

Cryptoanalysis of the 340-bit RSA Algorithm using SBC

Nelson Darío Pantoja¹, Anderson Felipe Jiménez²,
Siler Amador Donado³, Katerine Márceles⁴

^{1,2,4} Institución Universitaria Colegio Mayor del Cauca, Popayán, Colombia

³ Universidad del Cauca, Popayán, Colombia

¹ dariopantoja@unimayor.edu.co, ² afjimenez@unimayor.edu.co

³ samador@unicauca.edu.co, ⁴ kmarceles@unimayor.edu.co

Abstract. This work used computer devices selected from requirements mainly aimed at their physical processing components to establish through metrics the importance of the hardware when conducting performance tests on the different cryptanalysis techniques applicable to be RSA algorithm. In order to select the adequate technique for cryptanalysis, performance tests were carried out among those chosen devices mentioned in the adequate technique was selected in terms of time and efficiency. To avoid compromising the integrity of the results, the tests were run similar environments and hardware and software levels, using Kali Linux as operating system and the Python language for the cryptanalysis technique, given compatibility with the three machines and their performance.

Keywords: RSA, cryptanalysis, SBCs, devices, server.

1 Theory of the Domain and Prior Works

The RSA algorithm is currently one of the most secure [1] to establish communication between an emitter and they receptor, hence, a rupture can lead to many consequences [2]. Its security is due to it being a cryptographic system that uses two keys, one private and one public, which in turn use extremely large primary numbers, generally 2048 bits – as recommended, requiring high computational efforts to decipher it by using factorization methods. With the frenzied progress in technology, multiple devices are available with computationally diverse capacities, which can be used to carry out different tasks and in this case specifically answer the question: what hardware components are important when executing a cryptanalysis process to the RSA algorithm? If cryptanalysis depends on the confrontational resources offered by the machine, it may be stated that greater resources would mean greater efficacy in executing cryptanalysis.

Given that the analysis of the paper Cryptanalysis of RSA: A Survey [3] proposes that the security of the RSA cryptographic system cannot be doubted because to date a devastating attack has been found in the failures that could have occurred are commonly due to poor implementations of the system.

Recently, the task of conducting cryptanalysis to the RSA algorithm has led researchers to using less conventional methods for this purpose; one of the best known

key extraction through acoustics, explained in the paper RSA Key Extraction via

Low-Bandwidth Acoustic Cryptanalysis [4], which although effective in some cases still seems a rather impractical method not applicable to all possible scenarios.

The article Twenty Years of Attacks on The RSA Cryptosystem, [5] BONEH, Dan, lists the most common attacks to accomplish breaking the keys of the RSA algorithm and groups them into four categories, thus, providing cryptanalysis methods candidates for research in this branch.

Mathematical PhD, Hugo Scolnik, in his article Mathematical foundations of the RSA method [6] claims not needing quantum computing to break the keys of the RSA algorithm with a high number of bits, besides providing results measured over time of techniques to decipher said algorithm.

The paper Factorization of 768-Bit RSA Modulus [7] explains the development of the process to achieve the objective of breaking the biggest number of RSA bits known to date; in addition, it mentions the goals previously reached.

1.1 Msieve: Factoring Tecnology

Msieve is a complete package of factorization, which automates the mathematical process in addition to choosing the appropriate algorithm according to the size of the number to factorize, being by default the quadratic sieve technique used for the larger numbers, this last technique optimizes the implementation of the choice of the polynomial that relates to the sieve of the numerical field algorithm, this software works on linux-based systems, which is ideal for implementation in SBC's.

2 Prior Research

2.1 Selection of Devices

To select the devices, a series of activities were undertaken that permitted knowing and classifying – from a set of criteria – the most common low-cost devices. The device selection process established parameters according to the processing capacity related to its CPU power measured in GHz, principally because these components are in charge of performing the arithmetic logic calculations. As a result, the Raspberry Pi 3 was chosen as the optimal, which was most easily accessed for the evaluation, as shown in Figures 1 and 2.

Table 1. List of devices used most often in the market.

