COMPARISON OF THE EFFICIENCY OF AES IMPLEMENTATIONS ON MAJOR WEB PLATFORMS

Uroš Arnaut*,
Milan Tair,
Mladen Veinović

Faculty of Informatics and Computing,
Singidunum University,
Belgrade, Serbia

Abstract:
In this paper, the authors present an experimental study covering the evaluation of the average encryption and decryption times using different programming languages for the Web. This study covers the use of the most commonly used AES implementations for four major Web programming and scripting languages: Java, Node.js, PHP and Python. The aim of the study is to determine the cost of encrypting and decrypting data on these platforms, expressed as type per byte of data. The experiment covers data encryption and decryption with the AES algorithm in the CBC mode with 128-bit, 192-bit and 256-bit keys. In this paper, we present the results and pros and cons of use of the AES algorithm implementations on these major Web platforms.

Keywords:
AES, Web platforms, Experimental Study.

INTRODUCTION

Data encryption has become essential for companies to prevent classified information from leaking out. It means that all employee personal information, usernames, passwords, and contacts stored in databases should not be exposed. For example, the more prominent company is, the more critical data set has to be stored. That means that companies should not write plain text (open text form) in a database, especially passwords, because if the information system gets exposed, there is a possibility that the database is exposed as well. The third side can access classified information, and they can harm employees and the company. That is why encryption plays a big part in every information system. Some companies are hiring other companies from the field of Cybersecurity to manage their classified data. Others are forming their teams. However, this will not be a topic in this paper.

Today we have many different techniques for data encryption. In this paper we will encrypt data that will simulate the most common file sizes (word and excel, for example). The algorithm that will be used is AES (Advanced Encryption Standard). Data will be encrypted and decrypted with mostly used programming languages like Java, PHP, Python, JavaScript, etc.

Correspondence:
Uroš Arnaut

e-mail:
uarnaut@singidunum.ac.rs

DOI: 10.15308/Sinteza-2021-153-157

Average encryption and decryption time will be calculated separately for every programming language in attention to compare performances.

Advanced Encryption Standard (AES) is a symmetric cryptographic algorithm using three different key sizes 128, 196, and 256 bits. When it was first introduced, the algorithm had more than three key sizes, but they were not accepted as standard. All data for encryption in the AES algorithm is processed in bytes, which means that all initial data, including the encryption key, is calculated in bytes.

The initial block size is 128 bits and all mathematical operations are performed with two-dimensional byte arrays. The number of rounds of execution of the AES algorithm depends on the length of the key.

The AES algorithm takes two inputs:

- The plain text, which needs to be encrypted;
- The key.

The key is usually accompanied by the initialisation vector, commonly abbreviated as IV in implementations of the AES algorithm in most programming languages.

The algorithm starts with the "Key Expansion," in which round keys are derived from the original key. After the expansion, an initial round occurs, in which the original key is applied to the plaintext. Depending on the key length, the algorithm will go through 9, 11, or 13 rounds. Each round performs four operations:

- Bytes substitution;
- Row shift;
- Mix columns; and
- Add Round Key.

Once these rounds are completed, one last round is performed, which has no "Mix columns" step.

After the encryption, AES outputs the cipher-text.

2. WEB PROGRAMMING LANGUAGES

Companies need to secure data now and then, and the best way is to use a secure algorithm with the most efficient programming language. Nowadays, people use encrypted data in everyday life. Social network sites, electronic newspapers, media web-sites, etc are using different algorithms, approaches, and methodologies to secure used data.

There are many programming languages that can be used on the Web. However, this research will focus on the ones most commonly used for Web application development [1, 2, 3, 4, 5].

2.1. JAVA

The Java programming language is a general-purpose, concurrent, class-based, object-oriented language. It is designed to be simple enough that many programmers can achieve fluency in the language. The Java programming language is related to C and C++ but is organized somewhat differently, with several aspects of C and C++ omitted, and a few ideas from other languages included. It is intended to be a production language, not a research language [2].

In our research, we use the Crypto Java library [7].

2.2. PHP

PHP *(recursive acronym for PHP: Hypertext Preprocessor)* is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML. What distinguishes PHP from something like client-side JavaScript is that the code is executed on the server, generating HTML sent to the client. The client would receive the results of running that script, but would not know the underlying code. You can even configure your web server to process all your HTML files with PHP, and then there's no way that users can tell what you have up your sleeve [8]. We used the latest PHP version 8.

In our research, we use the OPENSSL Cipher PHP extension [9], which supports the AES algorithm in CBC mode with key sizes 128, 192 and 256 bits [10].

