Performance Analysis of Encryption Capabilities of ARM-based Single Board Microcomputers

Sandor R. Repas

Abstract—**In the few years since the Raspberry Pi was released in 2012, countless microcomputers based on the ARM architecture have been introduced. Their small size, high performance relative to their power consumption, and the ability to run the popular Linux operating system make them ideal for a wide range of tasks. Information security is an area of particular importance. Different encryption and encoding algorithms play an important role in almost all areas of information security. However, these algorithms are very computationally intensive, so it is important to investigate which microcomputers can be used for these tasks, and under which trade-offs.**

The performance of ten different microcomputers is investigated and presented for the application of common symmetric and public-key encryption and decryption, digest creation and message authentication protocols, such as RSA, AES, HMAC, MD5, SHA.

Reliable encryption requires the generation of reliable (pseudo)random numbers (Cryptographically Secure Random Numbers, CSRN), and microcomputers based on ARM SoCs usually have hardware implemented (pseudo)random number generators. The applicability of the random number generators of different microcomputers are investigated and presented; test methods are described, and recommendations are made.

Index Terms—**ARM; encryption; performance; security; random numbers**

I. INTRODUCTION

As semiconductor technologies continue to evolve, microprocessors and microcontrollers have emerged that can be produced at ever lower cost and with ever lower power consumption, and offer ever higher performance. As the degree of integration has increased, it has become possible to produce integrated circuits that incorporate both the microprocessor and the additional circuitry (e.g. memory, graphics, and USB controllers.). These chips (System on a Chip, SoC) are used to build the increasingly popular smart phones, and the popularity of smart handsets is accelerating the development of these circuits. Increasingly powerful SoCs have also enabled the emergence of Single Board Computers (SBCs), the best known of which is the ARM architecture-based [1] Raspberry Pi [2] with a 700 MHz single-core processor and 256 MB of memory, released in 2012. Thanks to rapid development, SBCs with eight cores and 2GB of memory are now available [3].

Today information security plays an increasingly important role, with encryption, decryption, digital signatures and signature verification being of high importance. However, encryption operations are mathematical operations with a very high computational demand. With the proliferation of increasingly powerful yet cost-effective SBCs, an important question is what encryption capabilities SBCs have, and thus how effectively they can be applied in the field of information security.

In the following, we present in detail our methods used to investigate the encryption capabilities of SBCs, as well as the results.

II. SUMMARY OF THE CURRENT RESEARCH RESULTS

Many papers have been published on Raspberry Pi and other SBCs, but none of them have explicitly investigated their encryption capabilities.

In [4], the authors present, among other results, their solution for the Raspberry Pi to create a secure TFTP (Secure Trivial File Transfer Protocol) to ensure the security of remote updates.

Researchers have investigated the performance of BeagleBone Black [5], BeagleBone, and Raspberry Pi SBCs using LMbench [6], and their proprietary application (CoAP, Constrained Application Protocol) to measure performance on constrained devices [7]. They concluded that the SBCs were less than half the speed of a modern computer, and the BeagleBone Black had the lowest latency of the three devices. Their important conclusion is that for IoT (Internet of Things) applications, faster and more expensive external memory has significantly less impact on the performance of an SBC than the type of the processor. In addition, running the graphical interface did not have a significant impact on performance.

In two papers, researchers presented results of memory and processor performance tests on four different ARM platforms in [8] and [9]. Their measurements were also compared to Intel Atom [10] processors, which produced similar results, but with significantly lower power consumption on ARM SoC-based

S. R. Repas is with the Department of Telecommunications, Széchenyi István University, Győr, Hungary (e-mail: repas.sandor@sze.hu).

DOI: 10.36244/ICJ.2023.2.6

TABLE I
THE MOST IMPORTANT PARAMETERS OF THE SELECTED SBCS

Model	CPU architecture	SoC type	CPU cores (pcs)	CPU freq. (GHz)	RAM size (GB)
Banana Pi	Cortex A7	AllWinner A20	2	1	1
Banana Pi M2	Cortex A7	AllWinner A31s	4	1	1
BeagleBone Black	Cortex A8	TI AM3359	1	1	0.5
ODROID-C1	Cortex A5	Amlogic S805	4	1.5	1
ODROID-U3	Cortex A9	Samsung Exynos 4412	4	1.7	2
ODROID-XU3 Lite	Cortex A15+A7	Samsung Exynos 5422	4+4	1.8+1.3	2
Orange Pi Mini	Cortex A7	AllWinner A20	2	1	1
Orange Pi Plus	Cortex A7	AllWinner H3	4	1.6	1
Raspberry Pi Model B+	1176JZ(F)-S	Broadcom BCM2835	1	0.7	0.5
Raspberry Pi 2 Model B+	Cortex A7	Broadcom BCM2836	4	0.9	1

devices. The authors also point out that this may change in the future.