NOMBRE	SOC	CPU	GPU	RAM	OS
Raspberry Pi Model B	Broadcom BCM2835	700 MHz ARM1176JZF-S core (ARM11 family)	700 MHz ARM1176JZF-S core (ARM11 family)	256 MB (shared with GPU)	Debian GNU/Linux, Fedora, Arch Linux ARM
MK802	Allwinner A10	1.5GHz? Cortex-A8	MALI400MP OpenGL ES 2.0	1GB / 512MB DDR3	Android 4.0,Puppy Linux, Ubuntu
Mele A1000	Allwinner	1GHz+ Cortex-A8	MALI400MP	512MB	Android 2.3,ubuntu, Debian,

NOMBRE	SOC	CPU	GPU	RAM	OS
	A10		OpenGL ES 2.0	DDR3	puppy, android ics
Rhombus-Tech A10 EOMA-68	Allwinner A10	1.2ghz Cortex A8 ARM Core	MALI400MP OpenGL ES 2.0 GPU	1gb	-
Gooseberry board	A10	1 Ghz - 1.5 Ghz 1.2 Ghz highest stable	Mali 400 MHz	512MB	Android 2.3, ubuntu, Debian, puppy, android ics
Pineriver H24/MiniX S	A10	1.2GHz	Mali400	512MB	Android 2.3/4.0, Puppy Linux, Ubuntu
Smallart UHOST	Allwinner A10	1GHz Cortex-A8	Mali400	1 GB	android 4.0, Puppy Linux, Ubuntu
A13-OLinuXino	Allwinner A13	1GHz A13 Cortex A8	Mali400	512 MB	Linux
VIA APC	VIA WonderMedia 8750	800 MHz ARM11	OpenGL ES 2.0	DDR3 512MB	Android 2.3
BeagleBoard Rev. C4	TI OMAP3530	720 MHz ARM Cortex-A8	PowerVR SGX	256 MB	Android, Ubuntu, Fedora, ArchLinux, Gentoo
BeagleBoard-xM	TI DM3730	1 GHz Cortex-A8	PowerVR SGX	512 MB	Angstrom, Android, Ubuntu, Fedora, ArchLinux, Gentoo
BeagleBone	TI AM3359 Sitara	500MHZ-USB Powered 720MHZ-DC Powered Cortex A8	SGX530	256MB DDR2 (128MB Optional)	Angstrom, Debian, Ubuntu, Fedora, ArchLinux, Gentoo, Sabayon
PandaBoard	OMAP4430	1GHz Dual-core ARM Cortex-A9 MPCore	PowerVR SGX540	1GB DDR2	Ubuntu, Ångström, Android,Supported by Linaro
PandaBoard ES	OMAP4460	1.2 GHz Dual-core ARM Cortex-A9 MPCore	PowerVR SGX540	1GB DDR2	Ubuntu, Ångström, Android,Supported by Linaro
Cotton Candy	Samsung Exynos 4210	1.2 GHz dual-core ARM Cortex-A9	quad-core 200 MHz Mali-400 MP	1 GB	Android, Ubuntu
CuBox	Marvell Armada 510 (88AP510)	800 MHz ARMv7	Vivante GC600	1 GB DDR3-800MHz	Ubuntu 10.04, Android 2.2, Linux kernel 2.6.x or later Android 2.2.x and later
Hawkboard	TI OMAP-L138	300-MHz ARM926EJ	-	128 MByte	Ubuntu, Fedora, Impactlinux
IGEP v2	Ti DM3730	ARM Cortex A8 1GHz	SGX530 @ 200 MHz	512 M	Android, Angstrom, Ubuntu
IGEP COM Proton	DM3730 (optional OMAP3530)	1GHZ ARM CORTEX A8 (720Mhz for OMAP3530)	SGX 530 (200Mhz) (110Mhz for OMAP3530)	512 MBytes	Android, Angstrom, Ubuntu
IGEP COM Module	DM3730 (optional OMAP3530) A	1GHZ ARM CORTEX A8 (720Mhz for OMAP3530)	SGX 530 (200Mhz) (110Mhz for OMAP3530)	512 MBytes	Android, Angstrom, Ubuntu, ..
Gumstix Overo series	AM3703, DM3730, OMAP3503, OMAP3530	ARM Cortex-A8 Up to 1GHz	-	512MB or 256MB	Ubuntu, Android, ..
Origen Board	Exynos4210	1.2GHz Dual Core Cortex-A9	Mali400 MP4	DDR3 1GB	Android, Ubuntu, ..
Nimbus	Marvel Kirkwood 6281	1.2 GHz	-	512MB	Debian
Stratus	Marvel Kirkwood 6281	1.2 GHz	-	512MB	Debian

