Benchmarking Performance of

Hashing (*SHA*), Symmetric (*AES – GCM*) and

Asymmetric (*ECC*) Encryption Algorithms

And Their Required Hardware BOM Cost Difference for Smart Meters

By: Pouya Ebadollahyvahed

1- What Is the Problem? In Which Situations Do I Need to Make a Decision?

When there is a device in the field (like a smart meter or any other IoT device) and a master station (like MDC / AHE / Head-End) that they are communicating, the communication should be secured. Customers (usually utilities) often ask for one of following methods as also specified in DLMS / COSEM:

- Use a common key (at both sides) and then use a symmetric encryption method like AES
- Use a pair of public / private keys and then use an asymmetric encryption method like ECC

Table 27 – DLMS/COSEM security suites

Security Suite Id	Security suite name	Authenticated encryption	Digital signature	Key agreement	Hash	Key-transport	Compression
0	AES-GCM-128	AES-GCM-128	–	–	–	AES-128 key wrap	–
1	ECDH-ECDSA-AES-GCM-128-SHA-256	AES-GCM-128	ECDSA with P-256	ECDH with P-256	SHA-256	AES-128 key wrap	V.44
2	ECDH-ECDSA-AES-GCM-256-SHA-384	AES-GCM-256	ECDSA with P-384	ECDH with P-384	SHA-384	AES-256 key wrap	V.44
All other reserved	–	–	–	–	–	–	–

DLMS User Association	2021-12-21	DLMS UA 1000-2 Ed. 11	227/614

Figure 1Security Suites of DLMS / COSEM Communication Protocol, Green Book

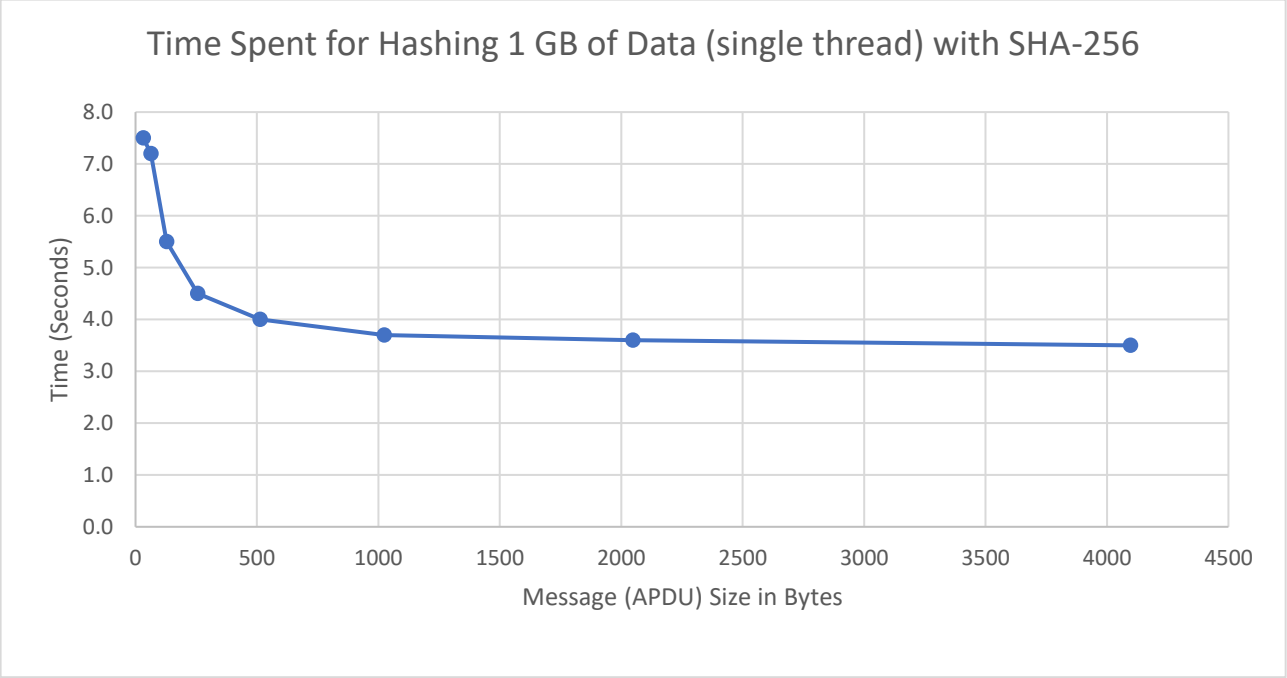
Since almost all of smart meters are certified for DLMS Suite 0, natively they support the first method (Symmetric Encryption) while to support the second method (Asymmetric Encryption) smart meters should be certified for DLMS Suite 1. **In this article we want to answer following questions**:

