Energy Awareness and Secure Communication Protocols: The Era of Green Cybersecurity

Marco Castaldo[1], Aniello Castiglione[2], Barbara Masucci[1], Michele Nappi[1], and Chiara Pero[1](✉)

[1] Department of Computer Science, University of Salerno, Fisciano, Italy
m.castaldo92@gmail.com, {bmasucci,mnappi,cpero}@unisa.it
[2] Department of Science and Technology, University of Naples Parthenope, Naples, Italy
castiglione@ieee.org, castiglione@acm.org

Abstract. The computational effort required to guarantee the security of a communication, due to the complexity of the cryptographic algorithms, heavily influences the energy consumption and consequently the energy demand of the involved parties. This energy request makes secure communication with low energy consumption a non-trivial issue. The aim of this work is to study, as well as evaluate, the way in which the cryptographic primitives used in secure communication protocols affect the workload of the CPU and, therefore, the energy expenditure of the interacting devices. Through the aforementioned analysis, attention will be focused on the need to consider with greater sensitivity the possibility of operating/undergoing cyber-attacks using the power consumption induced by secure communications. The main focus is to exaggerate the workload of the target devices in order to produce the maximum energy consumption and have a kind of Denial-of-Service attack. The paper studies the contribution of energy consumption introduced by the different part of "secure" primitives within the TLS protocol. As a conclusion, it is shown how Cryptography is often used not in the proper way, i.e., it may introduce costs that are sometimes higher than the value of the "goods" to protect.

Keywords: Energy · Green computing · Cybersecurity · Network security · Applied cryptography · TLS protocol

1 Introduction

The right compromise between security and performance it is always known to depend on the implementations. After the recent EU regulations and the always more restrictive security policies adopted by the leading companies operating on the Internet it is essential and unavoidable to adopt policies, protection methods as well as encryption of data and communications. The main focus of the paper is to study and measure how cryptographic primitives impact on the CPU usage, and consequently on the power consumption of the involved devices. There is the need to consider new kind of attacks that (are both distributed and local) lead to resource starvation. Such kind of attacks aim at exploiting the consumption of some hardware resources of the victim host that, for example, is the CPU time, the memory amount, the free space on disk and many others. Those attacks are characterized for being "no noise", hence not perceivable by the traditional IDS but only by custom agents located close the single machines. A specific implantation of such attacks is the one that see as the main objective the energy consumption of the target system. Those are what we called power-attacks. Those attacks have, as their main disruptive purpose, not just the objective of the saturation of the system's resources but also the exacerbation of the system workload that brings to an high level of energy consumption. Being such attacks particularly stealth, they result to be more and more dangerous since the caused damage could have been detected after a long time together with an irreparable financial-loss. This study therefore highlights that the "protection" methods should not be more expensive than the value of data themselves.

2 Related Works

Security protocols as well as cryptographic primitives, are known to have a significant impact in terms of computational overhead. In fact, several studies have demonstrated that the above-mentioned elements (i.e., security protocols and cryptographic primitives) have a significant impact on the usage and on the workload of the adopted CPUs [9,13,17,23,24].

However, researchers have proposed interesting approaches aimed at engineering and implementing "light-weighted" security protocols describing different ways of operational modes as w ell as different usages in order to minimize the energy consumption. In [6] and [14] it has been analyzed the impact of "lightweight" Cryptography on the energy consumption of sensor nodes, observing that such kind of algorithms allows to reduce in a remarkable manner the electricity consumption of devices that were not powered, in most cases, by a source of constant energy but from small batteries.

In [10] and [26] have been proposed ad-hoc protocols that have low power involvement for the mutual authentication among peers. In [15] and [16], instead, have been analyzed the energetic constraints related to network protocols and in the key management in the domain of Wireless Sensor Networks (WSN). In a more specific way, in [12] has been proposed an encryption model that is power-adaptive for a WSN characterized by devices powered by solar energy. In [7], instead, has been analyzed the computational burden, and so the energy consumption, inducted by the TLS-based communications among IoT devices. Using a more high level approach, on the contrary, in [11] have proposed operational techniques aimed at minimize the energy consumption of the devices involved in the secure wireless sessions. Analogously, in [18] as a suggestion for eventual future directions for engineering new security mechanisms that are *energy-efficient*, has been done a detailed empirical analysis made on the most used security protocols among Internet communications, with the aim of easily identify *energy-bottlenecks* within the context of those mechanisms of secure communication. The work [3] instead, offers a (*cross-devices*) parametric mathematical formulation that allows to evaluate the energetic/computational impact introduced by the secure communication on mobile devices.