Using BeagleBoard and PandaBoard [11], researchers investigated the potential of using ARM SoC-based devices in HPC (High Performance Computing) applications [12], with a focus on computing performance and power consumption. The authors concluded that due to the high power consumption of devices that are redundant for HPC applications (e.g. USB, HDMI, VGA, etc.), SBCs based on generic SoCs are not well suited for HPC designs.

In [13], the authors investigated six different types of SBCs from several aspects, with the aim of finding out the performance of a heterogeneous cluster built from different SBCs in discrete-time simulations performed in parallel as described in [14] and [15]. Their important result is that multicore performance should be the primary consideration in the calculations.

These publications do not investigate the performance of SBCs during encryption operations; therefore, it is necessary to develop methods for measuring this and to perform the tests.

III. TEST METHODS

The devices chosen for the performance tests and the test methods applied are described below.

A. Selection of Devices

In selecting the SBCs, the primarily following criteria were considered:

- The Raspberry Pi is a must due to its pervasiveness, as it greatly increases the usability of the results.
- It is important to measure as many different SoCs as possible, thus providing a comprehensive picture for comparing each SoC.
- Include in the tests two SBCs from different manufacturers, based on the same SoC. This should help to find out how much of the performance depends on the SoC and how much on the external components (e.g. memory) used with the SoC.
- At least one SBC based on a SoC using the Big-Little [16] architecture will be investigated. In this way, the advantages and disadvantages of such an SoC will be identified.

Table I shows the main parameters of the selected SBCs. (The datasheets for each SoC were not always made available by the manufacturers, so I could not include some parameters, such as cache size.)

B. Test Environment for the Encryption Performance Measurements

Linux was installed on all SBC devices to perform the tests. If the manufacturer provides or recommends a Linux version for the device, that version was used. In all cases, we tried to make only the most necessary changes, avoiding any modifications that could affect performance. The only exception to this was disabling the launch of the graphical interface on all devices so that it did not affect the measurement results.

To perform the measurements, we needed to implement the network shown in Fig. 1. The measurement process was started from the laptop at the top of the figure and its progress could also be monitored from there. The server on the left of the figure controlled the measurement, and collected and pre-processed the data. The ten SBCs tested are shown at the bottom of the figure. To ensure comparability, all measurements were also performed on the Sun Sunfire X2100 M2 computer on the right side of the figure, which contained an Opteron 1222 dual core CPU and 4 pieces of 2GB DDR2-5300 ECC RAM modules.

IV. MEASUREMENTS AND RESULTS

To ensure accurate results, the measurements were automated using bash shell scripts. Each measurement was repeated 16 times, of which only the results of the last 11 times

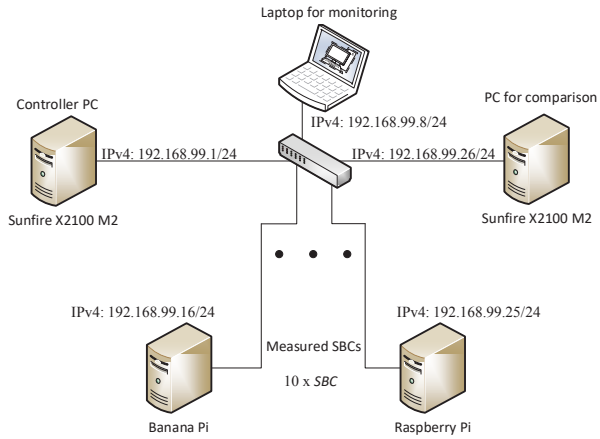

Fig. 1. Topology of the test network for the measurements

were processed. This avoided the influence of the storage system's speed during the measurements (by using cache). The openssl program was used for the measurements. Where it is applicable, measurements were also made using one and then all cores. The measurements were also performed on the Sunfire X2100 M2 computer, making comparison easier.

In order to avoid errors, the results were collected in text files, pre-processed using awk scripts and then evaluated.

In the following, the measurement results will be described and then evaluated in detail.

A. *Symmetric Key Encryption*

Today's most widely used symmetric key encryption is the Advanced Encryption Standard (AES) using the Rijndael algorithm [17]. It is also used with 128, 192 and 256-bit-long keys. Its use is very widespread. Its performance in file encryption has been investigated in Cipher Block Chaining (CBC) mode, which greatly increases the protection against algorithmic attacks.