1- How much it will impact the hardware BOM cost to migrate from DLMS security suite 0 to suite 1?

2- Is it really necessary and does it really improve the security if we use asymmetric encryption (or digital signature) instead of symmetric one?

2- Performance of SHA-256 with Different Message Sizes

This test is about performance of hashing operation based on SHA-256 algorithm (DLMS Security Suite 1). In this test 1GB of data is virtually transferred several times and each time with different message (= APDU) sizes (each time all APDUs had the same length). The total time of hashing of this 1GB is measured. Following diagram shows the total time spent to run SHA-256 on 1GB of date with different APDU sizes.



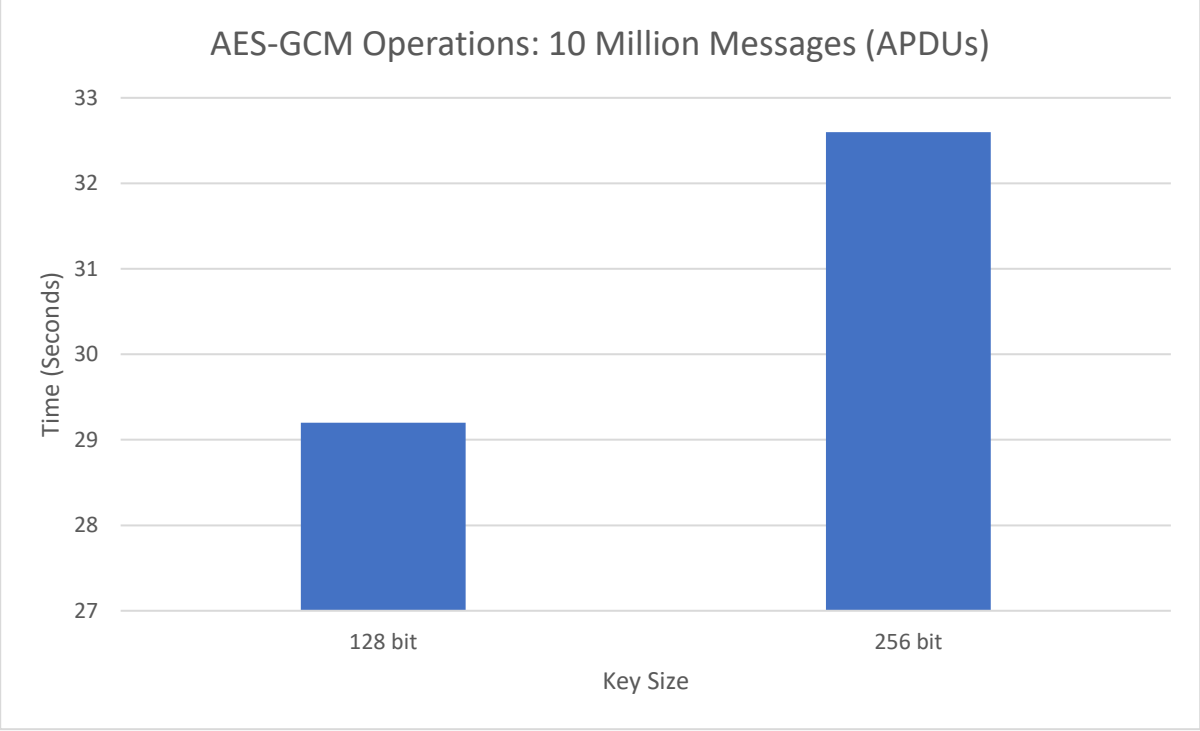
3- Performance of AES with 128-bit and 256-bit Key Sizes

The second test which is more important is encrypting 10 million messages (APDUs) with symmetric encryption method which is AES – GCM but with different key sizes. Two key sizes are tested in this test: 128-bit (based on DLMS security suite 0 and 1) and 256-bit (based on DLMS security suite 2).