3 Energy-Efficient Security Protocols

The goal to execute/configure a secure protocol that will result to be efficient from an energy point of view, can be achieved in two different ways:

Efficient Cryptography Strategies, by making efficient the cryptographic primitives that define the functioning of the security protocol choosing in an accurate and pertinent way the usage of some specific cryptographic algorithms depending on the needed security level as well as on the operating scenario. This is achieved by using in a combined manner hardware and software techniques [5,19,20] in order to improve the encryption performances and, in turn, the energy consumption inducted by such operations.

Energy-Cognizant Security Protocol, by making the security protocols able to dynamically adapt their functioning depending on the working environment by means of the adoption of a set of empirical rules that describe the best operational mode, in some given circumstances, to the advantage of the energy preserving.

In both scenarios, the challenge to obtain secure communications that results efficient from the energy point of view may be dealt with in a very efficient way by analyzing in depth its mode of operation, its security requirements and also the computational bottleneck of the security protocols. The security protocols commonly used, such as SSL/TLS or IPSec, offer indeed the possibility to satisfy security objectives by choosing specific cryptographic primitives within a predefined set. Moreover, the corresponding peers may preliminary agree on the security parameters that influence the operational working modes of the cryptographic algorithms chosen for the secure communication. The computational efforts made by the adopted mechanisms with the aim of obtaining secure communications, influence in a considerably way the consumption of energy and, as a consequence, the energy demand of the devices involved in the communication.

4 A Parametric Assessment Model: Evaluate Energy Consumption of Cryptography Process in Secure Communications

Today, many research efforts in the fields of energy efficiency and power management are concentrated on portable computers and mobile devices without constant power. Approximately 80% of the energy consumed by a mobile terminal is used to transmit data on the different communication channels available [1]. In particular, these entities are connected to radio access subsystems that can be considered "legacy" network nodes, in which the generated traffic originates and ends. These radio subsystems are defined by an IEEE 802.11 Access Point in Extended WLAN network topologies or by a UMTS/LTE Radio Access in the cellular or WWAN scenario. It is necessary, in any case, to point out that the modern wireless devices are often equipped with multiple network interfaces, each characterized by an antenna and a power amplifier. In particular, this amplifier is the component with the greatest impact on overall energy consumption. Therefore, the amount of energy required for a device to support communications can be divided into two different components:

- **Fixed part.** Fixed component based on specific hardware and software features of the device.
- **Variable part.** Variable component, characterized by a proportional request of energy that varies over time in relation to the activities of the device.

Processing the above, each wireless network interface requires a fixed amount of energy, measured $Watt = Joule/second$ ($W = J/s$), so that it can remain operational. Furthermore, it is noted that in order to minimize this energy expenditure, technological research has defined and introduced different operating states for network interfaces. However, even in these states the interface consumes a fixed amount of energy, although significantly lower than that required by the same device in full operation. In addition to these fixed consumption, the energy required for the implementation of a communication is also characterized by consumption that varies proportionally to the transmission time, as well as to the amount of transmitted data (this amount of energy is typically expressed in $\mu J/bit$ or, in an equivalent manner, in $W/Gbps$). As expressed, it is considered, the total energy drained by the data transmission process as the sum of the fixed and variable energy consumption of all the network interfaces of the specific device.

4.1 Encryption Energy Consumption

It has been shown in [8] that private-key cryptographic techniques are about a thousand times faster than public-key ones, since the latter require much more computational power and are therefore more expensive in terms of energy consumption. Therefore, in an energy-saving perspective, guaranteeing the best compromise between safety and energy consumption, a sort of "hybrid approach" is needed, characterized by the combined use of both cryptographic techniques. To formulate a mathematical model for estimating the energy consumption of the overexposed cryptographic operations on a device, it is necessary to take into consideration different aspects: the different processing capacities of the devices, the available network interfaces as well as the number of active sessions for each interface. In order to realistically represent the energy consumption of mobile devices, it is also necessary to consider some form of dependence on their mobility profile. A *mobility scheme* is a structure capable of modeling the behavior of a mobile terminal in relation to different endpoints; this structure should be able to describe the movement patterns of the devices (the so-called "mobility model") whose operational variables must include the position, speed and acceleration of the mobile node, as well as how these parameters vary over time. This information appears to be necessary for a reliable estimation of the transmission power required by mobile devices, in particular when performing secure communications in motion. More in detail, the mobility factor μ_i, can be considered as a measure that characterizes the behavior of the node i during a sample time interval Δt and can be modeled, as described in [25]. Formally:

$$\mu_i = \frac{1}{k\Delta t}\sum_{k=0}^{k-1} \mid A_i(k\Delta t) - A_i\left((k+1)\Delta t\right) \mid \qquad with\ A_i(t) = \frac{1}{N-1}\sum_{j=1}^{N} D_{i,j}(t) \quad (1)$$

where N is the total number of nodes in the network, $D_{(i,j)}(t)$ is the distance between nodes i and j at time t, and with $k = \frac{T}{\Delta t}$ where T is the total observation time.