2.3. JAVASCRIPT (USING THE NODE.JS PLATFORM)

Node.js is a server-side platform useful for building highly scalable and fast applications. Node.js is a platform built on v8, the JavaScript runtime that powers the Chrome browser designed by Google. Node.js is designed to be great for intensive I/O applications utilizing the non-blocking event-driven architecture.

While Node.js can synchronously serve functions, it most commonly performs operations asynchronously. That means that when an application is developing, events with a call-back registered for handling the return of the function is called. While awaiting the return, our application's next event or function can be queued for execution. Once the first function completes, its call-back event is executed and handled by the function call that invoked the call-back. Node.js is a platform built on Chrome's JavaScript runtime for quickly building fast,

scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it light-weight and efficient, perfect for data-intensive real-time applications that run across distributed devices [4].

In our research, we use the Crypto library [12].

2.4. PYTHON

Python is an interpreter, interactive, object-oriented language. It incorporates modules, exceptions, dynamic typing, very high-level dynamic data types, and classes. It supports multiple programming paradigms beyond object-oriented programming, such as procedural and functional programming. Python combines remarkable power with very clear syntax. It has interfaces to many system calls and libraries and various window systems and is extensible in C or C++. It is also usable as an extension language for applications that need a programmable interface [5].

In our research, we use PyCryptodome [14].

3. RESEARCH METHODS

For this research, we have used an experiment method to generate research data. A simple encryption and decryption program was written that automates the generation of the research data. The generated data was analyzed using quantitative analysis methods and the analysis outputs were presented in the results and discussion section of this paper.

The experiment covered in this paper aims to help identify the most efficient Web programming language and its most commonly used implementation of functions, methods of libraries for the use of the AES algorithm for encrypting data. The AES keys used in the experiment were 128-bit, 192-bit, and 256-bits in size. The AES was configured in the CRC mode.

To ensure equal conditions, the data for encryption was generated ahead of the experiment and stored in files and read before use for encryption. When measuring the time for encryption and decryption, only the time it takes to perform the encryption and decryption is measured in the highest precision possible in that platform.

Most platforms support microsecond precision, while Java supports nanosecond precision. To ensure that the results are of the same resolution, the microsecond precision was used for all platforms. There were a total of 30 randomly generated files, whose sizes ranged from 145 bytes to 8MB.

To ensure high precision, the process of encrypting and decrypting this data was repeated 100 times for each combination of data size and AES key size.

After the completion of this process, average times for both actions have been calculated.

The experiment does not cover the following topics:

- Generating cryptographic keys;
- Sharing cryptographic keys;
- Saving cryptographic keys.

The experiment was performed on the same hardware and software. The computer used to run this automated process used a 64bit Windows operating system, a total of 4GB of RAM and an Intel i5 3300 @ 3.0GHz 3.1GHz processor.

During the execution of the research data generation process, all non-essential processes were turned off and the system was disconnected from the Network. All high-load software was turned off for the duration of the experiment.

Table 1 shows the list of programming languages covered by this research, as well as theirs versions.

```
keys := Map(
  128 => read contents from 128.key,
  192 => read contents from 192.key,
  256 => read contents from 256.key
)
FOR len IN List(128, 192, 256) DO
  FOR file IN List( *.data files ) DO
    data := read contents from file
    key  := keys(len)
    encTime := 0
    decTime := 0
    FOR i := 1 TO 100 DO
      start := time_µs()
      encrypted = AES_enc(size, len, data)
      encTime := encTime + time_µs() - start
      start := time_µs()
      AES_dec(size, len, encrypted)
      decTime := decTime + time_µs() - start
    END
    store_results(file, len, encTime, decTime)
  END
END
```

Listing 1 - The data generation algorithm

4. RESULTS AND DISCUSSION

After running the previously mentioned programs that generate the research data containing the information about the size of the encrypted data, determined from the file, the key size, identified by the length of the key, in bytes as well as total encryption and decryption times, we can create a set of raw data records for further analysis.

Considering that the collected times of completion of data encryption and decryption are based on a total of 100 iterations, we have calculated the average encryption and decryption time per iteration. The increased number of iterations has helped minimize noise which may have occurred due to unexpected events on the operating system.

To further enable direct comparison of these results, these values were divided by the total size of the encrypted data to generate information about the total time per byte, for each combination of key size and programming language and its platform.

Information about the average amount of time in nanoseconds needed to complete encryption and decryption of a single byte of data, grouped by the platform, operation and key size, gained from the analysis of the acquired data is shown in Figure 1.