All messages (APDUs) have fixed 128-byte length. For all tests, operations consist of:

- Encryption: 50%
- Decryption – successful: 25%
- Authentication (with planned failed result): 25%

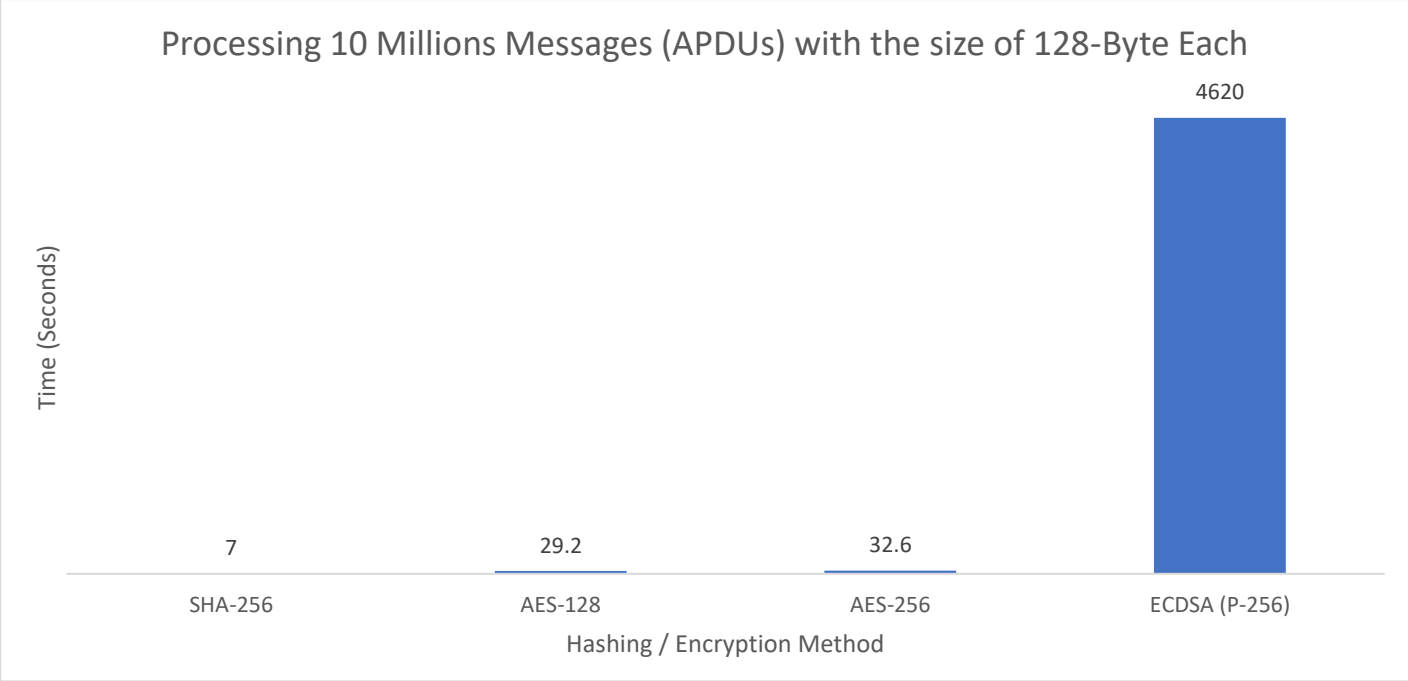
Test data is standard test data and downloaded from the web site of NIST.



4- Performance of SHA vs AES vs ECDSA

The last test is benchmarking performance of hashing (SHA-256), symmetric encryption (AES-GCM 128 and 256) and digital signature (ECDSA Curve **secp256r1** which is also called NIST P-256).

