Journal of Theoretical and Applied Information Technology
31st January 2020. Vol.98. No 02

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

UNDERSTANDING ENERGY PROFILE FOR SOFTWARE APPLICATION

[1]AREEJ A. AHMED, [2]SHAWKAT K. GUIRGUIS, [3]MAGDA M. MADBOLY

[1,2,3] Department of Information Technology, Institute of Graduate Studies and Researches, Alexandria University, Alexandria, Egypt

[1] Research and Development Center, Ministry of Electricity, Baghdad, Iraq

[1]areej_abdulmunem@yahoo.com , [2] shawkat_g@yahoo.com, [3]mmadbouly@hotmail.com

ABSTRACT

The expansion of energy-aware software is considered major necessity for a lot of computers and smart phone devices. Coinciding with the high importance of computers, electricity cost increasing, Energy measurement techniques have been evolved for measuring and enhancing the energy consumption of software layers. In spite of this fact, these techniques can't determine enough energy information. And they can't locate how and where the energy is wasted within software. Therefore, a framework is proposed to offer valuable imagination for programmers to write energy efficient code and software. And to creates a comparison study in energy evolution model of software applications to gain opportunities for energy minimizing.

Keywords: *AES, software methods, Energy utilization, Energy growth, Jalen approach, key length.*

1. INTRODUCTION

In the last time, energy aware software is widespread, coinciding with using portable electronic devices such as smart phones, laptops, and tablets. In these devices energy is becoming an essential issue. Therefore, energy saving is considered sustainable development goal [1]. Recently, scientific research confirmed that energy is consumed in information technology systems by both hardware and software components [2]. Software consume energy indirectly, it is influencing on the energy consumption of the hardware [3]. Therefore, Software has a high weight in consuming energy comparing with hardware [4].

Therefore, minimizing the energy consumption of software application is an essential necessity and a technological challenge [5]. Moreover, program developers lack for deep knowledge of energy consumption. So, energy microscope at the application level is needed to diagnose the hungry parts [6] [7]; to improve new methods consuming lower energy; and to balance performance vs. energy consumption towards building energy efficient applications [8].

To address energy aware shortcomings, we propose in this paper a framework for presenting the energy models of some encryption and decryption algorithms by using the most remarkable approach; and comparing energy evolution models of applications. The other sections are introduced as follows. Section 2, we offered motivations keys and aims that lead to this work. In section 3, the framework and methodology is described and system implementation steps are illustrated. Section 4, outlines the results from the experimental evaluation to ascertain and assess the framework that proposed in section 3. The analysis and discussion of these results are in section 5. Finally, the conclusion and future work in Section 6.

2. MOTIVATIONS KEYS & AIMS

Energy represents a critical matter in the handled electronic private devices especially laptops and mobile phones. Therefore, researchers' attention is directed to measure energy consumption by preparing different approaches. However, these approaches have many advantages and some limitations or drawbacks. This paper contribution is therefore presenting measurement approaches and tools in order to make a comparison among them, and then facilitates choosing the most efficient approaches and tools.

Energy measurement is classified as following:

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

- Hardware based measurement [9]: Hardware based measurement is represented the most reliable methodology. But it has some restrictions such as: it suffers from noise, has a portion of power loss during measurement process, and it demands a certain types of battery.
- Software based measurement (internal profiling) [10]: When we taking about the internal (profiling) software based methodology, the external hardware is not demanded. However, energy profiling software has many determinations: it generates additional energy consumption caused by energy logging application, and has some of installation troubles.
- And software based measurement (external profiling) [11]. Which limited by tasks supplied by OS API. And it has several data communication problems.

Therefore, in this paper a comparison is presented among measurement methodologies as shown below in table 2.1.

Table 2.1: Energy measurement methodologies

Hardware based	Software based internal profiling	Software based external profiling
Accurate measuring	Only estimation	Very accurate
Measure energy for whole system	For internal components	For internal components
Not expensive	Free	Expensive
Easy to use	Easy to use	Hard to use
Required additional hardware	Not required	Required
Obtainable everywhere	Online download	Hard to find
Hard analysis	deep analysis	Very deep analysis

PowerScope [12] represent the best example about hardware based measurement tool which is offering energy usages of applications by a digital multimeter and a distinct computer. This tool is most reliable and it has three basic components: system monitor, energy analyzer and energy monitor.

Several approaches supply fine-grained energy measurement, like eLens [13] which is a peripheral approach that combines the both concepts together: per-instruction energy evolution model to detect the energy consumption for each method inside application, and the program analysis to specify paths of energy information, eLens is characterized by accuracy, rapidity, and there isn't a necessity to vary the mobile operating system or to require the power meter.

e-Surgeon [14] is a software approach used for assigning high consuming energy points of Application Servers. This approach includes two levels of energy measurement: Operating system level and process level. The first level is used for measuring the energy usage of coarse grained level (such as energy consumption of processes used by different hardware devices CPU, network, memory), and the latter is used for measuring the energy consumption at fine grained level (classes and methods).

Application profiling tools are so valuable but it is restricted by programming language such as ANTS profiling for .Net language [15].

The internal profiling software based measurement use the external hardware [16][10]. this is clearly occurs when we use eprof [17] to estimate energy utilization at code level; although Eprof is helpful approach but it has some restrictions so it requires a fundamental changes to kernel, thus limiting the flexibility of the approach.