The measurements were performed using the openssl speed command, with all three key lengths, 8k block size, on 1 and then with multiple threads. The use of multiple threads allowed the simultaneous use of multiple CPU cores.

1) *Single Thread Results*

The average values of the speed results obtained in the runs is shown in Table II, while the standard deviations are shown in Table III. The averages are visualized for better comparison in Fig. 2.

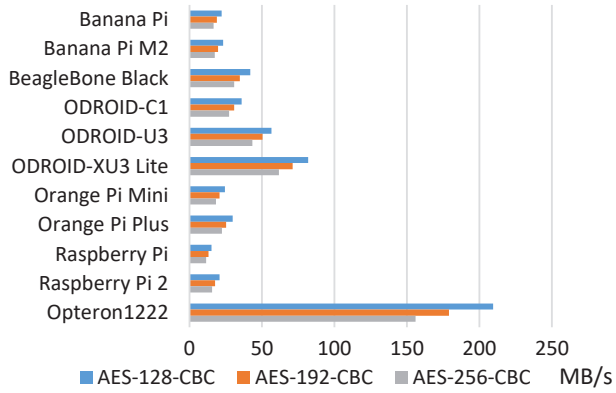

Fig. 2. Average speed values of AES-CBC encryption (MB/s), 1 thread, 8k block

TABLE II
AVERAGE SPEED VALUES OF AES-CBC ENCRYPTION (MB/S), SINGLE THREAD, 8K BLOCK

Model	AES-128	AES-192	AES-256
Banana Pi	22.06	18.88	16.60
Banana Pi M2	23.15	19.74	17.49
BeagleBone Black	41.85	34.70	30.82
ODROID-C1	35.95	30.86	27.26
ODROID-U3	56.59	50.44	43.27
ODROID-XU3 Lite	81.85	71.05	61.77
Orange Pi Mini	24.45	20.81	18.33
Orange Pi Plus	29.72	25.32	22.29
Raspberry Pi Model B+	15.06	13.04	11.50
Raspberry Pi 2 Model B+	20.62	17.58	15.59
Opteron 1222	209.50	179.08	156.01

The first column of Table II shows the type of SBC tested. The second column is the average of the amount of data encrypted per second in MB when using 128-bit key length AES CBC (AES-128-CBC). The third column contains the average of the data volume encoded per second using AES-192-CBC, and the last column contains the average of the data volume encoded per second using AES-256-CBC. The corresponding columns in Table III contain the standard deviation values for the calculated mean values.

2) *Discussion of the Results*

The analysis of the values shows that:

- Odroid-XU3 Lite is the fastest, Odroid-U3 is the second, and BeagleBone Black is the third one.
- The Raspberry Pi is the slowest.
- The two SBCs based on the same SoC produced different results (Banana Pi and Orange Pi Mini), but the difference is only around 10%.
- The standard deviation values are low compared to the average values, so the measured values are stable for the devices tested.
- The performance of each SBC relative to the other is not significantly affected by the key length used for encoding.
- Even the speed of the fastest SBC is a fraction of that of the Opteron 1222-based system (e.g. for AES-128-CBC: 209.5/81.85=2.56).

3) *Multi-thread Results*

Multi-threaded runs have been used to test SBCs based on SoCs that contain multiple processor cores. In each case, the measurement was performed on as many threads as the processor has in the given SoC, so that all cores participated in the coding and the aggregate performance could be measured. In the case of Odroid-XU3 Lite using the Big-Little architecture, the measurement was also performed using 4 threads due to the different core speeds. To present the results, we have chosen the AES-256-CBC encoding values, which are shown in Table IV.

The second column shows how many threads were used for each measurement. The third column shows the average amount of the encoded data from the last 11 measurements. The data in the fourth column has already been presented for the single

TABLE III
STANDARD DEVIATION OF SPEED VALUES OF AES-CBC ENCRYPTION (MB/S), SINGLE THREAD, 8K BLOCK

Model	AES-128	AES-192	AES-256
Banana Pi	0.04	0.03	0.03
Banana Pi M2	0.07	0.04	0.04
BeagleBone Black	0.08	0.06	0.05
ODROID-C1	0.20	0.16	0.14
ODROID-U3	0.48	0.42	0.37
ODROID-XU3 Lite	0.07	0.01	0.03
Orange Pi Mini	0.04	0.08	0.05
Orange Pi Plus	0.03	0.00	0.00
Raspberry Pi Model B+	0.00	0.00	0.00
Raspberry Pi 2 Model B+	0.01	0.03	0.01
Opteron 1222	0.13	0.09	0.07