In each test 10 million messages (APDUs) with fixed length of 128-byte are processed.



5- Security of Symmetric and Asymmetric Encryption Algorithms

While a number of people think asymmetric encryption methods are more secure, the reality is opposite. For example, security of ECC with N-bit key is equivalent to a symmetric algorithm with the key size of N/2 bits (so very roughly we can say security of ECC-256 is equivalent to AES-128). The reason is that in symmetric encryption a third party (like a cracker) doesn't know anything about the encryption key while in asymmetric encryption public key is published and known. Since public and private keys are mathematically correlated so we can claim any 3rd party knows something about private key too.

6- Screen-shots of the **Benchmarking Software** and Tests

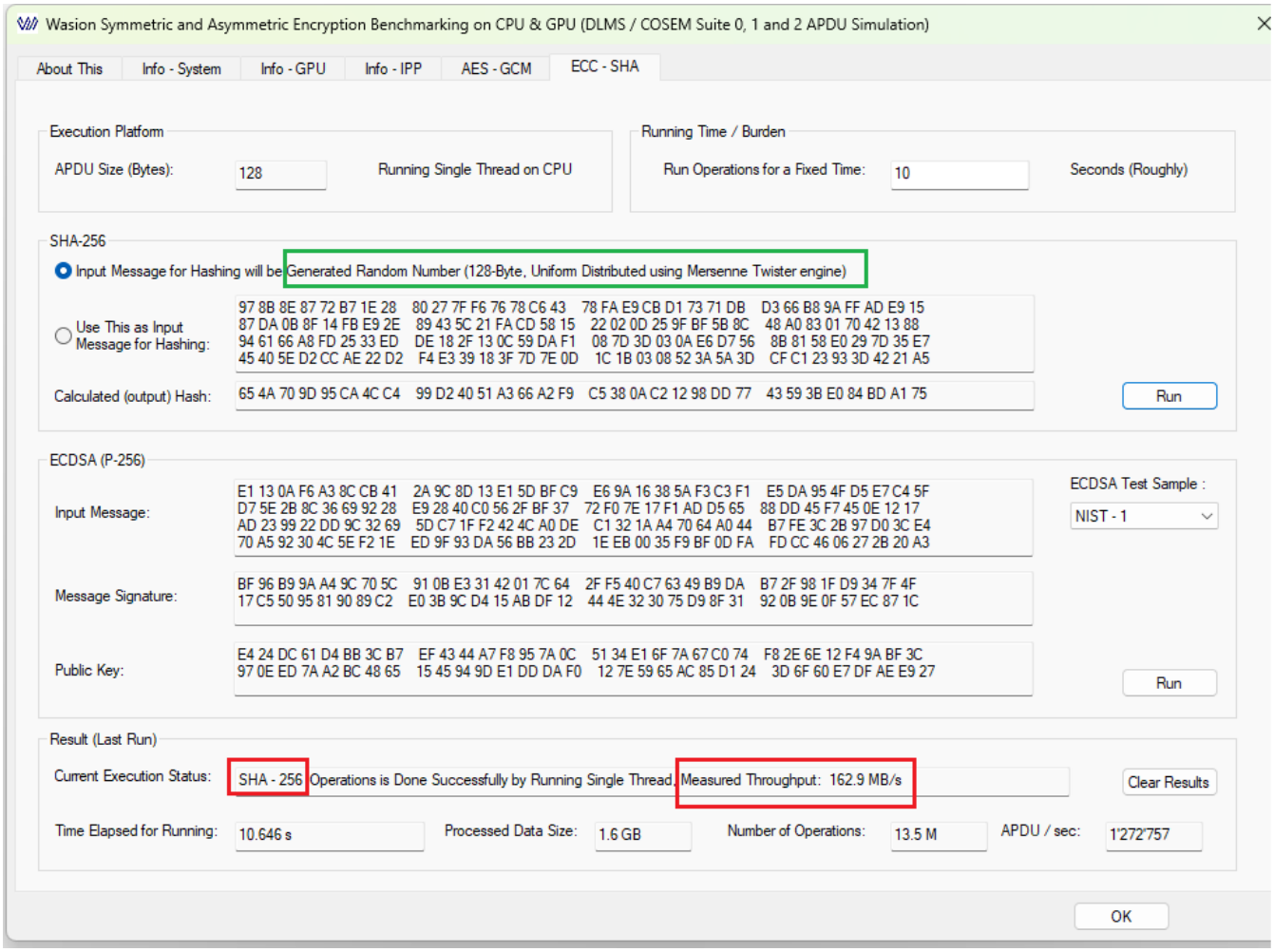


Figure 2SHA-256 Test Screen

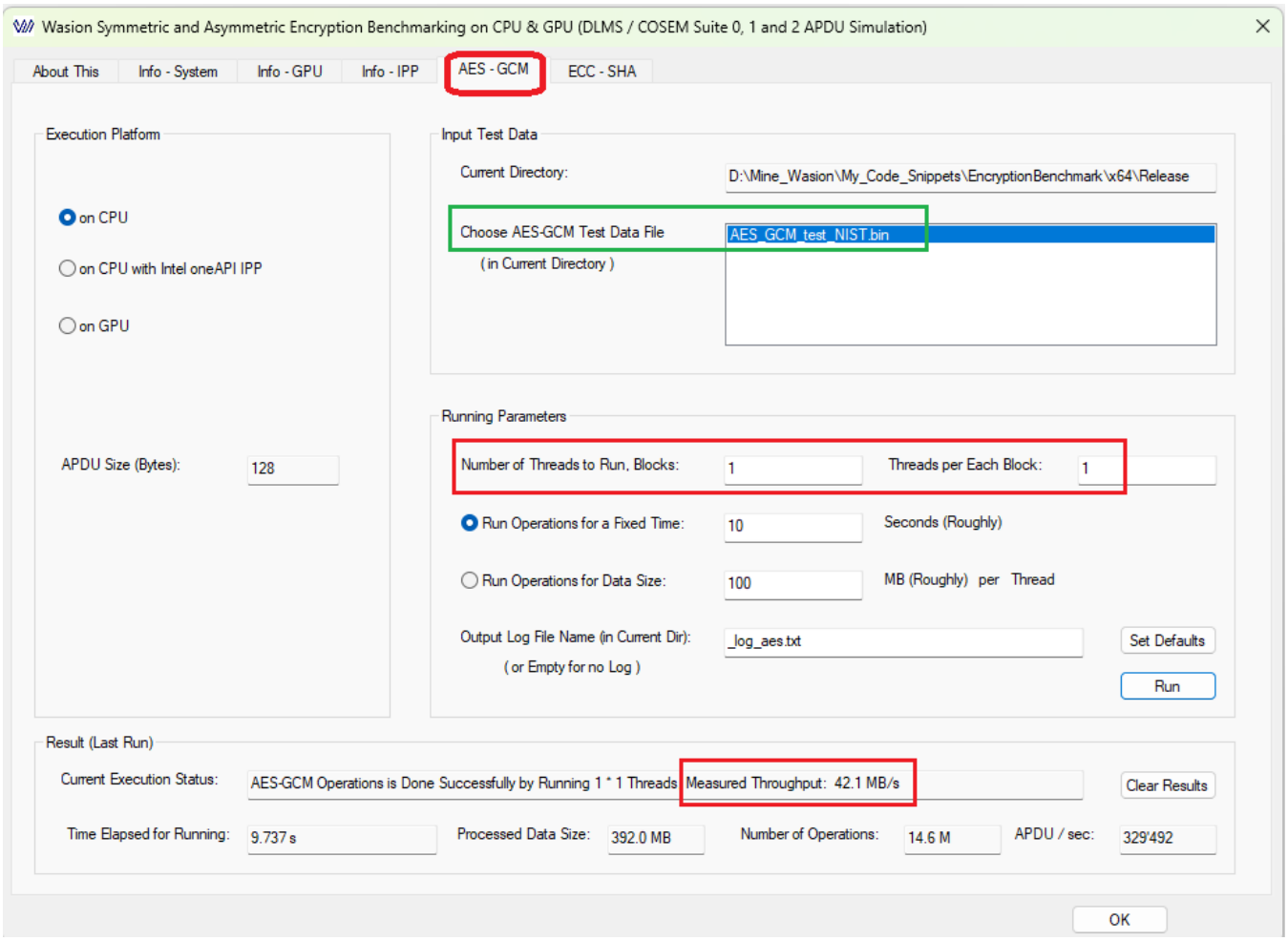


Figure 3AES-GCM Test Screen (Running on CPU with 1 Thread only)