Other software tool is Joulemeter [18] that compute the energy consumption of both hardware (virtual machines, servers, desktops, and laptops) and software applications running on a computer.it can measure energy usage of computer resources, like CPU utilization and screen brightness, But its model can't estimate without laboratory benchmarks this makes it low flexibility [19].

PowerTop[20] is a Linux tool which diagnoses software component that consume more

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

energy. PowerTOP has advanced interface to present and estimate the energy profile; allowing developers determining which processes taking more energy [21]. PowerTop share similarities with pTop tool [22] in profiling at a process-level. pTop is extremely accurate and doesn't require extra hardware, but it has little overhead. pTop provides energy-aware APIs in mobile devices and cloud computing. Another version exist called pTopW [23].

Energy Checker [24][11] is a software tool that estimates the power consumption of the application by using counters. Therefore, it needs a hardware powermeter, so it is hard to deduct the energy consumption of software.

Also other software based approaches are PowerAPI [14][25] and Jalen [26] , Both tools employs power models for measuring the energy utilization of software. PowerAPI evaluates in a real-time the energy consumption of process for variant hardware resources, by submitting an API. Jalen is building to offer the energy consumption of applications at the code level (for processes and even for methods).

In this paper a comparison study of energy measurement and estimation approaches is presented as followed in table 2.2

Table 2.2: Comparative study of energy measurement approaches of software

Issue	PowerScope	pTop	PowerAPI	Jalen	PowerTop	Energy Checker	Joulemeter	Application profiling
Hardwar dependency	Dependent.	Independent.	Independent.	Independent.	Independent.	Dependent.	Independent.	Independent.
Accuracy	More accurate than energy estimation (Reliable).	Offers energy information in real time.	Precise enough to displace hardware power meters.	Most accurate by profiling energy hotspots.	Maximum accuracy with running in calibration mode.	Accuracy depends on hardware power meter.	Useful accuracy about 20%.	By determining energy leakages in applications.
Suitable for monitoring	Process, procedure within process.	Process.	Process.	Methods, classes.	Application, Process.	Application.	Process.	Software, Application.
overhead	Has no profiling overhead (offline profiling)	Not negligible (3% of the CPU and 0.15% of memory).	Depends on accuracy, tools' cost, and the applications.	Null to the application, but for the agent is around 3%.	Relatively small overhead.	Has no overhead.	Has additional Overhead of application training.	Low overhead 2.7 %.
Observed resources	Current, program counter.	Hardware resources (CPU, Network, Disk).	Hardware resources.	Software resources.	Hardware resources.	Hardware resources, application counter.	Hardware resources.	Software resources, parameters.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

In spite of the previous approaches is measuring energy consumption of software application accurately. But they lack of energy measurement approaches depend on context information and their finite granularity. They have restricted understanding of how energy is consumed by software. A lot of researches are performed to conclude the internal function of software, but they failed in providing enough energy information. Others can't diagnose energy leaks; Also they can't diagnose which process or method is consumed more energy than is necessary. Therefore the core concern of the research is:

- Detecting how and where energy is consumed at the code level. It is a great necessity to assign the hungry process or method within software for improving energy efficient software.
- Also offer energy evolution model and knowing the effect of changing input parameter on energy consuming scenario. This is considered a fundamental requirement to give guidelines for software programmers to choose the best energy efficient method for their software, and finally to implement the green coding principles.

3. PROPOSED METHODOLOGY

3.1 The Framework Methodology

It's predictable that energy orientation of methods is changing when its parameters are changed. Knowing this effect is very valuable for application developers to choose optimal parameter value [27]. So; this framework is proposed to infer the energy evolution model of software code depending on the variability of their input parameters.