TABLE IV
AVERAGE SPEED VALUES OF AES-256-CBC ENCRYPTION (MB/S), MULTIPLE THREADS, 8K BLOCK

Model	Thr.	AES-256 *x* threads (MB/s)	AES-256 1 thread (MB/s)	Rel. acc.	Std. dev.
Banana Pi	2	33.22	16.60	2.00	0.02
Banana Pi M2	4	69.96	17.49	4.00	0.02
ODROID-C1	4	109.57	27.26	4.02	0.04
ODROID-U3	4	88.43	43.27	2.04	7.38
ODROID-XU3 L.	4	195.01	61.77	3.16	1.01
ODROID-XU3 L.	8	276.21	61.77	4.47	2.20
Orange Pi Mini	2	36.36	18.33	1.98	0.01
Orange Pi Plus	4	46.55	22.29	2.09	6.40
Raspberry Pi 2	4	62.31	15.59	4.00	0.01
Opteron 1222	2	310.88	156.01	1.99	0.32

thread measurement, it is only presented again here for ease of reference.

The relative acceleration in column 5 is the ratio of the average speeds in the fourth and third columns. The last column contains the standard deviation values for the third column.

4) Discussion of the Results

The analysis of the values shows that:

- Odroid-XU3 Lite is the fastest, Odroid-C1 is the second and Odroid-U3 is the third.
- The Odroid-U3 and Orange Pi Plus produce large standard deviations. The measured performance is not constant, shows large fluctuations and the system behavior can only be estimated imprecisely.
- Among devices with low standard deviation, the fastest SBCs are Odroid-XU3 Lite, Odroid-C1, Banana Pi M2 and Raspberry Pi 2.
- The relative acceleration is almost equal to the number of cores used, except for the two devices with high standard deviation and the Big-Little architecture Odroid-XU3 Lite. By increasing the number of cores, a linearly proportional acceleration is obtained.
- The performance of the fastest SBC (Odroid-XU3 Lite with 8 threads) now approaches that of the Opteron 1222 system.

5) Message Digest and Authentication

- The use of hash function and message authentication code is also essential to ensure secure communication. They are generally used to ensure integrity and detect tampering. The analyzed protocols and some of their characteristics:
- MD5
  - 128 bit-long digest
  - Not secure, but used for compatibility reasons.
- SHA1
  - 160 bit-long digest
  - No longer recommended
  - Widespread, widely known and used.
- SHA256
  - 256 bit-long digest
  - Recommended for use.
- SHA512
  - 512 bit long digest
  - Not always recommended due to its slowness.
- HMAC
  - MD5-based message authenticator
  - Keyed-Hash Message Authentication Code

6) Single Thread Results

The average values of the results obtained in the runs is shown in Table V, while the standard deviations are shown in Table VI. The averages are visualized for better comparison in Fig. 3.

The structure of the table is very similar to the one used for the previous tables, so it is not explained in detail.

7) Discussion of the Results

The analysis of the values shows that:

- The fastest is Odroid-XU3 Lite, the second is Odroid-U3.
- The slowest is Raspberry Pi.
- The two SBCs based on the same SoC produced different results (Banana Pi and Orange Pi Mini), but again the difference is only around 10%.
- The standard deviation values are low compared to the average values, so the measured values are stable for the devices tested.
- The relative performance of each SBC is not necessarily the same for different tasks (e.g. Odroid-C1 and BeagleBone Black MD5: 118.71/102.88=1.15, while for SHA1: 73.62/77.57=0.95).
- Even the speed of the fastest SBC is a fraction of that of the Opteron 1222-based system (e.g. for MD5: 562.56/240.15=2.34, while for SHA512: 290.29/95.70=3.03).

TABLE V
AVERAGE SPEED VALUES OF DIGEST AND MESSAGE AUTHENTICATION (MB/S), SINGLE THREAD, 8K BLOCK

Model	MD5	SHA 1	SHA 256	SHA 512	HMAC
Banana Pi	82.49	44.29	26.01	22.35	82.62
Banana Pi M2	82.96	46.30	27.93	14.12	82.83
BeagleBone Black	102.88	77.57	56.30	37.88	104.70
ODROID-C1	118.71	73.62	43.43	36.66	118.88
ODROID-U3	200.43	118.47	69.70	59.88	200.17
ODROID-XU3 L.	240.15	158.88	91.43	95.70	242.87
Orange Pi Mini	91.53	48.22	28.33	24.41	91.90
Orange Pi Plus	111.43	58.67	34.52	29.69	111.98
Raspberry Pi	51.92	29.12	18.79	9.11	53.15
Raspberry Pi 2	73.77	41.19	24.84	12.56	73.65
Opteron 1222	562.56	430.60	186.18	290.29	562.81