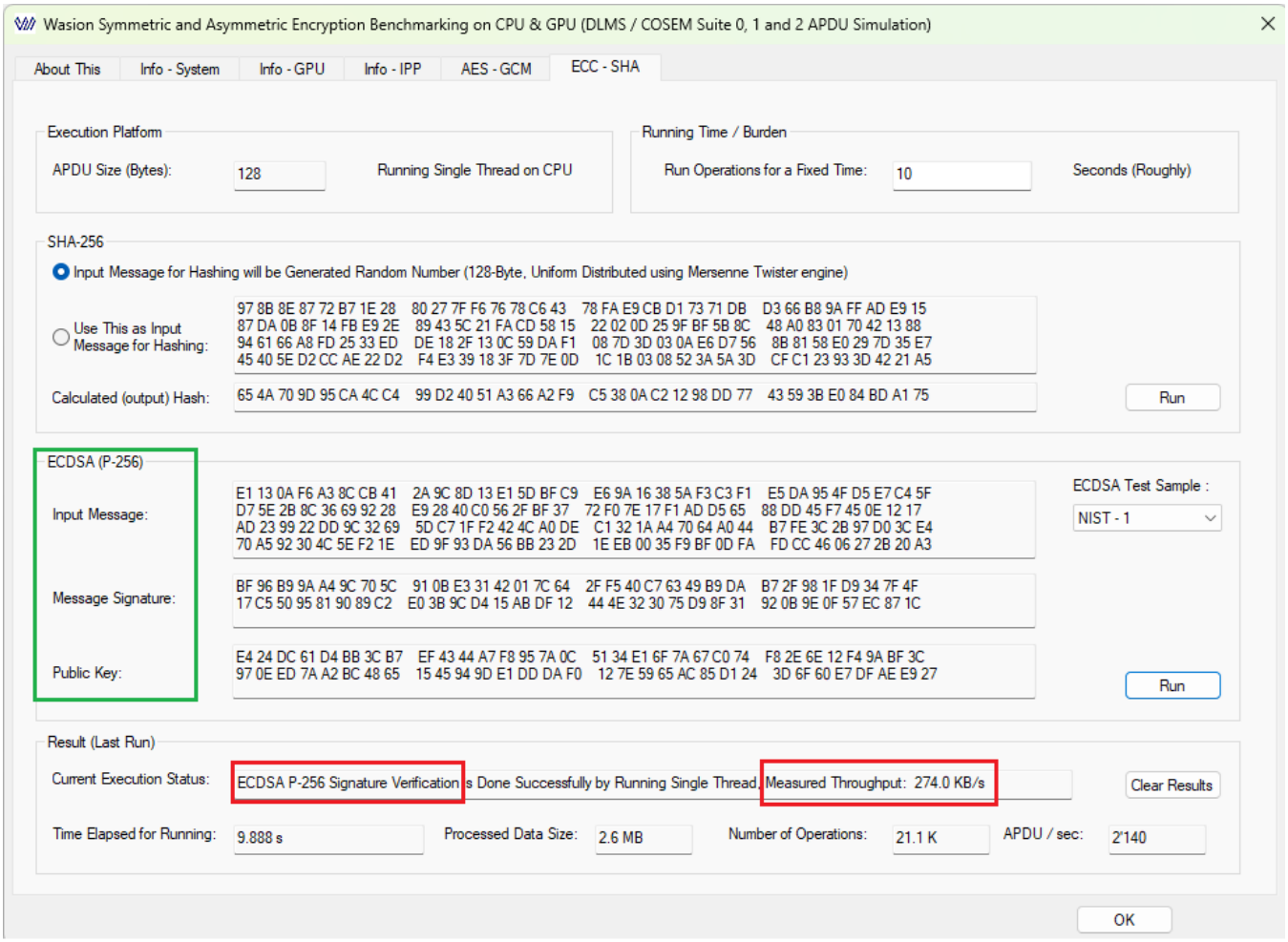


Figure 4ECDSA Test Screen

```
Wasion America, Encyption Benchmarking App, Log File

Algorithm               : AES-GCM
Operations              : Encryption, Decryption and Authentication
Input Vector File Size  : 6'558'654 B

Executed Platform       : CPU, 12th Gen Intel(R) Core(TM) i7-12700H
Logging Time            : Wed Oct  9 15:31:16 2024



 Total Number of Encryption Operations                                  : 148.1 M
 Total Number of Decryption Operations with Successful Authentication   : 73.6 M
 Total Number of Decryption Operations with Planned Failed Authentication: 74.5 M
 Total Number of            Operations with  128-bit  key               : 98.7 M
 Total Number of            Operations with  192-bit  key               : 98.7 M
 Total Number of            Operations with  256-bit  key               : 98.7 M
 Total Volume of Messages Processed                                     : 7.9 GB



Operations Details:

Operation Encryption; Sequence No # 01 H
        Plain Text                 :    D5 DE 42 B4 61 64 6C 25    5C 87 BD 29 62 D3 B9 A2
        Key                        :    7F DD B5 74 53 C2 41 D0    3E FB ED 3A C4 4E 37 1C
        Initial Vector             :    EE 28 3A 3F C7 55 75 E3    3E FD 48 87
        Encrypted Text (Reference) :    2C CD A4 A5 41 5C B9 1E    13 5C 2A 0F 78 C9 B2 FD
        Encrypted Text (Calculated):    2C CD A4 A5 41 5C B9 1E    13 5C 2A 0F 78 C9 B2 FD

Operation Encryption; Sequence No # 02 H
        Plain Text                 :    00 7C 5E 5B 3E 59 DF 24    A7 C3 55 58 4F C1 51 8D
        Key                        :    AB 72 C7 7B 97 CB 5F E9    A3 82 D9 FE 81 FF DB ED
        Initial Vector             :    54 CC 7D C2 C3 7E C0 06    BC C6 D1 DA