4.2 Modelling Encryption Energy

Informally, "session" is defined as the interaction between two endpoints in order to exchange messages in a certain amount of time. For each session s, the energy consumption related to cryptographic operations of the device, indicating it with $\varepsilon(s)$, combining the two different energy factors defined in Sect. 4: a fixed amount of energy, not dependent on the data transmitted, and a variable quantity, depending on the quantity of traffic exchanged between the parties and which takes into account the energy-proportional behavior of modern hardware equipment. The variables C_1,C_2,C_3, respectively denote the energy consumption related to a single Authentication operation, Key Exchange and Key Setup. To estimate the energy consumption for each session, it is therefore necessary to define the number of security operations (Authentication, Key Exchange and Key Setup) that a given endpoint performs with respect to another. To this end, the variables R_1,R_2,R_3 denote the number of operations performed (per session) by a device, respectively for Authentication operations, Key Exchange and key initialization. The *data-independent* (fixed) contribution of Authentication, Key Exchange and key configuration operations for each s, taking into account the mobility and communication models of the involved devices, can be modeled via a parameter ϕ_s, formally defined in Eq. 2.

$$\phi_s = \left(C^{(AU)} * R^{(AU)}(s)\right) + \left(C^{(KX)} * R^{(KX)}(s)\right) + \left(C^{(KS)} * R^{(KS)}(s)\right) \quad (2)$$

The part of data-dependent energy consumption necessary, instead, to establish a secure session s is proportional to the amount of data to be processed denoted by p. In particular, it depends (per session) on the amount of energy needed to encrypt a single byte, which is based on several factors; therefore, it is easy to appreciate how the amount of data to be processed influences energy consumption. The energy values associated with the various cryptographic activities can be further determined by considering the specific power characteristics of the CPU, together with the number of cycles required to manage cryptographic processing, which depend on both the efficiency of the encryption and the size of the *payload* p. Each individual Ψ component that characterizes the energy consumption of an given *mobile terminal* within a secure session, can be estimated generically, based on the architecture's implementation and specifications, using Eq. 3:

$$\Psi = M_X \cdot Y \cdot F \cdot V^2 \tag{3}$$

where M_X is the number of machine instructions necessary to execute the operation X, Y is the average number of CPU cycles necessary for the execution of the machine instruction, F is the switching capacity of the CPU (measured in $Farads$) and V is its voltage in exit input (measured the $Volts$).

Clearly, the number of instructions/cycles performed increases with the algorithmic complexity of the associated safety activity, leading to greater energy consumption/demand. Therefore, the energy required for the cryptographic operations for each session is characterized s, through the sum of the fixed and variable energy consumption, as shown in Eq. 4

$$\varepsilon(s) = \phi_s + p \cdot \varepsilon\left(c_{enc}(s), key_{enc}(s), mode(s)\right) \tag{4}$$

As a result, the amount of energy needed to perform cryptographic operations on a device n during a certain time interval Δt, represented with $\varepsilon^n(\Delta t)$, is obtained from the summation of all the individual energy requests associated with each secure session $s \in \sum_n$ involving the device n in the aforementioned time interval, as described in Eq. 5.

$$\varepsilon^n(\Delta t) = \sum_{s \in \sum_n(\Delta t)} \varepsilon(s) \tag{5}$$

5 "Benchmarking" Energy Consumption of a TLS Secure Communication

The *Transport Layer Security* (TLS), successor and evolution of the Secure Socket Layer (SSL), is an IEFT standard and, to date, is one of the most widely used security protocols on the Internet. This security layer allows you to obtain secure communications between two corresponding parties (end-to-end) on TCP/IP networks by offering certain security services: encryption, authentication and integrity protection of data exchanged on unprotected networks. Its most common use is to establish secure web connections: HTTPS (HTTP over TLS). However, although less clearly, this level of security is widely used to encrypt data sent to/from e-mail or together with many other Internet protocols where secure communications are needed. With reference to the Internet Protocol Suite, better known as the TCP/IP suite, TLS is typically superimposed on the Transport Layer or integrated with the higher level applications (e. g., the web browsers) and is divided into two sub-layers: the *TLS Protocol Layer* and the *TLS Record Layer*. The TLS Record Layer operates directly above the transport level and is used, mainly, to encapsulate what comes from the higher level protocols: TLS Handshake Protocol, TLS Change Cipher Specification Protocol and TLS Alert Protocol. The TLS Protocol Layer operates, instead, immediately below the Application Layer, allows the authentication between the peers, the negotiation of the security parameters and the initialization of the same. Of the three protocols that defining it, the TLS Handshake is certainly the most complex. This *"handshake"* consists of a sequence of back-and-forth communications between the corresponding parties aimed at mutual authentication and negotiation of the cryptographic parameters necessary for the establishment of a secure session. For example, the cipher suite ECDHE-RSA-AES128-SHA256 makes explicit the use of ECDH for Key Agreement, RSA for Authentication, while AES-1287 and SHA-256, respectively, for Data Encryption/Decryption operations and Integrity Check (Fig. 1).