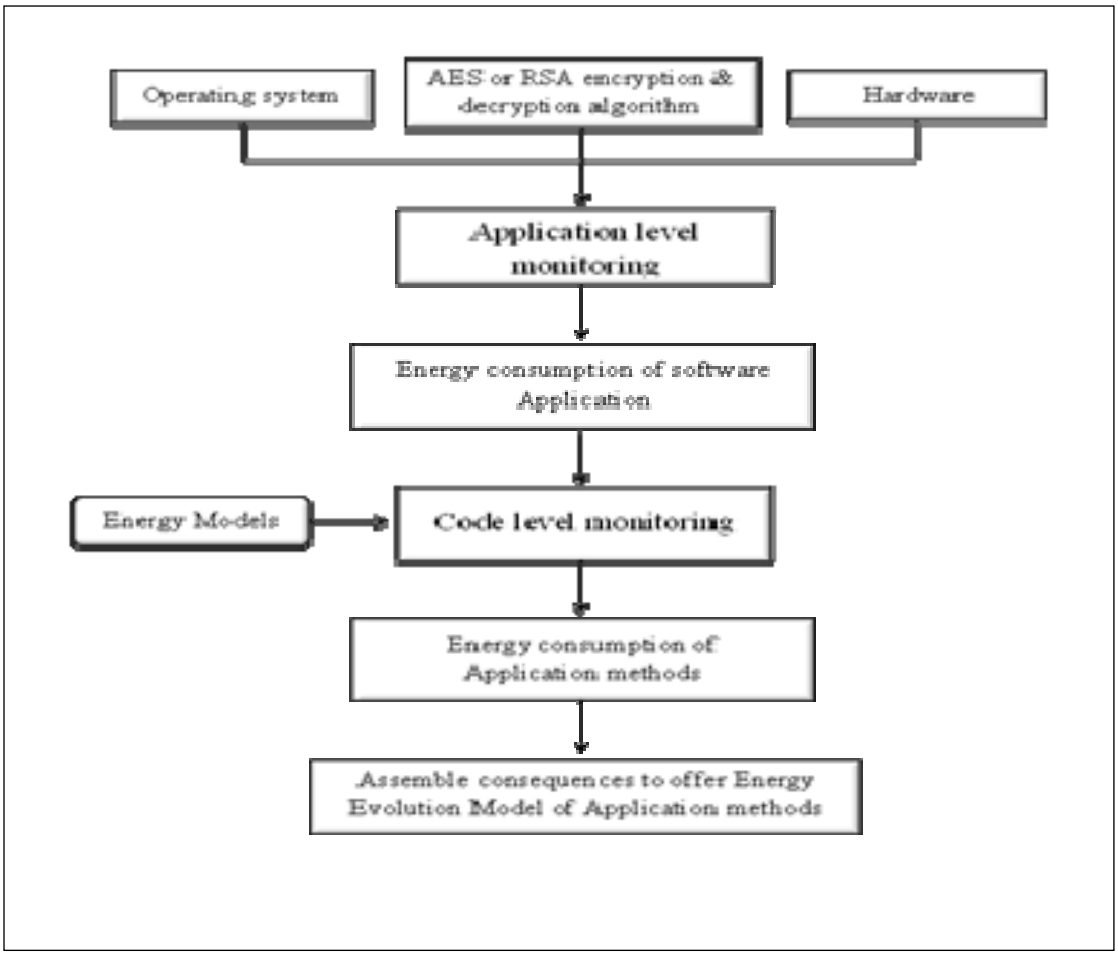

Figure 3.1: Framework methodology.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

To illustrate this framework, methodology is demonstrated in Figure 3.1 as follows: at first gathering CPU utilization of the executing method in an application. These data are collected through an intermediary such as a virtual machine after injecting the jar file for measurement approach as a Java agent during the start of the program. Then the total energy consumption of the application (in this framework an AES) is collecting by CPU. This information is acquired using application level measurement tool (Power API). Later, data which gathered previously is used with code level energy models to measure the energy consumption of application methods by hardware resource. This is done by using statistical sampling version of jalen (STS) measurement approach [28] because it doesn't require any changes on the application code and it has a little overhead. Finally, computing and correlating all consequences after many execution cycles to present energy evolution model of application methods through a virtual machine while varying its input parameters.

3.2 Energy Measurement & Energy Models for CPU

Energy models for CPU are discussed in [26] is used as illustrated in formula (1):

$$Power^{CPU}_{method} = Power^{CPU}_{software} \times Utilization^{CPU}_{method} \quad (1)$$

Then $Power^{CPU}_{software}$ is gained from formula (2)

$$P_{CPU} = \frac{\sum_{frequencies} P^{f}_{CPU} \times t^{f}_{CPU}}{\sum_{frequencies} t^{f}_{CPU}} \quad (2)$$

In Java programming language the application code is mostly executed within threads, so at first the power consumption for threads is computed then for methods in the last observing cycle as in formula (3):

$$Power^{CPU}_{thread} = \frac{Power^{CPU}_{software} \times Utilization^{CPU}_{thread}}{Duration_{cycle}} \quad (3)$$

$Power^{CPU}_{software}$ represents the power consumed, $Utilization^{CPU}_{thread}$ is the CPU time of the thread , and $Duration_{cycle}$ is the duration of the monitoring cycle. Later CPU utilization for methods is computed in formula (4):

$$Utilization^{CPU}_{method} = \frac{Duration_{method} \times Utilization^{CPU}_{thread}}{\sum_{m \in Methods} Duration_{m}} \quad (4)$$

Also in the last observing cycle, $Duration_{method}$ is representing the execution time of method, and $Duration_{method}$ is representing the execution time of all methods. Finally, the power consumption of methods of formula (1) is computed by using formula (5):

$$Power^{CPU}_{method} = \frac{Utilization^{CPU}_{method} \times Power^{CPU}_{thread}}{Duration_{cycle}} \quad (5)$$

In this framework, AES is used as the most significant standard for symmetric algorithm because AES provides strong encryption and has been selected by National Institute of Standards and Technology (NIST) as Federal Information Processing Standard in 2001, and in 2003 the United States Government announced that AES is secure enough to protect classified information up to the top secret level.

To display the effect of modifying the key length as input parameters (that specifies the number of repetitions of transformation rounds that convert the plaintext to the ciphertext); an AES symmetric encryption/ decryption algorithm is constructed with ability creation variant key length (1024 and its multiples) to obtain method's energy directions.

In more details, first the algorithm is run with creating specific key length, and encrypting/ decrypting in many times a large text [29]. Then it's the time Jalen working (which is a Java observing agent to compute the method's energy consumption of an algorithm. Energy values are saved in csv file. Then this process is running but with other input parameter to graph energy evolution model finally towards detecting methods which are hungry for energy.