        Encrypted Text (Reference) :    0E 1B DE 20 6A 07 A9 C2    C1 B6 53 00 F8 C6 49 97
        Encrypted Text (Calculated):    0E 1B DE 20 6A 07 A9 C2    C1 B6 53 00 F8 C6 49 97

Operation Encryption; Sequence No # 03 H
        Plain Text                 :    69 31 A3 EA 07 A9 E9 52    07 33 4F 02 74 A4 54 DD
        Key                        :    77 B0 A5 8A 1E 60 54 1E    5E A3 D4 D4 20 07 94 0E
        Initial Vector             :    AE 7A 27 90 4D 95 FE 80    0E 83 B3 45
```

Figure 5 Log File Generated by AES-GCM Test, Consists Details of all Encrypted / Decrypted Messages

7- Hardware BOM Cost

What is the cost difference between an IoT device (like an Energy smart meter) which only supports DLMS security suite 0 and the 2nd device which also complies with DLMS security suite 1?

The 1st device can run only AES-GCM 128 bit which is relatively light-weight algorithm (based on the results of tests in topic 3 and 4 of this article) while the 2nd device should also support running heavy-weight algorithms like ECDSA which is almost 160 times slower[1].

Based on my personal experiences for the 1st device an "ARM Cortex M0+" MCU is enough while for the 2nd device we need a high end MCU like "ARM Cortex M4" based one. While the retail price for the 1st MCU in **Digikey** is just 1 USD, the 2nd one (typical) in the same store costs 7 USD. Since the 1st MCU needs much less & cheaper driving peripherals (decoupling capacitors, crystals and ….) and occupies smaller area on PCB, the total BOM cost difference between these two devices is even more.

The other difference between hardware design of these two devices is their energy consumption; obviously the 1st device with simpler MCU and less peripherals consume less energy and is more battery friendly.