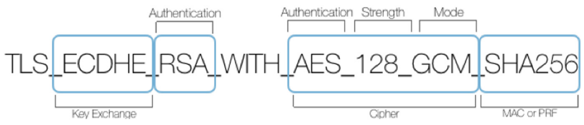


Fig. 1. TLS: cipher suite name construction

With reference to the other two protocols of the TLS Protocol Layer it is observed that the TLS Change Cipher Specification allows the corresponding parties to dynamically update the cryptographic suites used in a secure connection. The TLS Alert, on the other hand, allows the forwarding, therefore the reception, of any warning and/or error messages to the corresponding parts.

5.1 TLS Cryptography Overhead

The speed of communication in a network is determined by two main factors: *bandwidth* and *latency*. Bandwidth is a measure that describes the amount of data that can be sent in a unit of time; the latency represents the delay between the forwarding of a message and its reception. Between the two just mentioned factors, bandwidth is certainly the least relevant because, in general, it is possible to buy more and more bandwidth. Latency, on the other hand, is a limiting factor whenever an interactive message exchange between two corresponding parties is required. In fact, in a typical request-response protocol a certain amount of time is required before a request sent by a client reaches the desired destination and, consequently, so that the eventual reply reaches the applicant. After the latency related to the communication between corresponding parties, the higher cost related to the use of Transport Layer Security derives from the execution of cryptographic operations. The cost is strictly determined by a number of factors including the *private key algorithm* chosen by the server, the *key size* and the *Key Exchange algorithm*.

– *Key size.* The effort required to break a cryptographic key is strictly dependent on its size. The bigger the key, the better its strength.

– *Key Algorithm.* In the current state of the art, the most widely used private key algorithms are: RSA and ECDSA. The RSA, although it is still the most widely used algorithm, is considered to be particularly under-performing from the moment in which it is considered that the minimum size of a cryptographic key, in order to be considered robust, is 2048 bits. ECDSA, on the other hand, is significantly faster than RSA, therefore definitely more suitable for obtaining better performances.
– *Key Exchange Algorithm.* From a theoretical point of view, it is possible to choose between three different algorithms for the key exchange operations: RSA, DHE and ECDHE. The choice of a key exchange algorithm that offers the best ratio between performance and security is definitely ECDHE. The best performances of the two DH-based algorithms are to be attributed to the robustness, therefore to the size of the security parameters negotiated during the configuration phase. It is considered, however, that in a concrete context it is not possible to make an arbitrary combination of the elements mentioned above. It is necessary, in fact, to use one of the four combinations proposed by the security protocol: RSA, DHE_RSA, ECDHE_RSA and EDCDHE_ECDSA.

5.2 TLS Key Exchange CPU Usage Evaluation

In order to evaluate the performance of the Key exchange procedures indicated in Subsect. 5.1, the experimentation method proposed in [22] was followed, appropriately modifying the microbenchmarking tool for OpenSSL proposed by Vincent Bernat [2]. The experimentation was performed in a Linux environment (Ubuntu 14.04 LTS) and taking advantage of the cryptographic library OpenSSL 1.0.1.f (preset by default from the version of the operating system in use) on a HP g6-2338sl laptop with an Intel Core i5-3230M processor at 2.60 GHz. In particular, the test tool included the parallel execution of two different threads (one for client and one for the server) and in the sequential execution of 1000 TLS Handshake, measuring the CPU consumption of the two different threads. It should also be noted that the different suites have been tested by appropriately tuning the key values and safety parameters, based on the values commonly used in a real context. Analyzing the results of this experimentation (Fig. 2), it is possible to observe that, with an RSA key of 2048 bits, a server need computational power about 6 times higher than that of the client. The diametrically opposite scenario is the one described by the use of DH, in which a client needs about two times the computational power of the server. It should be noted, in fact, that the exchange of keys operated with DH is the slowest among the algorithms also used with "weak" security parameters (1024 bits), but it is definitely slower, approximately 15 times, if used with parameters at 2048 bits. However, the most efficient server-side suite seems to be ECDH where the ratio of CPU usage time between client and server is about 2 and the processor usage time values are significantly smaller. Furthermore, with respect to ECDHE, it is noted that the exchange of DHE keys also affects the size of the server-side handshake from 320 to 450 bytes, depending on the robustness of the security parameters used. Finally, it is agreed that the use of DHE, as well as of ECDHE, imposes on the client a workload greater than that of the server. It is necessary to highlight that although these results are fully representative of a possible real implementation, they have been obtained in a virtual scenario and are strictly dependent on the specific release of OpenSSL adopted. In fact, it is explicit that in a concrete operating scenario the performance of Transport Layer Security varies according to the libraries, devices and CPUs involved.