3.3 Proposed System Implementation &Graphical Design

The implementation of the proposed system, which comes with better understanding for energy profile, is as follows: at first running the system associated parts and then moving to a question that specifies whether to select the application which is wanted to monitor its energy, or to display a previous energy profile: this step is determined manually by the user.

CMD window is run for monitoring the application, and an initialization the energy profile procedure starts to begin calculating energy consumption of an application by using Jalen. This depends on a specification to the key length and the

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

loop quantity (the number of times the encryption and decryption of specific text). The previous step is also determined manually by the user. Next, the key generation process is done to create the key which is used to encrypt a specific text. Then retrieve the original text by using the key that generated previously. Later, energy profile is stopping and followed by saving results in CSV file which display the energy profile finally.

Proposed system used three Graphical User Interfaces (GUIs) as follows: First UI is named (Jalin) and made by Microsoft.net framework 4.5. Windows form "help in opening CMD window, running commands, displaying a list of energy information, and at the end drawing the energy chart".

Second UI is CMD that used for monitoring energy consumption of an application (AES). Third and last UI: is made by Java programming language and consists of three buttons determining the applications' main steps (key generation, encryption, and decryption).

4. EXPERIMENTATIONS

This section reports the results from the experimental evaluation including several experimental results performed by concluding the energy directions of an AES and RSA algorithms to ascertain and assess the framework that proposed in chapter 3 as follows:

4.1 Research Material

The proposed framework described in section3consists of the following applications: The AES and RSA encryption and decryption applications that are built in Java Script Programming Language version 7 and 8, Maven version 3 is used to obtain Jalen measurement, and finally Graphical User Interface made by Microsoft.net framework 4.5. Windows form that used for opening CMD, running commands and at the end drawing the energy chart. The energy consumption of methods in AES algorithm is measured using files downloading from https://github.com/adelnoureddine/jalen (the statistical version of Jalen).

Then framework is tested and validated in experiments by using files downloading from the website http://www.arvindguptatoys.com/arvindgupta/cbt14-Short%20Stories%20For%20Children.pdf. The framework is implemented with a laptop: Dell inspiron n 5010 with the following specification: Intel® core ™ i3 CPU M350 at 2.27 GHz processor with 4.00 GB Installed memory (RAM) DDR3 SDRAM PC3-10600, 666.7 MHz, this machine equipped with operating system Windows ® version 8.1 Professional Edition, and system type: 64 bit operating system (x64- based processor).

. These instructions give guidance on layout, style, illustrations and references and serve as a model for authors to emulate. Please follow these specifications closely as papers which do not meet the standards laid down, will not be published.

4.1.1 Evaluation metrics

- **Objective:** the main objective of this work is to offer an energy profile for software applications while varying their input parameters and concluding the energy directions of an AES and RSA algorithms.
- **Performance description:** performance description of this work with the change which we will talk about in comparison method 4.1.2 is very necessary.
- **Target:** creating framework to reach our objectives, and to provide guidelines for software developers to help them implement green coding principles.

4.1.2 Comparison Method

JALEN UNIT profiler application presented by the University of Lille Ph.D. students in France [30] which is used to present energy evolution model of application methods. While JALEN UNIT profiler uses Jalen for monitoring energy consumption of applications at the code level by using Intel Core 2 Duo 6600 processor at 2.40 GHz and running Lubuntu Linux 13.04 64 bits operating system. This allows to see the impact of changing both the hardware component and the operating system on power efficiency of an RSA algorithm based on Hardware components consume energy.

Jalen associates the energy consumption of hardware to the software code that initiated the task for hardware components. Therefore, energy consumption is highly dependent on hardware components.

4.2 Experimental Results

In this section, the framework is tested with many experiments to evaluate the amount of energy consumed of an AES algorithm in each experiment, and then made to prove our search point by results. In the first experiment, encryption and decryption is done for 50 times So that to obtain enough

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

execution evidence to measure energy utilization, but in the second, the encryption and decryption is done for 100 times.

For each one there is consideration procedure as follow:

- At the beginning, the AES algorithm is run with creating specific key length 1024
- And encrypting/ decrypting the given text.
- At this point Jalen observes and computes the energy consumption for the whole AES via requisition Power API; and for the methods of an AES especially. In addition to CPU time.
- Energy values are saved in csv file.

Then the former procedure is rerunning but with double key length (2048, 3072, 4096, 5120, and 6144). At last all outcomes get together to graph energy evolution model finally towards detecting methods which are hungry for energy.

In the previous framework, the procedure is performed for ten times at each key length; and then extracts the average energy utilization of the all application. Furthermore, is performed for the run time.

Table 4.1: Energy profile of whole AES in variant key length.