NOMBRE	SOC	CPU	GPU	RAM	OS
SheevaPlug dev kit (Basic)	Marvel Kirkwood 6281	1.2 GHz ARM9E	-	512 MB	Ubuntu/Debian
GuruPlug Standard	Marvel Kirkwood 6281	1.2 GHz	-	512 MB	Ubuntu/Debian
GuruPlug Display	Marvell ARMADA 168	800MHz	-	512MB	Ubuntu/Debian
DreamPlug	Marvel Kirkwood 6281	1.2 GHz	-	512MB	Ubuntu/Debian
D2Plug	Marvell PXA510	800MHz	-	1GB	Ubuntu/Debian
Trim-Slice series	NVIDIA Tegra 2	1 GHz	-	1 GB DDR2-667	Android, Ubuntu, ..
Snowball	STEricsson Nova A9500	1GHz Dual Cortex A9	Mali 400	1GByte	Linaro (Ubuntu, Android)
i.MX53 Quick Start Board	Freescall i.MX535	1GHz	-	1GB of DDR3	Linaro (Ubuntu, Android)
Genesi Efika MX Smarttop	Freescall i.MX515	ARM Cortex-A8 800MHz	-	512MB	Ubuntu
FriendlyARM Mini 210s	Samsung S5PV210	1 GHz Cortex-A8	PowerVR SGX540	512 MB	Linux-2.6.35, Android 2.3, 4.0, WindowsCE 6.0
Embest DevKit8600	TI's Sitara AM3359	720MHz ARM Cortex-A8	SGX530	512MBytes	Linux 3.1.0, Android 2.3 and WinCE 7
Embest SBC8018	TI AM1808	375MHz ARM926EJ-S	128MByte	128MByte	Linux2.6.33 and WinCE 6.0
Embest SBC8530	TI DM3730	1GHz ARM Cortex-A8	-	512MByte	Linux2.6.32, Android 2.2 and WinCE 6.0
Embest DevKit8500D	DM3730	1GHz ARM Cortex-A8	-	512MB	Linux2.6.32, Android 2.2 and WinCE 6.0.15
TechNexion Infernopak	TI OMAP3530	600Mhz	POWERVR SGX 530	128 MB	Linux 2.6.x, Windows CE 6.0 BSP or Android
TechNexion Thunderpack	TI OMAP3530	600Mhz	POWERVR SGX 530	256 MB	Linux 2.6.x, Windows CE 6.0 BSP or Android
VIA ARTiGO A1200 Fanless	Chipset VIA VX900	1.0GHz x86 VIA Eden X2 L2 Cache 2MB	VIA Chrome 9	2Gb Up to 4GB DDR3	"ordinary" X86 OS
VIA ARTiGO A1150	Chipset VIA VX900H	1.0GHz VIA Eden X2	VIA Chrome 9	2Gb Up to 4GB DDR3	"ordinary" X86 OS
DMP - eBox 3350MX	-	1Ghz Vortex86MX (i586, no CMOV)	-	512MB	i586 compatible OS
DMP - eBox 3310MX-AP	-	933MHz Vortex86MX+	-	1GB DDR2	i586 compatible OS

CPU information	
Vendor	unknown
CPU(s)	4
Model name	ARMv7 Processor rev 4 (v7l)
Frequency	unknown
L2 cache	unknown
More details	

Fig. 1. Characteristics of the Raspberry Pi 3 processor (1.2 GHz, according to the Raspberry official web page). Source: Author's information.