TABLE VI
STANDARD DEVIATION OF SPEED VALUES OF DIGEST AND MESSAGE AUTHENTICATION (MB/S), SINGLE THREAD, 8K BLOCK

Model	MD5	SHA 1	SHA 256	SHA 512	HMAC
Banana Pi	0.23	0.11	0.05	0.02	0.16
Banana Pi M2	0.13	0.11	0.04	0.03	0.17
BeagleBone Black	0.08	0.10	0.09	0.07	0.15
ODROID-C1	0.72	0.37	0.23	0.19	0.69
ODROID-U3	2.05	1.21	0.68	0.54	1.80
ODROID-XU3 L.	0.18	0.25	0.14	0.08	0.20
Orange Pi Mini	0.53	0.47	0.05	0.02	0.20
Orange Pi Plus	0.03	0.01	0.00	0.00	0.08
Raspberry Pi	0.10	0.01	0.03	0.00	0.03
Raspberry Pi 2	0.08	0.03	0.06	0.01	0.15
Opteron 1222	1.11	0.17	0.85	0.08	0.71

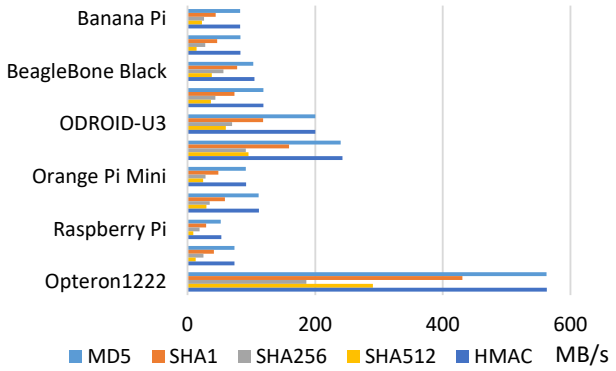

Fig. 3. Average speed values of digest and message authentication (MB/s), single thread, 8k block

TABLE VII
AVERAGE SPEED VALUES OF SHA256 MESSAGE DIGEST CREATION (MB/S), MULTIPLE THREADS, 8K BLOCK

Model	Thr.	SHA256 *x* threads (MB/s)	SHA256 1 thread (MB/s)	Rel. acc.	Std. dev.
Banana Pi	2	52.04	26.01	2.00	0.08
Banana Pi M2	4	111.79	27.93	4.00	0.13
ODROID-C1	4	174.79	43.43	4.02	0.21
ODROID-U3	4	260.81	69.70	3.74	112.01
ODROID-XU3 L.	4	317.19	91.43	3.47	6.99
ODROID-XU3 L.	8	442.94	91.43	4.84	31.49
Orange Pi Mini	2	56.32	28.33	1.99	0.13
Orange Pi Plus	4	72.00	34.52	2.09	32.10
Raspberry Pi 2	4	99.54	24.84	4.01	0.09
Opteron 1222	2	370.94	186.18	1.99	1.72

TABLE VIII
AVERAGE EXECUTION TIMES OF 100 PIECES OF ENCRYPTION AND DECRYPTION BY RSA ALGORITHM (IN SECONDS) – LOWER IS BETTER!

Model	2048 bit encr.	2048 bit decr.	4096 bit encr.	4096 bit decr.
Banana Pi	2.71	7.89	3.14	38.54
Banana Pi M2	2.43	7.07	2.82	34.42
BeagleBone Black	3.03	7.68	3.42	34.58
ODROID-C1	1.74	5.05	2.02	24.56
ODROID-U3	1.10	3.30	1.28	16.12
ODROID-XU3 L.	1.18	2.73	1.31	11.41
Orange Pi Mini	2.88	7.62	3.28	35.33
Orange Pi Plus	2.22	6.16	2.55	29.27
Raspberry Pi	6.63	15.19	7.36	64.64
Raspberry Pi 2	2.93	8.15	3.37	38.84
Opteron 1222	0.50	0.74	0.52	1.89

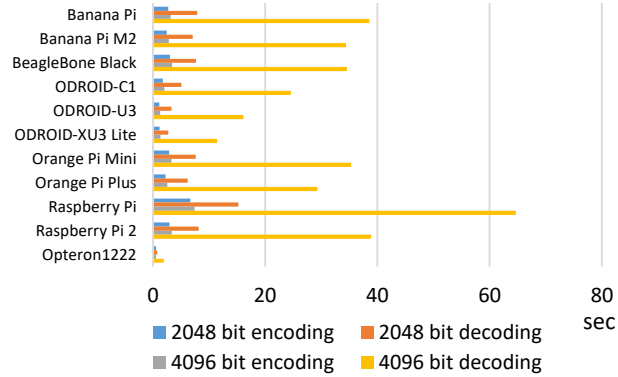

Fig. 4. Average execution times of 100 pieces of encryption and decryption by RSA algorithm (in seconds) – Lower is better!