Application Key length	Run time (m)	Average energy in Joule	No. of encryption & decryption (times)
AES 1024	NULL	NULL	50
AES 2048	NULL	NULL	50
AES 3072	10:15	1391.4494	50
AES 4096	13:11	1805.333	50
AES 5120	19:29	2185.716	50
AES 6144	22:03	2346.280	50
AES 1024	8:17	812.655	100
AES 2048	14:25	1674.933	100
AES 3072	15:37	1972.112	100
AES 4096	18:58	2418.530	100
AES 5120	22:46	2591.085	100
AES 6144	28:57	2886.285	100

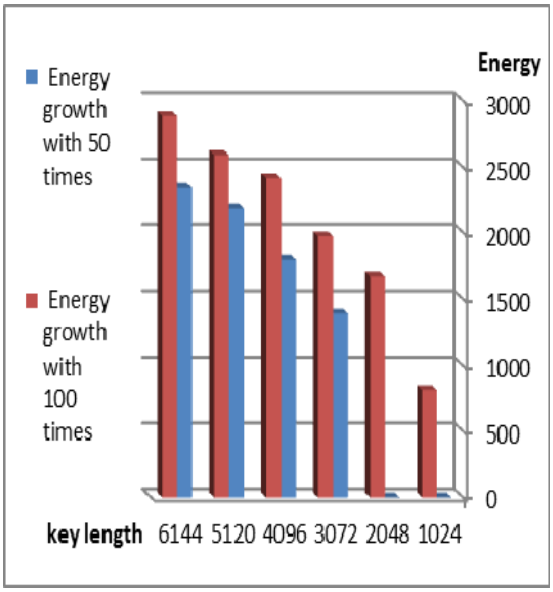

Figure 4.1: Energy growth of whole application.

Therefore, the STS version of Jalen is appointed to indicate which method is responsible about this energy progress. Results, in figures 4.2 and 4.3 infer that there are two hungry methods for energy in AES algorithm:

- AES.AddRoundKey, which is the basic element in AES energy consumption.
- And AES.SubBytes.

The energy values of the rest methods are too light in comparing with the above methods. So, these outcomes underline the requirement for deep measurement at the code level.

There are many values of energy which are not obtainable as offered in tables 4.1, 4.2, and 4.3; this is happened when we generate AES's key with length 1024, and 2048 in 50 times encryption and decryption.

These insensible values belong to many considerations:

- The operating system is highly advancement in compared with other operating systems.
- The speed of the processor makes Jalen doesn't feel or recognize these values.
- Besides to the above, AES is considered very lightweight algorithm.

Journal of Theoretical and Applied Information Technology
31st January 2020. Vol.98. No 02

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 4.2: Energy profile of AES methods with variant key length in 50 times.

Application Key length	Run time (m)	Energy of Add Round Key in Joule	Energy of Sub Bytes in Joule
AES 1024	8:17	862.694	68.740
AES 2048	14:25	1218.576	287.805
AES 3072	15:37	1533.140	358.128
AES 4096	18:58	1730.216	415.852
AES 5120	22:46	1814.038	531.009
AES 6144	28:57	2223.145	652.353

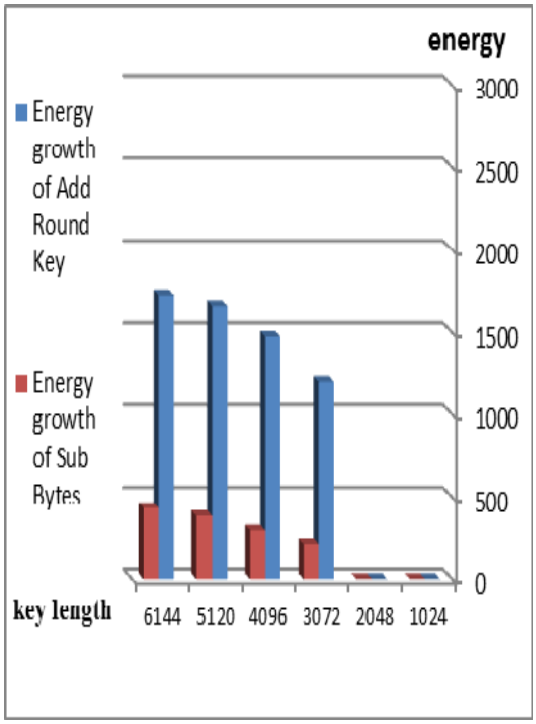

Figure 4.2: Energy growth of application methods in 50 times.

Table 4.3: Energy profile of AES methods with variant key length in 100 times.

Application Key length	Run time (m)	Energy of Add Round Key in Joule	Energy of Sub Bytes in Joule
AES 1024	NULL	NULL	NULL
AES 2048	NULL	NULL	NULL
AES 3072	10:15	1201.804	218.184
AES 4096	13:11	1479.226	298.364
AES 5120	19:29	1655.512	385.395
AES 6144	22:03	1729.222	434.324

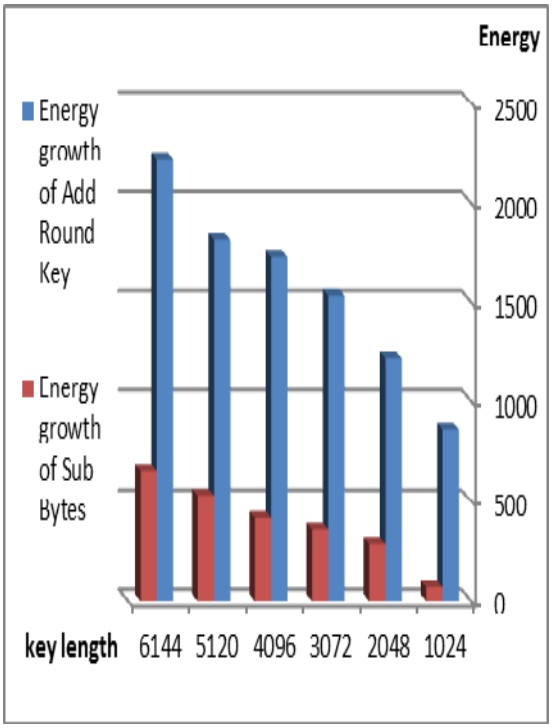

Figure 4.3: Energy growth of application methods in 100 times.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

AES also doesn't require extra energy. This fact is corroborated when we tried to encrypt and decrypt a small text in 50, 100, and even 100000 times; energy value was also approximately null or not obtainable.