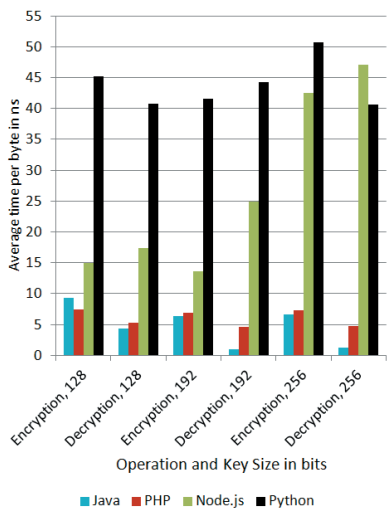

Figure 1 - Experimental results

This figure helps visually conclude that in terms of performance, in most cases Java has shown that it is the best choice for data encryption in terms of average time needed to complete decryption of data, while it was only slower than PHP when performing encryption using a 128-bit key. PHP was second in terms of speed. PHP was configured in such a way as not to use pre-loading and PHP opcache for caching compiled code between iterations. The Node.js platform was third in the overall rating. However, it did end up slower than Python when performing decryption using a 256-bit key. Python, on the other hand, was by far the slowest of all four platforms. Python also did not utilize any code caching and was interpreted without using the pre-compilation option.

Considering these results of average time (in ns) to complete encryption and decryption, it is clear that at the moment, AES implementations in Java and PHP are, performance-wise, the most efficient and fastest.

Table 2 contains average times (in milliseconds) needed for the completion of encryption and decryption operations on each platform for contents ranging from 145 bytes to 8MB in size.

Data size	Java		PHP		Node.js		Python	
	Encryption	Decryption	Encryption	Decryption	Encryption	Decryption	Encryption	Decryption
145B	0,0013	0,0005	0,0018	0,0017	0,0070	0,0097	0,0400	0,0400
232B	0,0017	0,0005	0,0020	0,0018	0,0070	0,0144	0,0300	0,0600
333B	0,0022	0,0005	0,0426	0,0335	0,0073	0,0118	0,0400	0,0500
459B	0,0027	0,0005	0,0024	0,0026	0,0091	0,0320	0,0400	0,0600
579B	0,0035	0,0006	0,0026	0,0027	0,0091	0,0137	0,0500	0,0400
591B	0,0035	0,0006	0,0025	0,0018	0,0079	0,0116	0,0300	0,0600
659B	0,0041	0,0006	0,0027	0,0019	0,0102	0,0122	0,0300	0,0600
926B	0,0046	0,0007	0,0114	0,0082	0,0099	0,0163	0,0200	0,1000
1KB	0,0056	0,0009	0,0034	0,0022	0,0189	0,0267	0,0200	0,0800
2KB	0,0071	0,0010	0,0035	0,0021	0,0114	0,0177	0,0200	0,0800
28KB	0,1773	0,0198	0,0519	0,0154	0,0809	0,1683	0,3200	0,3299
56KB	0,3295	0,0369	0,1028	0,0300	0,1443	0,2662	0,6599	0,5999
84KB	0,4882	0,0550	0,1547	0,0453	0,2527	0,3828	0,9698	0,8999
112KB	0,6819	0,0742	0,2053	0,0573	0,2439	0,5109	1,2199	1,2098
143KB	0,8419	0,0945	0,2723	0,0748	0,3543	0,6393	1,5498	1,5098
223KB	1,3651	0,1516	0,4106	0,1149	0,6092	1,1449	2,3996	2,3397
285KB	1,7549	0,1898	0,5273	0,1460	0,7792	1,7376	3,2994	2,9898
334KB	2,1195	0,2474	0,6196	0,1695	1,0055	1,9356	3,8194	3,5796
445KB	2,8507	0,3381	0,8234	0,2226	1,2196	2,4182	5,4692	4,7795
557KB	3,6301	0,4341	1,0220	0,2803	1,4020	5,6863	6,9892	6,3091
570KB	3,9280	0,4403	1,0383	0,2833	1,4466	1,7638	6,6889	6,1593
668KB	4,4360	0,4758	1,2305	0,3390	1,7143	2,1266	8,1588	7,1591
779KB	5,6277	0,6444	1,4393	0,3943	2,2629	2,5276	9,8085	8,8790
891KB	5,7979	0,6153	1,6400	0,4470	2,0848	2,5663	11,1185	10,0487
1.1MB	8,4281	0,8539	2,6979	1,1123	2,5725	3,0785	14,2283	13,0181
1MB	6,5069	0,6842	1,9483	0,5476	2,5700	3,1188	12,2284	10,8585
2MB	16,8804	1,8979	5,7596	2,4347	6,3440	6,5832	30,5859	28,0962
1.5MB	9,7235	1,2348	3,7038	1,5790	3,2954	4,3860	19,0072	17,4979
4MB	28,4921	3,7201	9,9686	4,2270	10,5812	13,5297	52,2828	48,2036
8MB	56,8131	17,6285	19,7981	8,5810	20,2340	29,9842	98,9467	91,9176

Table 2 - Average times (in ms) by data size

Based on the experimental data, we can conclude that on almost all platforms, AES decryption in CBC mode, regardless of the key size is faster than encryption. Also, we can conclude that the compiled language Java outperforms other platforms. Of all interpreted languages, Python was least efficient in performing AES encryption and decryption, on average.