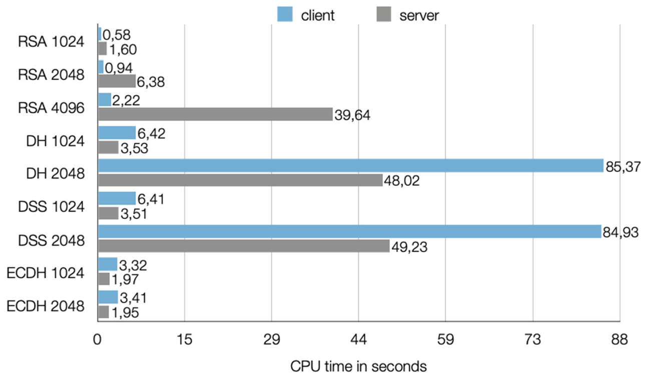


Fig. 2. Experimental results: CPU usage performance comparison of TLS key exchange algorithms

5.3 TLS Symmetric Encryption CPU Usage Evaluation

In terms of CPU consumption, the TLS Handshake is certainly the most CPU-intensive process. It is, however, known that cryptographic operations, particularly those relating to symmetric key cryptographic processes, have a significant impact on the CPU usage load. This overhead is strictly dependent on the encryption algorithm used, on its operating mode and on the integrity checking functions performed. Therefore, with reference to the experimental setup described in Subsect. 5.2, we worked to determine the performance characteristics of the various cipher suites. The performed tests, like the previous ones, were divided into two different threads (one for the client and one for the server). In particular, the client thread sends about 1 Gb of data to the server thread, in blocks of 16 kB, according to the cryptographic suites, to date, most used and deemed safe. In accordance with what is considered in [22], in evaluating the results obtained and reported in Fig. 3, the suite AES-128-CBC was considered as a reference element since, to date, widely used and considered safe. From the above graph, it is possible to consider that AES clearly offers the best performances. It is possible to observe, in fact, that without an hardware accelerator it is faster than all the other ciphers except for RC4. With the AES-NI module, instead, it is agreed that AES-128-CBC is 2.77 times faster than CAMELLIA-128-CBC. Compared to the faster execution of AES, AES-128-GMC-SHA256, it is observed that CAMELLIA-128-CBC is four times slower. AES-128 in authenticated mode (GCM), on the other hand, is 1.4 times faster than the reference AES suite. These results are rather encouraging, considering that, to date, AES is one of the most "robust" available algorithm that can be used. In conclusion it is correct to observe that, although it is common practice to operate server-side benchmarks, it is good practice to evaluate the performance of the encryption also on the client side. This consideration comes from the need to evaluate the effects of secure communications on mobile devices.

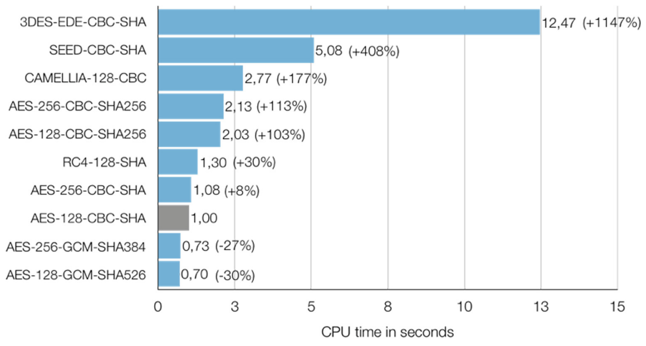


Fig. 3. Experimental results: CPU usage performance of various cipher suites

6 TLS Communication Energy Consumption Evaluation

From the previous tests it emerged that client and server employ a different amount of computational resources in the implementation of a secure communication, finding a different performance profile for each of the possible cipher suites. It is precisely from this performance gap that the idea of evaluating the energy impact of secure communications is born in order to assess how much the application of Cryptography can rise possible energy-oriented attacks. By virtue of the above-mentioned considerations, in order to evaluate the energy expenditure caused by secure communications, hence cryptographic operations, a further experimentation scenario has been defined. In particular, this scenario for secure client server communications consists of two devices interconnected with each other on a LAN using wireless access points. Specifically, the server is represented by the same device used for the previous experiments while the client is a laptop equipped with an Intel Core i5-4278U processor with a clock rate of 2.60 GHz. It is explicit that the server is defined by an Apache Web Server configured in such a way as to make HTTPS connections through the use of a suitably defined self-signed certificate with RSA key at 2048 bits. The energy consumption due to the use of the individual cryptographic suites operated for the definition and implementation of a secure communication were obtained by using a series of repeated requests from the client to the server, with different levels of concurrency, measuring the current absorbed by the power supply. The measurement of the amount of energy absorbed by the server was performed using the TP-Link HS100 smart plug on which the server was powered. By defining appropriate scripts, it was then possible to detect the instantaneous values about the energy absorption of the device. In Fig. 4 the results obtained from the execution of the previously described tests are made explicit. With reference to what is reported in the aforementioned graphic representation, and in relation to the testing method described above, it should be noted that the obtained results, although fully indicative of the problem under analysis, appear to be purely illustrative. From the reported bar graph, in fact, it is not possible to fully appreciate the different amount of energy needed to implement a secure communication by means of the different cryptographic primitives. A more in depth analysis, in fact, should not be limited to assessing the standard variation of the amount of energy drained by the server to satisfy in parallel requests made by n users, but should consider these values in relation to the amount of time needed to satisfy the same requests.