We also test our framework with other software application (RSA encryption / decreption application).

The energy profiler application used to infer energy growth of both AES and RSA algorithms at the code level while varying its input parameters. This is done by using Jalen with Intel® core ™ i3 CPU processor; and Windows operating system (as offered in section 4.1).

In our framework strategy figure 4.4 shows energy profile of RSA algorithm, and illustrates the exponential growth when the application is run with a variant key length.

The same strategy which done with AES, is executed here with RSA; but with more multiples of key length.

Table 4.4: Energy Profile Of Whole RSA In Variant Key Length.

Application key length	Run time(m)	Average energy in Joule
RSA 1024	NULL	NULL
RSA 2048	NULL	NULL
RSA 3072	NULL	NULL
RSA 4096	NULL	NULL
RSA 5120	8:49	1218.691
RSA 6144	9:32	1365.772
RSA 7168	10:44	1682.225
RSA 8192	16:58	2943.006
RSA 9216	21:18	3680.821

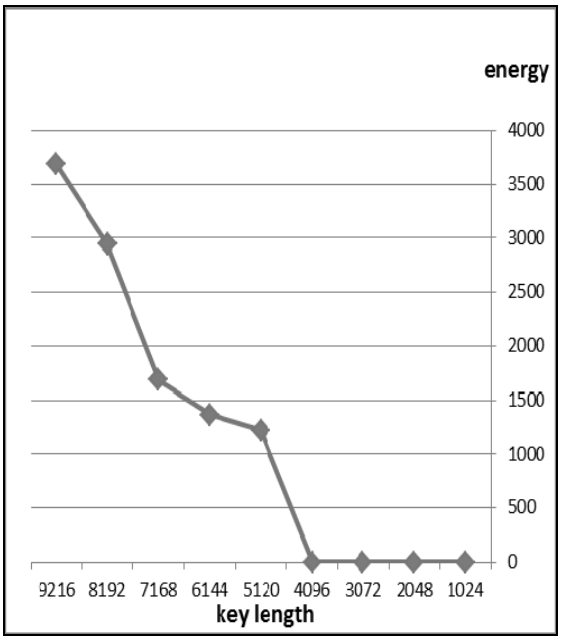

Figure 4.4: Energy Growth Of Whole Application (RSA) In Our Framework.

In more details, at first we run RSA with 1024, 2048, 3072 and 4096. But we discovered that there are many values of energy which are not obtainable as offered in table 4.4; this is happened when we generate RSA's key with length 1024, and 2048 in 100 times encryption and decryption. Our explanation for these insensible values is:

- The operating system is highly advancement in compared with other operating systems.
- The speed of the processor makes Jalen doesn't feel or recognize these values.

But we note that there is an obvious growth in energy consumption. This is happened when we run RSA with 5120 key length. Then energy expansion is continuing with 6144, 7168, 8192, and finally 9216 key length.

4.3 Comparison with Previous Experiments

In this framework, performance description of Jalen with different hardware and operating system is very valuable. Therefore, experiments results in the second scenario with RSA show that there are many values of energy consumption which are not available and Jalen

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

doesn't sense them, as shown in table 4.4 and described in the previous section (4.2). This signifies that energy consumption value of RSA is clearly born with 5120 key length and more; this is through 100 times of encryption and decryption.

While in the French study [30], energy consumption value of RSA methods begin since 1024 key length is generated. And it is still increasing in consumption. we noticed that the energy consumption of RSA become obvious at the key length 3000 bit and more obtained through 10 times of encryption and decryption, and by using Intel Core 2 Duo 6600 processor and Linux operating system as shown in figure 4.5.

This variant explain the impact of changing both the hardware component and the operating system on power efficiency of an RSA algorithm based on Hardware components consume energy.

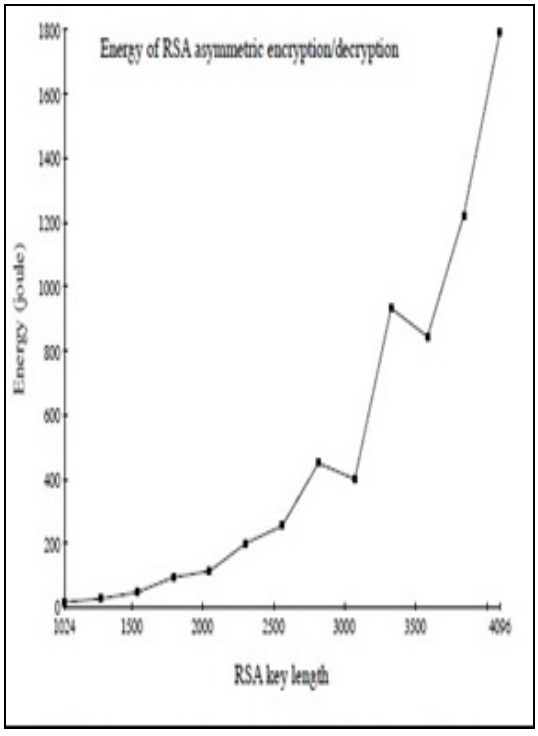

Figure 4.5: Energy Growth Of Whole Application (RSA)[30].