8) Multi-thread Results

To present the results, we have chosen the SHA256 digest generation, which are shown in Table VII. The structure of the table is the same as in Table IV.

9) Discussion of the Results

The analysis of the values shows that:

- Odroid-XU3 Lite is the fastest, Odroid-U3 is the second, and Odroid-C1 is the third.
- The Odroid-XU3 Lite, Odroid-U3 and Orange Pi Plus produce large standard deviations (The ratio of speed to standard deviation for devices Odroid-U3 and Orange Pi Plus are almost identical). The measured performance is not constant, shows significant fluctuations and the system behavior is not predictable.
- Among devices with low standard deviation, the fastest SBCs are Odroid-C1, Banana Pi M2 and Raspberry Pi 2.
- The relative acceleration is almost equal to the number of cores used, except for the three devices with high standard deviation. By increasing the number of cores, a linearly proportional acceleration is obtained.

B. Public key Cryptography

Public key cryptography is generally used for the encrypted transmission of symmetric keys, and the creation of digital signatures. The methods currently in use are extremely computationally intensive and slow, so they are usually used in combination with symmetric encryption for transmitting large amounts of data. The most widely used RSA encryption, developed by Ron Rivest, Adi Shamir and Leonard Adleman, was investigated with 2048 and 4096 bit long keys, as the minimum key length currently recommended for adequate security is 2048 bits.

1) RSA Encryption Results

In the RSA encryption, the same randomly generated file was encrypted and decrypted with the same key pair for each SBC. To test the performance of encryption using the 2048-bit key, one 1920-bit file was encrypted or decrypted 100 times in each measurement cycle for the 2048-bit key, and a 4000-bit file for the 4096-bit key (RSA is only able to encrypt data to a maximum amount equal to the key size, minus padding and header data.). The results of the test are presented in Table VIII and graphically displayed in Fig. 4.

2) Discussion of the Results

The analysis of the values shows that:

- Odroid-U3 is the fastest for encoding, Odroid-XU3 Lite is the second, while for decoding, the order is reversed. Odroid-C1 is the third one in all cases.
- The slowest is the Raspberry Pi.
- The two SBCs based on the same SoC produced different results (Banana Pi and Orange Pi Mini), with the Banana Pi being faster in the encoding operation and the Orange Pi Mini in the decoding operation. In all cases, the differences were below 10% for the two SoCs.
- The relative performance of each SBC to the other is not necessarily the same for the different tasks.

- Even the speed of the fastest SBC is a fraction of that of the Opteron 1222-based system (e.g. for 4096-bit key length decoding: 11.41/1.89=6.04).

C. Random Number Generation

Random numbers and their generation play a key role in cryptography. Without the right random numbers, secure encryption cannot be done. Generating true random numbers (TRNs) with computers is almost impossible. There are several algorithms for generating pseudo random numbers (PRN), which are recommended for different purposes. Some are explicitly not recommended for encryption tasks, while others are suitable (e.g. Dual_EC_DRBG).

According to the available information, (almost) all ARM-based SoCs under investigation have some kind of hardware random number generator (HWRNG). The following information has been extracted from publicly available documentation:

- Amlogic S805: Built-in LSFR Random number generator.
- TI AM3359: Crypto Hardware Accelerators (AES, SHA, PKA, RNG).
- Allwinner A20: 160-bit hardware PRNG with 192-bit seed.
- Allwinner A31: 160-bit hardware PRNG with 192-bit seed.
- Allwinner H3: 160-bits hardware PRNG with 175-bits seed. 256-bits TRNG.
- Samsung Exynos 5422: Hardware Crypto Accelerators: AES, DES/3DES, ARC4, SHA-1/SHA-256/MD5/HMAC/PRNG, TRNG, PKA, and Secure Key Manager.

The two SoCs produced by Broadcom also contain some form of HWRNG, but no documentation has been found.

1) Support

HWRNG is not well documented for any of the SoCs examined. Only partial information could be found.

A common Linux driver for all Allwinner SoCs HWRNG is produced, but at the time of testing it was not yet working reliably.

No information could be found for Samsung SoCs.

The HWRNG of the TI AM3359 SoC is supported in the new kernels, however the Linux released for the BeagleBone Black does not yet have this kernel.

The Amlogic S805 in Odroid-C1 is supported.