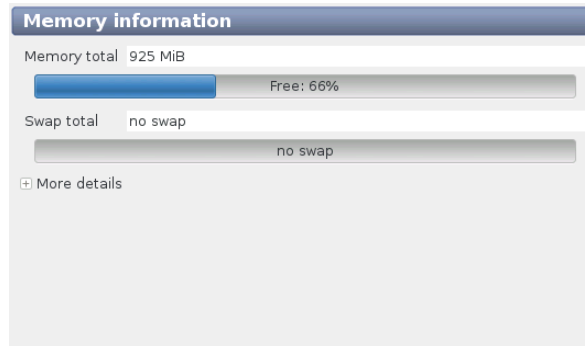


Fig. 2. Raspberry Pi 3 RAM memory information. Source: Author's information.

According to the previous criterion, and to compare results, the same test was evaluated on two more devices with superior characteristics over the first.

The second device was a personal computer (PC) with intermediate characteristics in its hardware, as seen in Figures 3, 4 and 5.

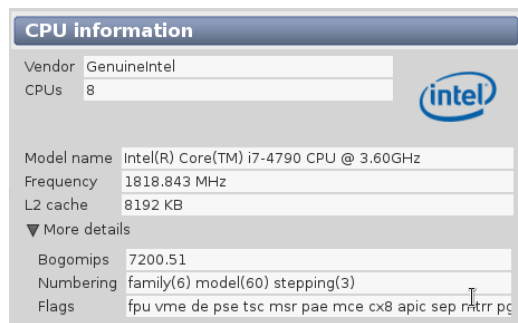


Fig. 3. Characteristics of the processor in the intermediate PC. Source: Author's information.

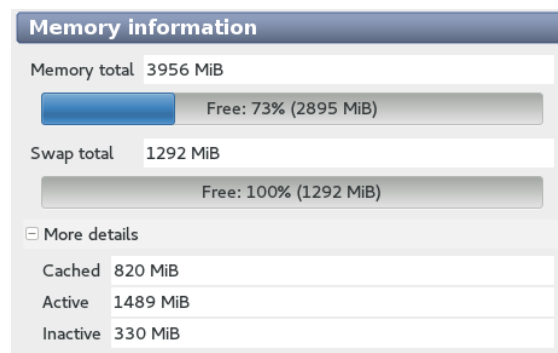


Fig. 4. RAM memory information of the intermediate PC. Source: Author's information.

Finally, a computer with features really higher than the previous ones, being this one generally used like server:

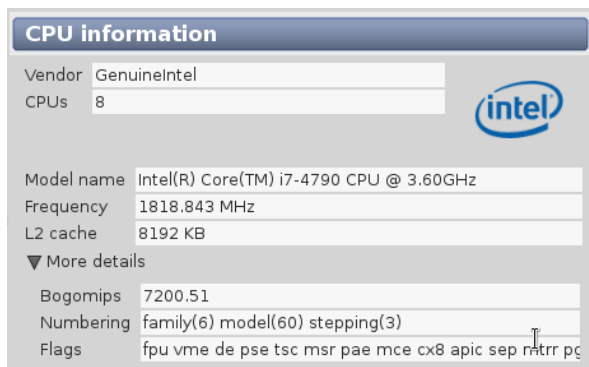


Fig. 5. Characteristics of the processor in the server. Source: Author's information.

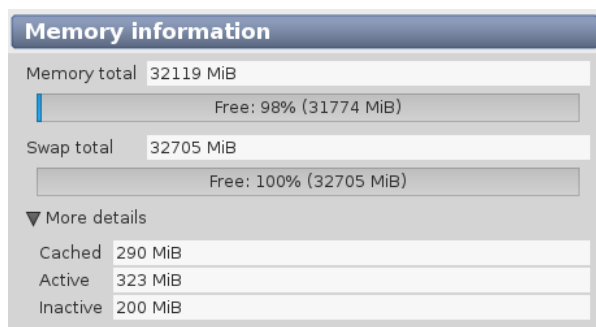


Fig. 6. Server's RAM memory information. Source: Author's information.