Although an energy consumption of RSA algorithm is increased in this work due to the different hardware that machines use thus consuming different amount of energy, also due to an alternative operating system, and the number of traces of encryption and decryption; but still keeping similar energy trends and distribution in RSA methods.

This proved that running software on different machines does not necessarily change the energy distribution between their software methods. Thus, proposing an empirical model of the evolution of energy consumption of software code is relevant, and can be used by the developers.

5. DISCUSSIONS

From our framework and experiments we obtained many essential principles:

- Presenting a comparative study among measurement approaches and tools is very important to minimize energy consumption of software by choosing the most efficient approaches and tools.
- Inferring the energy profile of software code depending on the variability of their input parameters is very valuable for application developers to choose optimal input parameter value.
- When the same framework is executed on a variant hardware, the values of energy will vary too. This belongs to the hardware nature. (The best example about this case is AES.AddRoundKey method).
- Observing energy consumptions at the code level is considered as a first class necessity to expose energy distribution. This knowledge provides deep understanding for programmers in order to choose alternative methods which require lower energy.
- A software approach (STS Jalen) is employed for deeper energy estimating.
- Studying the impact of changing both the hardware component and the operating system is very valuable. This will allow knowing the power efficiency of software application based on Hardware components consume energy.
- Furthermore, monitoring energy progress at the application level is not enough. So, we need for more analysis to understand the fundamental reasons for this energy evolution.
- Our work is useful to use in software industry especially in software designing companies by replacing and choosing optimal input parameter value which making the application consuming less energy (i.e key length in AES encrypting/ decrypting algorithm). Also making programmers choose alternative methods which require

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

less energy (choosing methods instead of AES.AddRoundKey, and AES.SubBytes methods in AES encrypting/ decrypting algorithm).

5.1 Research Pros and Cons

Pros:

- This work gives us deep layer of knowledge of the energy usage and distribution within software.
- Offers the validity of the framework by detecting the hungry methods which leading to the exponential growth of AES algorithms.
- Preparing guidelines for software programmers to assist them choosing the most efficient method.
- Proposed system benefits are increased with the using of AES encryption and decryption algorithm which is very secret and robust algorithm, in addition of using buy a large number of people.

Cons:

- This framework observes methods individually without noticing the interrelation between methods. This is due to many requirements such as a large time, and the professional team who have a deep knowledge in software engineering.
- Mathematical analysis of this work is still manual; therefore, using analysis techniques is an essential requirement, such as Principal Component Analysis (PCA). This could be helpful to knowing the effect of changing the input parameter on the energy usage of the method.

6. CONCLUSION & FUTURE WORK

Learning and limitations are gained from our framework such as:

- The existing framework in this paper illuminates and maps the energy evolution trend of application methods depending on its parameters. But, it doesn't define the indirect impact on other methods.
- Therefore, studying this impact is very necessary. Also we need to present energy profile depending on other input parameter. Thus, leads for clear description on energy utilization profile. Significantly, leads to conclusions that might help a developer to choose efficient software code.
- The current work depends on CPU energy only. So, our future work is a framework which depends on disk energy in presenting energy evolution models for software methods.

REFERENCES:

[1] A. Noureddine, A. Bourdon, R. Rouvoy, and L. Seinturier, "Monitoring Energy Hotspots in Software", *In Automated Software Engineering Journal (ASEJ)*, ISSN: 0928-8910 (print version) 1573-7535 (electronic version), No. 10515, Feb. 2015.

[2] G. Kalaitzoglou, M. Bruntink, and J. Visser, "A Practical Model for Evaluating the Energy Efficiency of Software Applications", *2nd International Conf. on ICT for Sustainability (ICT4S 2014)*, Stockholm, Aug. 24-27, 2014, pp. 77-86.

[3] L. Ardito, "Energy Aware Self- Adaptation in Mobile Systems", *In Proceedings of the International Conference on Software Engineering (ICSE)*, San Francisco, CA, USA ,2013, pp. 1435-1437.

[4] A. Noureddine, A. Bourdon, R. Rouvoy, and L. Seinturier, "A preliminary study of the impact of software engineering on green IT", *In First International Workshop on Green and Sustainable Software (GREENS)*, Jun. 2012, pp. 21–27.

[5] A. Trefethen, and J. Thiyagalingam, "Energy-aware software: Challenges, opportunities and strategies", *Journal of Computational Science*, Vol. 4, Issue.6, Nov. 2013, pp. 444-449.

[6] T. Hönig, C. Eibel, W. Schröder-Preikschat, B. Cassens, and R. Kapitza, "Proactive Energy-Aware System Software Design with SEEP", *In Proceedings of the 2nd Workshop on Energy Aware Software-Engineering and Development, (*Oldenburg, Germany, 25 Apr, 2013, pp. 6-7.

[7] G. Costa, and H. Hlavacs, "Methodology of measurement for energy consumption of applications", *11th IEEE/ACM International Conf. on Grid Computing (GRID), Brussels*, Oct. 25-28, 2010, pp. 290-297.