The other key difference in design of above-mentioned devices is that for the first device a single core MCU suffices while if the 2nd device is supposed to run some real-time or semi real-time tasks then its MPU / CPU should be multi-core. The reason is that asymmetric algorithms as we tested and published its results in topic 4 of this article are very slow and keep the MPU / CPU core busy for long time and this delay cannot be tolerated in real-time systems so they should be executed on separate core of MPU / CPU.

[1] Please refer to topic 4 of this article (benchmarking test result)

8- Test platform (Hardware & Software Specification)

Machine	Laptop Asus TUF Gaming F15
CPU	Intel Core i7 12700 @ 4 GHz (**single core execution**)
RAM	32 GB @ 3.2 GHZ[2]
OS	Windows 11 Enterprise 64-bit (23H2)
Language	ISO C++17[3] MFC for GUI
Compiler	Visual Studio 2022 for Windows V17.10 (MSVC)
Compile Mode	64-bit Release Mode (optimized for speed)

[2] All data (raw / plain data, encrypted data, keys and any other) are kept in, loaded from and stored in RAM so there were no disk operations during tests.
[3] Only ECDSA was coded in ANSI C

9- Conclusion

I. Impact of message (APDU) size on performance

   a. Result of the test discussed in topic 2 of this article shows that sending data in longer chunks (= bigger messages / APDUs) increases encryption/decryption/hashing performance (more than 100%). Sending data in longer messages also removes communication latency and increases communication performance. The only problem with bigger (longer) messages is that if the receiver, receives a message with invalid checksum / CRC / hash which is usually caused by telecommunication channel noise, the whole message should be resent again and, in this case, bigger messages cause more overhead. In DLMS / COSEM protocol APDUs, with 128 to 512 Byte sizes prove better overall performance.

II. Performance difference between AES-GCM-128 and AES-GCM-256 is measured almost 10%.

III. Performance of AES-GCM operation is measured 42 MB per second per CPU core in our tests which doesn't appear as challenging neither for an IoT device MPU nor for a master station machine (MDC / AHE / Head-end system)

IV. Asymmetric encryption algorithms like ECC and ECDSA are way slower than symmetric ones. In our tests it is measured as almost 160 times. Their implementations are also harder and takes longer time and ends up with longer code (personally I can say by factor of 10). To handle complicated mathematical calculations required by asymmetric algorithms, a richer and more expensive processors (and hardware) are also needed.

   a. It is advised that for IoT communications, symmetric keys are being encrypted by an asymmetric method, being transferred from one side to another side and then for the rest of the communication only symmetric encryption methods like AES are being used.

10- Table of Acronyms and Abbreviations

Acronym / Abbreviation	Stands For
AES	Advanced Encryption Standard
AES-GCM	Advanced Encryption Standard - Galois Counter Mode
AHE	Advanced Head-End
APDU	Application Protocol Data Unit
B	Byte
b	bit
COSEM	Companion Specification for Energy Metering
CPU	Central Processing Unit
DLMS	Device Language Message Specification
DLMS UA	Device Language Message Specification User Association
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
GB	Giga Byte (Giga means almost 10 to the power of 9)
H	Hexadecimal
IoT	Internet of Things
KB/s	Kilo Bytes per Second (Kilo means 1024)
M	Mega (almost one million)
MB/s	Mega Bytes per Second (Mega means almost one million)
MDC	Meter Data Collector / Collection
MSVC	Microsoft Visual C
NIST	National Institute for Standards and Technology
Oct	October
OS	Operating System
R	Registered
SHA	Secure Hash Algorithm
TM	Trade Mark
Wed	Wednesday

11- The Next Article on This Topic

In this article we made some benchmarking between SHA, AES-GCM and ECC (ECDSA) algorithms but all of their code is developed to run on CPU and in single-threaded mode only.

In the next article, the benchmarking will be about:

- Running on CPU: single thread VS multi thread
- Running on CPU: on Windows VS on Linux
- Running on CPU (conventional) VS using INTEL oneAPI library
- Running on CPU VS running on GPU (Nvidia CUDA)