516                            |
| 2      | 1024         | 6.905                           | 11.172                           |
| 3      | 16           | 0.164                           | 0.258                            |
| 4      | 32           | 0.414                           | 0.516                            |
| 4      | 64           | 0.633                           | 0.984                            |
| 4      | 128          | 1.297                           | 2.156                            |
| 4      | 256          | 2.648                           | 4.461                            |
| 4      | 512          | 5.148                           | 9.078                            |
| 4      | 1024         | 10.758                          | 18.383                           |

3. Conclusion

The first aim was to analyse the performance of the elliptic curve cryptography in a programming environment, using the PHP 7.2 version and the OpenSSL library. The findings indicate that the optimal elliptic curve to use in the research environment constructed in this way is the prime256v1 curve with a 1.11% efficiency factor, which corresponds to a security level of RSA 3072 bytes.

In the 160 – 223 bytes group, the most efficient (with the same ratio) elliptic curves are prime192v1, prime192v2 and prime192v3, in the group of 224 – 255 bytes – prime239v1, prime239v2 and prime239v3, in the group of 256 – 383 bytes – prime256v1, in group 384 – 511 bytes - secp384r1, and in the 512+ bytes group – brainpoolP512l1.

The second aim was to analyse the performance of the Argon2i hashing algorithm, which is part of the Sodium cryptographic library and utilised both from the system console and the PHP 7.2 version. The time required to perform the hashing operations in the console version are similar to those in the PHP version, with the increase in time and depending on the memory acquired for use by the Argon2i algorithm.

It is also important to note that the algorithm efficiency is largely influenced by the number of threads used. With 1 GB of memory used, six iterations and one thread, the time taken to perform this operation was almost 12 seconds. With four threads, it dropped by almost half and with eight threads it reached just over 4.5 seconds. Of course, the above mentioned example is only a limiting example of the Argon2i algorithm in the production environment, as it is charged with a heavy use of processor power and the memory. The use of 16MB or 32MB memory at three iterations and two threads is the optimal solution in terms of time and the use of hardware resources.

2.3. The outcome of the experiment

The results of the tests covering the Argon2d and Argon2i algorithms, performed from the systems (console) level, are presented in the Table 9.

The results of the tests covering the Argon2i algorithm, performed from the PHP language level, for one, three and six iterations are presented in the Table 10.

| Thread | Memory in MB | Argon2i (in seconds) |
|--------|--------------|-----------------------|
| 1      | 16           | 0.0404                |
| 2      | 64           | 0.1690                |
| 3      | 512          | 1.4336                |
| 4      | 1024         | 2.9056                |
| 5      | 16           | 0.0826                |
| 6      | 64           | 0.3781                |
| 7      | 512          | 3.1983                |
| 8      | 1024         | 6.4867                |
| 9      | 16           | 0.0692                |
| 10     | 64           | 0.4265                |
| 11     | 512          | 5.8553                |
| 12     | 1024         | 11.8148               |
| 13     | 16           | 0.1349                |
| 14     | 64           | 0.2332                |
| 15     | 512          | 6.8337                |
| 16     | 1024         | 0.0918                |
| 17     | 16           | 0.1636                |
| 18     | 64           | 0.3166                |
| 19     | 512          | 6.6304                |
| 20     | 1024         | 0.2475                |
| 21     | 16           | 0.0807                |
| 22     | 64           | 0.2855                |
| 23     | 512          | 8.5802                |
| 24     | 1024         | 0.3177                |
| 25     | 16           | 0.1344                |
| 26     | 64           | 0.3055                |
| 27     | 512          | 0.5987                |
| 28     | 1024         | 0.5112                |

References

[1] Blake I., Seroussi G., Smart N.: Krzywe eliptyczne w kryptografii. WNT, Warszawa 2004.
[2] Kobliż N.: Algebraiczne aspekty kryptografii. WNT, Warszawa 2000.
[3] Kobliż N.: Wykład z teorii liczb i kryptografii. WNT, Warszawa 2006.
[4] https://w3tech.com/technologies/details/pl-php/all/all (available: 14.11.2019).
[5] https://www.toiobe.com/toiobe-index (available: 14.11.2019).
[6] https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2018 (available: 14.11.2019).
[7] https://password-hashing.net (available: 25.04.2019).

M.Sc. Eng. Marcin Duka
e-mail: mariduk781@student.polsl.pl

PhD student at the Faculty of Automatic Control, Electronic and Computer Science of the Silesian University of Technology. Professionally associated with the IT industry since 1993, in particular in software design and its implementation. Author of many Web projects, including the ISOWQ (International Studies of Website Quality) ranking system. His scientific interests related to the issues of data exploring, concurrent programming and IoT.

http://orcid.org/0000-0001-5773-0459

otrzymano/received: 16.12.2019 przyjęto do druku/accepted: 15.09.2020