The two Broadcom SoCs found in the Raspberry Pi SBCs are also supported.

2) Tests

a) Entropy

Due to the shortcomings of the documentation, we were only able to examine the quality of the random numbers to a limited extent: we only performed statistical analysis on the (pseudo)random numbers generated by the SBCs. The most commonly used tools for statistical analysis and their latest versions are the following:

- Diehard [18]
- Dieharder 3.31.1 [19]
- NIST Special Publication 800-22rev1a 2.1.2 [20]
- Ent [21]
- rngtest [22]
- TestU01 1.2.3 [23]
- Practically Random 0.94 [24]

To perform the tests, 10GB of (pseudo)random numbers were generated and analyzed. The results of the analyses are summarized in Table IX.

TABLE IX
RESULTS OF THE STATISTICAL ANALYSIS OF THE GENERATED RANDOM NUMBERS

Model	Dieharder	Ent X^2 distribution	NIST 800-22
ODROID-C1	Passed	suspect (98,71%)	1 error
Raspberry Pi	1 weak (bitstream)	Ok (59,7%)	Success
Raspberry Pi 2	1 weak (rank 32x32)	almost suspect (90,83%)	1 error

TABLE X
THE SPEED VALUES OF THE HARDWARE RANDOM NUMBER GENERATORS

Model	Speed
ODROID-C1	7.3MB/s
Raspberry Pi	105kB/s
Raspberry Pi 2	147kB/s

The results show that none of the random numbers generated by the systems can be used for encryption. However, the Linux kernel is prepared to use multiple sources for random number generation, so the weakness of one source is not necessarily a problem, but the use of HWRNG can speed up random number generation. It is also important to note that the lack of proper documentation (hence knowledge of how the SoC HWRNG works) is also a drawback for its use in encryption applications.

b) Visual Analysis

Rather just for interest, we also visually examined the quality of the random numbers produced. Fig. 5. shows the images created from the generated random numbers in 256 by 256 grids of (24 bit) RGB values. A close inspection of the figures does not reveal any anomalous repetition or shape.

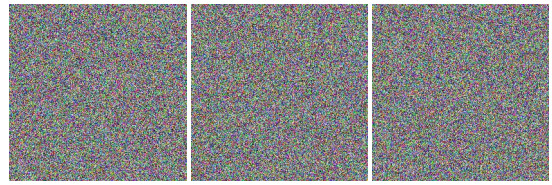

Fig. 5. Images created from random numbers generated by Odroid-C1, Raspberry Pi, and Raspberry Pi 2

c) Speed

Table X shows the speed of random number generation using the HWRNG for the three SBCs under study.

V. CONCLUSIONS

The results show that even the slowest SBC has sufficient performance to perform the encryption task required in a normal application. For more specialized applications with higher amount of encrypted traffic, the characteristics of each SBC need to be taken into account.

If only one processor core can be used efficiently due to the application, then the BeagleBone Black is recommended rather than Raspberry Pi Model B+ (While it is worth noting that the Odroid-XU3 Lite produces the highest speed with single thread in almost all cases.).

The use of Odroid-XU3 Lite, Odroid-U3 and Orange Pi Plus should be avoided due to their performance fluctuation under heavy load. Further investigation is required to determine the reasons.

Due to its performance and predictable, stable operation, we recommend using Odroid-C1 for encryption applications (moreover, its HWRNG is the fastest among the devices tested). For its speed, the Bana Pi M2 is also a good choice. For its coverage, and thus support and awareness, the Raspberry Pi 2 is recommended.

The HWRNGs we tested do not provide reassuring entropy, but we found the fewest problems with Raspberry Pi.

Different memory speeds of SBCs based on the same SoCs do not significantly affect the performance of the SBC.

VI. SUMMARY AND FURTHER RESEARCH

Based on the results, SBCs are cost-effective, energy-efficient devices that are well suited for information security applications.

Further measurements of encryption capabilities (e.g. HTTPS, SCP, SFTP, IPsec) need to be developed and performed to determine their potential applications. The performance of network transmissions is an important area to be investigated.

Finally, it is important to investigate the virtualization capabilities of each SBC, as well as its compatibility with other operating systems in the security domain. (e.g. OpenBSD, FreeBSD.)

ACKNOWLEDGMENT

We thank HunNet-Média Ltd. for providing us with the single-board computers for the studies, thus contributing to the publication.