7 "NoCrypto Client TLS Communication": A TLS Energy Oriented Attack

By virtue of the contents of Sect. 5.2, the most complex cryptographic operation performed in a TLS Handshake is the *RSA decryption*. This operation is performed by the server in order to obtain the pre-master secret sent to it by the client in encrypted form. In this scenario, a potential attacker could operate by sending a large number of requests to a target server causing it to perform the aforementioned decryption operation for the sole purpose of exacerbating its computational load. The described modus operandi requires that the same client performs some cryptographic operations, significantly less expensive than server-side operations, but that however involve a significant computational overhead. According to what reported in the previous section and using the strategy proposed in [21] as framework, the script described below has been developed. How much in reference has been elaborated through the use of the Libevent library in order to manage the events, such as the reading of the reply messages sent by the server and the forwarding of further messages to the target. In this script, the messages needed to define the TLS Handshake (Client Hello, Client Key Exchange, Change Cipher Specification and Finished messages) are prepared based on what are the formatting specifications of the protocol [4]. More specifically, the Client Hello message consists of the client Hello header and the body of the message. The client key exchange message is constructed based on the length of the server's RSA key. The Change Cipher Specification message consists of a single byte with value 1. The Finished message, finally, is a structure containing a random data sequence. Analyzing the execution of this script, we observe that the attack begins with the creation of a new socket. This socket is used to open a connection to the server and send the first record structure containing the Hello Client message. Once the receipt of the corresponding Hello message server is

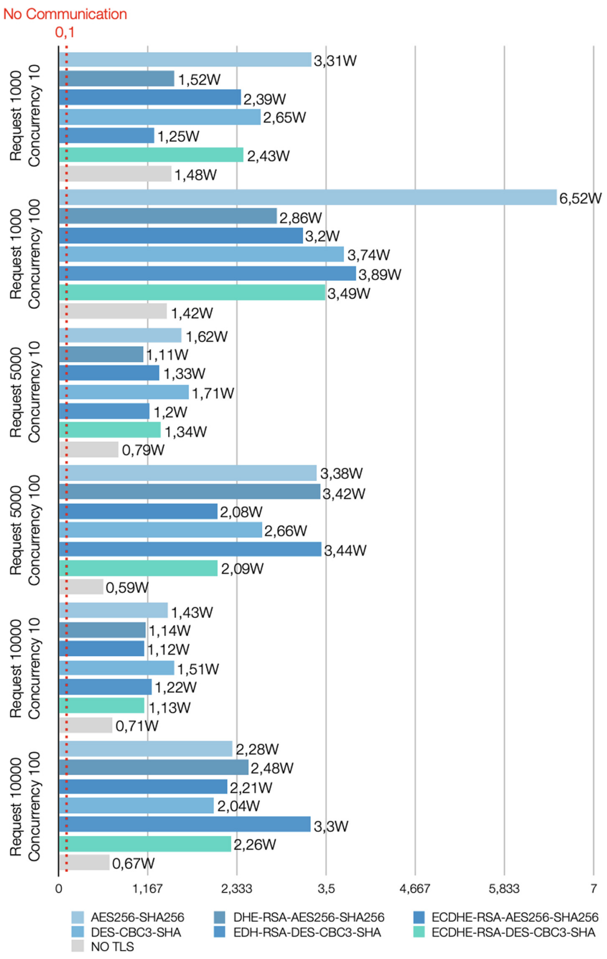
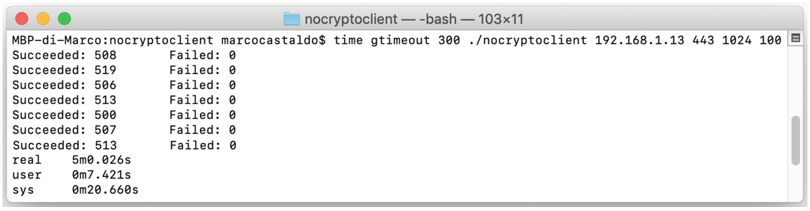


Fig. 4. Experimental results: energy consumption of various cipher suites

confirmed, the tool continues sending the rest of the preassembled message from the Client Key Exchange message followed by a single additional record containing the Change Cipher Specification message and the finished message. The server, having received the Key Exchange message client, decrypts it, verifying, however, that the message received is not properly formatted, and thus closing the connection. It is therefore necessary to highlight the need to correctly configure these messages. Indeed, the forwarding of random or malformed handshake messages would lead to the closure of the server-side connection. The proposed script operates in such a way that the client uses and sends to the server hard-coded handshake messages and therefore does not operate any cryptographic operation.