5. CONCLUSION

This study covered the most commonly used AES implementations for four major Web programming and scripting languages: Java, Node.js, PHP, and Python. The study aimed to determine the cost of encrypting and decrypting data on these platforms. The experiment has covered data encryption and decryption with the AES algorithm in the CBC mode with 128-bit, 192-bit, and 256-bit keys.

Results of the study show that decryption mode, almost all platforms, regardless of the key size is faster than encryption mode. Experiment shows as well that Java programming language has outperformed other used web-based platforms when it comes up to the speed of data encryption and decryption. We should consider that the version of Java which was used was at let least two years outdated compared to the release dates of other programming languages and platforms, this result. PHP programming language had similar results as Java while a 128-bit key was used, even better when it comes up to encryption. Both platforms performed those processes under 10 ns on the used hardware and software configuration. The performance of Node.js and Python AES implementations was not up to the authors expectations. While using a 128-bit key, Node.js was slower than Java and PHP when performing both the encryption and decryption operations. The Python implementation was the slowest.

We can conclude that Java is a platform which has the most efficient AES implementation. When choosing between interpreted languages, PHP is the second best option for this particular job. Security issues of the mentioned algorithms, as well as their implementation with other libraries were not addressed in this paper.

In our future research we will endeavor to reduce the impact of hardware and software configuration on the performance of these on the AES implementations on these and other platforms. We will aim to expand the list of tested programming languages and platforms and to test different implementations, using other, less-popular libraries and packages for each language. Additionally, we will endeavor to expand the list of tested languages to other platforms, and not only those used for Web application development.

REFERENCES

[1] B. Eastwood, "The 10 most popular programming languaes to learn in 2020," University Northeastern, 18 June 2020. [Online]. Available: https://northeastern.edu/graduate/blog/most-popular-programming-languages/. [Accessed 25 January 2020].

[2] IEEE Spectrum, “Interactive: The Top Programming Languages,” 2020. [Online]. Available: https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2020. [Accessed 7 May 2021].

[3] Statista, “Most used programming languages among developers worldwide, as of early 2020,” Shanhong, February 2020. [Online]. Available: https://statista.com/statistics/793628/worldwide-developer-survey-most-used-languages/. [Accessed 7 May 2021].

[4] Wappalyze, "Programming languages technologies market share," 2020. [Online]. Available: https://wappalyzer.com/technologies/programming-languages/. [Accessed 7 May 2021].

[5] Datanyze, "Programming Languages Market Share," 2020. [Online]. Available: https://datanyze.com/market-share/programming-languages--67. [Accessed 7 May 2021].

[6] Oracle, "Chapter 1. Introduction," Oracle, [Online]. Available: https://docs.oracle.com/javase/specs/jls/se15/html/jls-1.html. [Accessed 25 January 2020].

[7] Oracle, “Java Cryptography Architecture (JCA) Reference Guide,” [Online]. Available: https://docs.oracle.com/javase/8/docs/technotes/guides/security/crypto/CryptoSpec.html. [Accessed 3 April 2021].

[8] PHP Documentation Group, "What is PHP?". [Online]. Available: https://www.php.net/manual/en/intro-whatis.php. [Accessed 25 January 2020].

[9] PHP, "OpenSSL," [Online]. Available: https://php.net/manual/en/book.openssl.php. [Accessed 5 April 2021].

[10] PHP, "Function openssl_get_cipher_methods," [Online]. Available: https://php.net/manual/en/function.openssl-get-cipher-methods.php. [Accessed 5 April 2021].

[11] C. Gackenheimer, Node.js Recipes - A Problem-Solution Approach, New York City: Apress, 2013.

[12] Node.js, “Crypto | Node.js Documentation,” [Online]. Available: https://nodejs.org/api/crypto.html#crypto_crypto. [Accessed 4 April 2021].

[13] Python Software Foundation., "General Python FAQ," Python Software Foundation., [Online]. Available: https://docs.python.org/3/faq/general.html#what-is-python. [Accessed 25 January 2020].

[14] H. Eijs, "PyCryptodome - Cryptographic library for Python," 9 February 2021. [Online]. Available: https://pypi.org/project/pycryptodome/. [Accessed 4 April 2021].