2.2 Software

To keep from compromising the integrity of the test results, these were conducted in equal environments regarding software; the three devices used 64-bit Kali Linux 2016.2 as operating system. To execute the cryptanalysis, the msieve open code program was used, which is developed under C language and can be executed in multiple operating systems. With regard to the generation of keys, such was carried out with OpenSSL, which – in turn – generally distributes cryptographic options to web sites for secure HTTPS access.

3 Experiments and Results

The first step established the controlled work environment, which was a computer laboratory with the physical space to conduct the corresponding experiments. This work setting has the selected computer equipment already mentioned. The experiment conducted

were tests with keys generated with OpenSSL; the keys generated were of 100, 256, and 340 bits (Fig. 7).

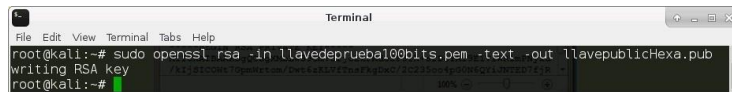


```

root@kali:~# sudo openssl genrsa -out llavedeprueba100bits.pem 100
Generating RSA private key, 100 bit long modulus
.....
e is 65537 (0x10001)
root@kali:~#
    
```

Fig. 7. Example of key generated with OpenSSL. Source: Author's information.

To convert the keys generated into a hexadecimal format, the following OpenSSL commands were employed (Figs. 8, 9, 10, and 11).



```

root@kali:~# sudo openssl rsa -in llavedeprueba100bits.pem -text -out llavepublicHexa.pub
writing RSA key
root@kali:~#
    
```

Fig. 8. Example of a hexadecimal key. Source: Author's information.

```

Private-Key: (100 bit)
modulus:
    0afc48b5385f57a673422fba49 =>870359746064855796858720598601
publicExponent: 65537 (0x10001)
privateExponent:
    09:19:b3:d5:c4:4c:59:dc:bd:74:b2:1e:81
prime1: 3021521809 (0xb418c391)
prime2: 4212375353 (0xfb13bf39)
exponent1: 1633169041 (0x61582e91)
exponent2: 915059401 (0x368ab2c9)
coefficient: 602854641 (0x23eed4f1)
-----BEGIN PUBLIC KEY-----
MCgwDQYJKoZIhvcNAQEBBQADFWAwFAINCvXItThFv6ZzQi+6SQIDAQAB
-----END PUBLIC KEY-----
    
```

Fig. 9. 100-bit hexadecimal key generated with Open SSL. Source: Author's information.

```

Private-Key: (256 bit)
modulus:
    00b2b980c4233da33323fb7f7a95040d8229593b8f7580897f133e09993b8ea091
    =>8083944246619864058992914082385909967586109685764138081385009245529733767313
publicExponent: 65537 (0x10001)
privateExponent:
    00:8c:eb:fd:df:29:96:61:47:62:b8:dc:84:70:59:
    38:b8:35:b6:ad:f2:29:ef:83:a8:2c:1e:15:12:d8:
    e3:20:01
prime1:
    00:e7:8c:58:db:51:04:31:46:cf:ee:2f:23:64:86:
    4a:91
prime2:
    00:c5:99:20:9f:ff:fe:f0:a8:39:b0:9e:a2:bb:d1:
    f6:01
exponent1:
    09:7e:50:9a:55:5d:05:a4:30:9c:44:64:80:17:9d:
    71
exponent2:
    57:02:fe:1d:d6:c1:b1:c1:b2:5d:b7:0d:5b:fd:b2:
    01
coefficient:
    08:6b:50:11:5e:23:1d:c5:47:54:a3:95:7e:6d:37:
    dd
-----BEGIN PUBLIC KEY-----
MDwwDQYJKoZIhvcNAQEBBQADKwAwKAIhALK5gMQjPaMzI/t/epUEDYIpWtuPdYCJ
fxM+Czk7jqCRAGMBAE=
-----END PUBLIC KEY-----
    
```