7.1 Attack Evaluation

Analyzing the output shown in Fig. 5 and obtained from the execution of the script, it is explained that each "Success" indicates that the web server has operated in order to decrypt the dummy message received and then return a bad record MAC message for each one.



```
MBP-di-Marco:nocryptoclient marcocastaldo$ time gtimeout 300 ./nocryptoclient 192.168.1.13 443 1024 100
Succeeded: 508       Failed: 0
Succeeded: 519       Failed: 0
Succeeded: 506       Failed: 0
Succeeded: 513       Failed: 0
Succeeded: 500       Failed: 0
Succeeded: 507       Failed: 0
Succeeded: 513       Failed: 0
real    5m0.026s
user    0m7.421s
sys     0m20.660s
```

Fig. 5. No crypto script: running

Furthermore, by observing the runtime client system parameters during the long execution of the script in a time interval of five minutes, confirmed the fact that being exempt from cryptographic procedures this operating mode does not burden the workload of the CPU, it is possible to highlight that the computational load of the client appears to be almost entirely related to I/O operations. The server side scenario, however, is diametrically opposed. From the graphs shown below (Figs. 6, 7) it is possible to immediately appreciate that after a few seconds from the execution of the script the CPU load and the processor temperature have reached values close to their maximum limit with a consequent increase in the energy consumption of the device. From the obtained results it is not trivial to observe that a client with a modest computing power can easily overwhelm a server with better computational capabilities simply by executing a large number of requests in parallel. It is further necessary to agree that, by

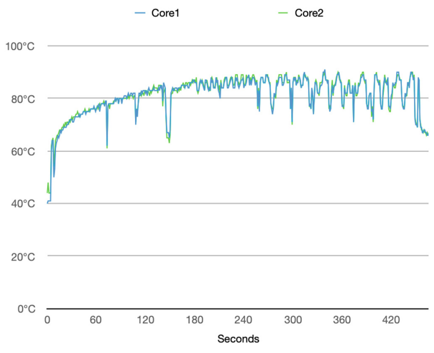


Fig. 6. No crypto script: temperature increase

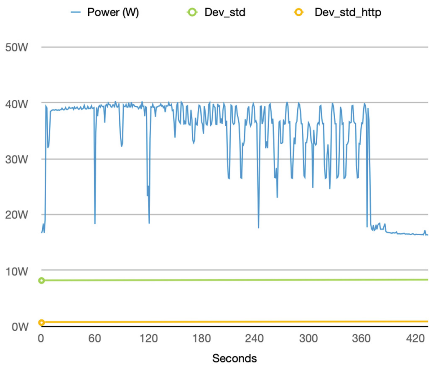


Fig. 7. No crypto script: energy consumption

making the most of the different computational complexity that exists between a client and a server, related to the different cryptographic procedures performed by the corresponding parties during the initialization of a secure communication, it is possible to define the suitable scenario for the implementation of a DoS attack.

8 Conclusions

The technological development and the ever increasing computational power of the devices implies an increase in the energy consumption of the resource itself: costs decrease, capacities increase, but, at the same time, energy consumption increases. Implementing policies and techniques for the protection of data and communications has become essential and indispensable. However, it is known that the cryptographic operations are particularly onerous in terms of CPU workload and, consequently, in terms of energy consumption. In this perspective an energy-aware approach is not yet fully disseminated. A total awareness of the type of data to be protected and on the type of network in which the communication is operated is necessary, in order to address the overexposed problems related to the different needs and different contexts. Therefore, it is necessary to carefully evaluate the choice of cryptographic primitives for a given context, considering the energy expenditure induced by cryptographic operations as a new constraint for the definition and realization of secure communications. An energy-oriented approach contributes to the reduction of the overall energy consumption of telecommunications systems, as well as of the devices that operate these communications, helping their sustainable growth and decreasing the environmental impact.