[8] G. Calandrini, A. Gardel, I. Bravo, P. Revenga, J. Lázaro, and F. Toledo-Moreo, "Power Measurement Methods for Energy Efficient Applications ", *International journal of sensors*, Vol. 13, Issue. 6, 18 June 2013, pp. 7786-7796.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[9] N. Thiagarajan, G. Aggarwal, A. Nicoara, D. Boneh, and J. Singh, "Who Killed My Battery: Analyzing Mobile Browser Energy Consumption", *ACM International Conference on Mobile Web Performance*, Lyon, France, Apr. 16–20, 2012, pp. 41-50.

[10] A. Pathak, Y. Charlie Hu, and M. Zhang, " Where is the energy spent inside my app? Fine Grained Energy Accounting on Smartphones with Eprof", *proceedings of the ACM Eurosys workshop*, 2012, pp. 29-42.

[11] A. Noureddine, R. Rouvoy, and L. Seinturier, "A Review of Energy Measurement Approaches", *ACM SIGOPS Operating Systems Review 47*, Mar 2013, pp. 42-49.

[12] G. Procaccianti, A. Vetro', L. Ardito, and M. Morisio, "Profiling power consumption on desktop computer systems", *In Proc. of the First international conf. ACM on Information and communication on technology for the fight against global warming,Verlag*, Berlin, 30 Aug 2011, pp. 110-123.

[13] S. Hao, D. Li, W. Halfond, and R. Govindan, "Estimating mobile application energy consumption using program analysis", *35th International Conf. on Software Engineering (ICSE)*, Press Piscataway, NJ, USA ,18-26 May 2013, pp. 92-101.

[14] A. Noureddine, A. Bourdon, R. Rouvoy, and L. Seinturier, "e-Surgeon: Diagnosing Energy Leaks of Application Servers", *INRIA Research Report, Project-Teams ADAM*, Jan. 2012, pp. 1-27.

[15] J. Vratislav, " *Performance Profiling for .NET Platform*", Master's diss. of Computer Science and Engineering, Czech Technical University, Faculty of Electrical Engineering, Prague, January 2012.

[16] M. Kambadur, and M. Kim, "Energy Exchanges: Internal Power Oversight for Applications", *CUCS Technical Report (cucs-009-14)*, Columbia University, New York, USA, Mar. 2014, pp. 1-11.

[17] S. Schubert, D. Kosti´c, W. Zwaenepoel, and K. Shin, "Profiling Software for Energy Consumption", *IEEE International Conf. on Green Computing and Communications (GreenCom)*, France, 2012, pp. 1-8.

[18] A. Kansal and F. Zhao, "Fine-grained energy profiling for power-aware application design", *Proceedings of the 1st Workshop on Hot Topics in Measurement and Modeling of Computer Systems at ACM Sigmetrics (HotMetrics'08)*, June 2008, pp. 26–31.

[19] H. Hassan, and A. S. Moussa, " Power Aware Computing Survey", *International Journal of Computer Applications (0975 – 8887)*, Vol. 90 , No.3 , March 2014, pp. 21-26.

[20] A. Faria, L. Aguiar, D. Lara, and A. Loureiro, "Comparative Analyses of Power Consumption in Arithmetic Algorithms Implementation", *IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, Changsha, Nov. 16-18, 2011, pp. 1247-1254.

[21] Intel, "LessWatts.org - Saving Power on Intel systems with Linux", http://www.lesswatts.org, 2011.

[22] T. Do, S. Rawshdeh, and W. Shi, "pTop: A Process-level Power Profiling Tool", *In Hot Power'09: Proceedings of the 2nd Workshop on Power Aware Computing and Systems*, Big Sky, MT, US, Oct. 2009.

[23] H. Chen, Y. Li and W. Shi, "Fine-Grained Power Management Using Process-level Profiling", *research gate article*, Jan. 2012.

[24] S. Zhang, "Adding Intel® Energy Checker SDK Instrumentation to Your Application", *Intel developer zone*, Santa Clara, USA, Jul. 2012.

[25] A. Noureddine, A. Bourdon, R. Rouvoy, and L. Seinturier, "Powerapi: A software library to monitor the energy consumed at the process leve"l, *ERCIM News journal*, (92), Jan. 2013, pp. 43-44.

[26] A. Noureddine, A. Bourdon, R. Rouvoy, and L. Seinturier, "Runtime Monitoring of Software Energy Hotspots", *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering(ASE'12)*, Essen, Germany, Sep. 2012, pp. 160-169.

[27] A. Ahmed, S. Guirguis, M. Madboly, "Energy Evolution Model for AES and RSA Algorithms", *International Journal of Engineering & Science Research (IJESR)*, Vol. 5, Issue. 7, July 2015, pp. 625-633.

[28] https://www.github.com/adelnoureddine/jalen.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[29] http://www.arvindguptatoys.com/arvindgupta/cbt14Short%20Stories%20For%20Children.pdf

[30] A. Noureddine, A. Bourdon, R. Rouvoy, and L. Seinturier, "Unit Testing of Energy Consumption of Software Libraries", *In Software Engineering Aspects of Green Computing track of the 29th Annual ACM Conference Symposium on Applied Computing (SAC'14)*, Gyeongju, South Korea ,Mar. 2014, pp.1200-1205.