Fig. 10. 256-bit hexadecimal key generated with Open SSL. Source: Author's information.

```
Private-Key: (340 bit)
modulus:
0b23057297dab86cd4a965e01854d7f26bed01adf289d2b34aed7ed8f636a8d5ce23d6b33c4a0f6a69f9b3
->1558974590155323540076393624175863694369612223119291374255782520022941558818389212336819091060799699379

publicExponent: 65537 (0x10001)
privateExponent:
01:85:42:95:26:ce:a2:27:99:d1:97:2b:45:a7:2f:
e4:d5:7f:82:9f:31:61:5d:68:46:0b:2f:c9:fb:58:
dd:85:92:93:3b:8c:a8:09:fe:3e:7b:b5:c1
prime1:
03:b4:2d:48:9b:f8:53:33:a2:5d:73:97:0c:71:c9:
92:f0:96:0a:e6:f2:4b
prime2:
03:01:c0:9e:15:d9:af:ab:61:92:b9:4c:ed:1a:fd:
03:73:9d:85:ad:b5:39
exponent1:
01:bb:77:a2:8a:30:6e:d9:ab:8b:01:d1:17:e4:f0:
5e:65:60:07:e1:54:59
exponent2:
00:ff:8a:ef:b0:77:55:4f:83:14:0f:ba:4f:18:df:
98:4e:c0:a3:c9:78:59
coefficient:
01:70:38:c5:ba:ef:4c:bb:de:fb:d2:f9:a2:a2:ac:
09:87:41:56:d6:71:63
-----BEGIN PUBLIC KEY-----
MEYwDQYJKoZIhvcNAQEBBQADAwgIIRCyMFcpfauGzUqXkgGFTX8mvtAa3yidKz
Sul+2PY2qX0I9azPEoPam5swIDAQAB
-----END PUBLIC KEY-----
```

Fig. 11. 340-bit hexadecimal key generated with Open SSL. Source: Author's information

By using the integer factorization program with the Number Field Sieve (NFS) factorization algorithm denominated msieve, proceed to enter module n from the key of public knowledge, which was subjected to the factorization attack to obtain the key with which to encrypt the message that will be transmitted (Fig. 12.)

```
Msieve v. 1.53 (SVN Unversioned directory)
random seeds: a79b8e6d 91c6df8e
factoring 1558974590155323540076393624175863694369612223119291374255782520022941558818389212336819091060799699379 (103 digits)
no P-1/P+1/ECM available, skipping
commencing quadratic sieve (103-digit input)
.
.
.
p52 factor: 1124994410756636672075595149516955324427504416830777
p52 factor: 1385762076014942273883725154208627052395961971569227
elapsed time 05:14:26
```

Fig. 12. Example of factorized key with msieve. Source: Author's information.

The msieve program registers its activity and results in a file called msieve.log, where it is evident that effectively module n was decomposed into two prime factors assumed as p and q and which can be used to obtain the key with which the message is encrypted.

4 Conclusion and Future Work

From the experiments conducted, it may be concluded in the first place that the architecture of a device's processor is quite relevant when carrying out a cryptanalysis process on the RSA algorithm through factorization, given that it influences directly on the time required for this operation but is not an impediment to be carried out. Upon evidencing the device's relevance regarding time, a comparative graph was made showing the degree of performance (Figs. 13, 14, and 15).

It is evident that the NFS encryption algorithm used through msieve behaves much more efficiently as the frequency of the processor increases.

We can also observe the behavior of the RAM memory, which eventually is different in the three devices; in addition, relatively little use is noted in relation to the total memory (Fig 16).

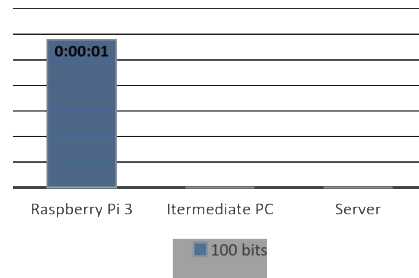


Fig. 13. Graph of Devices vs. Time for 100 bits. Source: Author's information.