REFERENCES

[1] https://www.arm.com/
[2] Raspberry Pi Foundation, https://www.raspberrypi.org/
[3] ODROID-XU3 Lite, https://www.hardkernel.com/shop/odroid-xu3-lite/
[4] M. A. M. Isa et al., "A Series of Secret Keys in a Key Distribution Protocol," in *Transactions on Engineering Technologies, London*, UK, 2-4 July 2014, pp. 615–628. DOI: 10.1007/978-94-017-9804-4_43
[5] BeagleBone Black, https://beagleboard.org/black
[6] LMbench - Tools for Performance Analysis, http://lmbench.sourceforge.net/
[7] C. P. Kruger and G. P. Hancke, "Benchmarking Internet of things devices," in *Proc. 2014 12th IEEE International Conference on Industrial Informatics (INDIN)*, Porto Alegre, Brazil, 27-30 July 2014, pp. 611–616. DOI: 10.1109/INDIN.2014.6945583
[8] R. G. Reed et al., "A CPU benchmarking characterization of ARM based processors," *Computer Research and Modeling*, vol. 7, issue 3, pp. 581–586. 2015. DOI: 10.20537/2076-7633-2015-7-3-505-509
[9] G. T. Wrigley, R. G. Reed, B. Mellado, "Memory benchmarking characterisation of ARM-based SoCs," *Computer Research and Modeling*, vol. 7, issue 3, pp. 607–613. 2015. DOI: 10.20537/2076-7633-2015-7-3-607-613
[10] Intel Atom processor, https://www.intel.com/content/www/us/en/products/details/processors/atom.html
[11] https://hu.mouser.com/new/pandaboardorg/pandaboardES/
[12] E. L. Padoinetal., "Evaluating Performance and Energy on ARM-based Clusters for High Performance Computing," in *Proc. 2012 41st International Conference on Parallel Processing Workshops*, Pittsburgh, USA, 10-13. September 2012, pp. 165–172. DOI: 10.1109/ICPPW.2012.21
[13] G. Lencse and S. Répás, "Method for Benchmarking Single Board Computers for Building a Mini Supercomputer for Simulation of Telecommunication Systems," in *Proc. 2015 38th International Conference on Telecommunications and Signal Processing (TSP)*, Prague, Czech Republic, 9-11 July 2015, pp. 246–251. DOI: 10.1109/TSP.2015.7296261
[14] G. Lencse, I. Derka, L. Muka, "Towards the Efficient Simulation of Telecommunication Systems in Heterogeneous Distributed Execution Environments," in *Proc. 2013 36th International Conference on Telecommunications and Signal Processing (TSP)*, Rome, Italy, 2-4 July 2013, pp. 304–310. DOI: 10.1109/TSP.2013.6613941
[15] G. Lencse and I. Derka, "Measuring the Efficiency of Parallel Discrete Event Simulation in Heterogeneous Execution Environments", *Acta Technica Jaurinensis*, vol. 9. no. 1. pp. 42–53, DOI: 10.14513/actatechjaur.v9.n1.394
[16] H. D. Cho, K. Chung, T. Kim, "Benefits of the big. LITTLE Architecture", TechOnline, https://www.techonline.com/tech-papers/benefits-of-the-big-little-architecture/
[17] Announcing the ADVANCED ENCRYPTION STANDARD (AES), Federal Information Processing Standards Publication 197, National Institute of Standards and Technology, November 26, 2001. Available: https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.197.pdf
[18] The Marsaglia Random Number CDROM including the Diehard Battery of Tests of Randomness, https://web.archive.org/web/20160125103112/ http://stat.fsu.edu/pub/diehard/
[19] Dieharder: A Random Number Test Suite, https://webhome.phy.duke.edu/~rgb/General/dieharder.php
[20] A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, https://csrc.nist.gov/publications/detail/sp/800-22/rev-1a/final, DOI: 10.6028/NIST.SP.800-22r1a
[21] ENT A Pseudorandom Number Sequence Test Program, https://www.fourmilab.ch/random/
[22] rng-tools, https://github.com/nhorman/rng-tools
[23] TestU01, http://simul.iro.umontreal.ca/testu01/tu01.html
[24] PractRand, https://github.com/MartyMacGyver/PractRand

Sandor R. Repas received his BA in Business Management University Hungary in 2009, BSc in Electrical Engineering from the Óbuda University in 2011, MSc in Electrical Engineering from the Széchenyi István University in Administration and from the Corvinus of Budapest, Budapest 2013 and MSc in Defence C3 System Manager from National University of Public Service in 2017. He received his PhD in computer science from the Széchenyi István University in 2018.

He is an Associate Professor at the Széchenyi István University, Győr Hungary. The main field of his research is the IPv6 implementation technologies. His other favorite topics are computer networking and information security. He has several certificates from Microsoft, Cisco, ISACA and other vendors.