References

1. Balasubramanian, N., Balasubramanian, A., Venkataramani, A.: Energy consumption in mobile phones: a measurement study and implications for network applications. In: 9th ACM SIGCOMM Conference on Internet Measurement, pp. 280–293 (2009)
2. Bernard, V.: TLS & Perfect Forward Secrecy (online). https://vincent.bernat.ch/en/blog/2011-ssl-perfect-forward-secrecy
3. Castiglione, A., Palmieri, F., Fiore, U., Castiglione, A., De Santis, A.: Modeling energy-efficient secure communication in multi-mode wireless mobile device. J. Comput. Syst. Sci. **81**, 1464–1478 (2015)
4. Castro-Castilla, A.: Traffic Analysis of an SSL/TLS Session (online)
5. Chang, J.K.-T., Liu, C., Gaudiot, J.-L.: Hardware acceleration for cryptography algorithms by hotspot detection. In: Park, J.J.J.H., Arabnia, H.R., Kim, C., Shi, W., Gil, J.-M. (eds.) GPC 2013. LNCS, vol. 7861, pp. 472–481. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38027-3_50
6. DelBello, C., Raihan, K., Zhang, T.: Reducing energy consumption of mobile phones during data transmission and encryption for wireless body area network applications. Secur. Commun. Netw. **8**(17), 2973–2980 (2015)

7. Gerez, A.H., Kamaraj, K., Nofal, R., Liu, Y., Dezfouli, B.: Energy and processing demand analysis of TLS protocol in internet of things applications. In: IEEE Workshop on Signal Processing System (2018)
8. Hardjono, T., Dondeti, L.: Security in Wireless LANs and MANs. Artech House (2005)
9. Jain, A., Bhatnagar, D.: Comparative study of symmetric key encryption algorithms. Int. J. Comput. Sci. Netw. **3**(5), 298–303 (2014)
10. Jakobsson, M., Pointcheval, D.: Mutual authentication for low-power mobile devices. In: Syverson, P. (ed.) FC 2001. LNCS, vol. 2339, pp. 178–195. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46088-8_17
11. Karri, R., Mishra, P.: Minimizing energy consumption of secure wireless session with QoS constraints. In: IEEE International Conference on Communication, pp. 2053–2057 (2002)
12. Kim, J.M., Lee, H.S., Yi, J., Park, M.: Power adaptive data encryption for energy-efficient and secure communication in solar-powered wireless sensor networks. J. Sens. **2016**, 1–9 (2016). https://doi.org/10.1155/2016/2678269
13. Kolamunna, H., et al.: Are wearable devices ready for secure and direct internet communication. GetMob. Mob. Comput. Commun. **21**(3), 5–10 (2017)
14. Lara-Nino, C.A., Diaz-Perez, A., Morales-Sandoval, M.: Energy and area costs of lightweight cryptographic algorithms for authenticated encryption in WSN. Secur. Commun. Netw. **2018**, 1–14 (2018). https://doi.org/10.1155/2018/5087065
15. Law, Y., Dulman, S., Etalle, S., Havinga, P.J.: Assessing Security-Critical Energy-Efficient Sensor Networks. Centre for Telematics and Information Technology (2002)
16. de Meulenaer, G., Gosset, F., Standaert, F., Pereira, O.: On the energy cost of communication and cryptography in wireless sensor networks. In: IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (2008)
17. Naylor, D., et al.: The cost of the "S" in HTTPS. In: 10th ACM International Conference on Emerging Networking Experiments and Technologies, pp. 133–140 (2014)
18. Potlapally, N.R., Ravi, S., Raghunathan, A., Jha, N.K.: A study of the energy energy consumption characteristics of cryptographic algorithms and security protocols. IEEE Trans. Mob. Comput. **5**(2), 128–143 (2006)
19. Potlapally, N.R., Ravi, S., Raghunathan, A., Lakshminarayana, G.: Optimizing public-key encryption for wireless clients. In: IEEE International Conference on Communications, vol. 2, pp. 1050–1056 (2002)
20. Ravi, S., Raghunathan, A., Potlapally, N., Sankaradass, M.: System design methodologies for a wireless security processing platform. In: IEEE Design Automation Conference, pp. 772–782 (2002)
21. Rescorla, E.: SSL/TLS and Computational DoS (online)
22. Ristić, I.: Bulletproof SSL and TLS. Feisty Duck (2014)
23. Salama, D., Kader, H., Hadhoud, M.: Studying the effect of most common encryption algorithms. Int. Arab J. e-Technol. **2**(1), 1–10 (2011)
24. Singelee, D., Seys, S., Batina, L., Verbauwhede, I.: The communication and computation cost of wireless security. In: 4th ACM Conference on Wireless Network Security, pp. 1–4, June 2011
25. Song, J., Miller, L.: Empirical analysis of the mobility factor for the random waypoint model. In: OPNETWORK, pp. 600–700 (2002)
26. Wong, D., Chan, A.: Mutual authentication and key exchange for low power wireless communications. In: MILCOM Proceedings Communications for Network-Centric Operations: Creating the Information Force (2001)