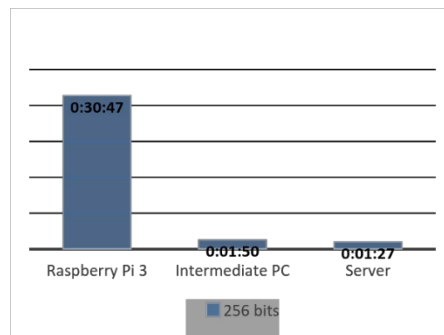


Fig. 14. Graph of Devices vs. Time for 256 bits. Source: Author's information.

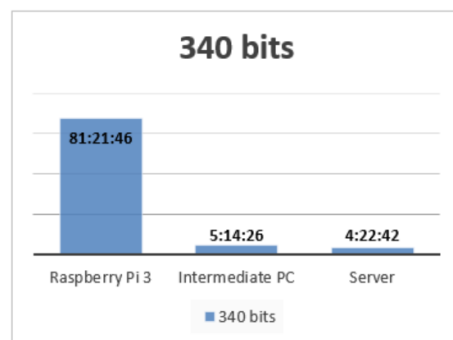


Fig. 15. Graph of Devices vs. Time for 340 bits. Source: Author's information.

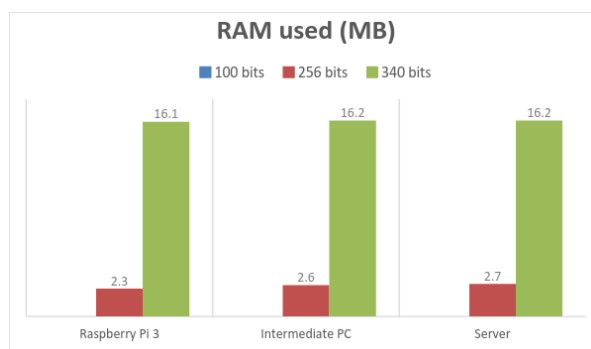


Fig. 16. RAM memory used by the device. Source: Author's information.

Based on the results, we want to provide, in a basic way, the recommendation of the use of SBC devices to perform cryptanalysis to the RSA algorithm, given that they can perform, although less efficiently, mathematical operations with the advantage of their cost in the market.

Future work: Evaluate the behavior of cryptanalysis on the RSA algorithm in a distributed system, where the selected attack is divided by the different nodes that make up the system. Establish a theoretical time according to the numerical complexity of the technique selected and the hardware characteristics of the device chosen to perform the cryptanalysis to, thus, predict the time of the cryptanalysis used with the different RSA key sizes. Compare cryptanalysis times on the RSA algorithm by using CPU and GPU with keys of at least 340 bits.

Acknowledgments: to the Cryptography group and to the GTI Research group of the University of Cauca and to Beta Bit seed of the Research and Development group in Information Technology of the University Institution Colegio Mayor del Cauca, for the support provided for the development of the project.

References

- 1 Zhou, X., Tang, X.: Research and implementation of RSA algorithm for encryption and decryption. In: Proceedings of 2011 6th International Forum on Strategic Technology, vol. 2, pp. 1118–1121 (2011)
- 2 Rivest, R.: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Mag. Commun. ACM* 21(2), pp. 120–126 (1978)
- 3 Cid, C.: Cryptanalysis of RSA: A Survey. SANS Inst. InfoSec Read. Room (2003)
- 4 Genkin, D., Shamir, A., Tromer, E.: RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis. In: Advances in Cryptology - CRYPTO 2014: 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17–21, 2014, Proceedings, Part I, J. A. Garay and R. Gennaro, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 444–461 (2014)
- 5 D. Boneh, Twenty Years of Attacks on the RSA Cryptosystem. *Not. Am. Math. Soc.*, vol. 46, pp. 203–213 (1999)
- 6 Scolnik, H.: Fundamentos matemáticos del método RSA. (2004)
- 7 Kleinjung, T.: Factorization of a 768-Bit RSA Modulus. In: Advances in Cryptology- CRYPTO 2010: 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August

- 15-19, 2010. Proceedings, T. Rabin, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 333–350 (2010)
- 8 Alimoradi, R., Arkian, H.: Integer Factorization Implementation. ICTACT Journal on Communication Technology 7(2), pp. 1310-1314 (2016)