Chapter 5

A Distributed Approach to Energy-efficient Data Confidentiality in the Internet of Things

Andrea G. Forte[1], Simone Cirani[2] and Gianluigi Ferrari[3]

In the Internet of Things (IoT) everything will be connected, from refrigerators to coffee machines, to shoes. Many such "things" will have a very limited amount of energy to operate, often harvested from their own environment. Providing data confidentiality for such energy-constrained devices has proven to be a hard problem. In this article, we discuss existing approaches to data-confidentiality for energy-constrained devices and propose a novel approach to drastically reduce a node's energy consumption during encryption and decryption. In particular, we propose to distribute encryption and decryption computations among a set of *trusted* nodes. We validate the proposed approach through both simulations and experiments. Initial results show that the proposed approach leads to energy savings (from a single node's perspective) of up to 73% and up to 81% of the energy normally spent to encrypt and decrypt, respectively. With such great savings, our approach holds the promise to enable data confidentiality also for those devices, with extremely limited energy, which will become commonplace in the IoT.

5.1 Introduction

The Internet of Things (IoT) will introduce a drastic change in our society as many common things that we use in our every-day life will stop being just objects and will start interacting with each other, with us, and with the environment. As all of these things talk to each other and to us, they will collect and share a very large amount of sensitive information. For example: car engines will collect and share data about the driver's driving habits; people's whereabouts will be shared and used to automatically provide new services; and what we eat and drink could be shared with insurance companies to monitor our eating habits and lifestyle. Similarly, people's vital signs (e.g., heart rate, blood pressure) may be collected and shared in a

[1]Internet of Things (IoT) Lab, Department of Engineering and Architecture, University of Parma, Parma, Italy. Andrea G. Forte was with the IoT Lab and AT&T (USA) when this work was carried out: he is now Chief Security Officer, Bitsian, New York, NY 10271, USA.
[2]Caligoo srl, Taneto di Gattatico (Reggio Emilia), Italy - simone.cirani@caligoo.com
[3]IoT Lab, Department of Engineering and Architecture, University of Parma, Parma, Italy. - gianluigi.ferrari@unipr.it

completely automated way, with privacy implications that are still not very clear. Given the capillarity of the IoT, its pervasiveness in our lives, and given the sensitive nature of the information it will handle, security in the IoT—and, specifically, *data confidentiality*—has become of utmost importance.

The goal of data confidentiality is to prevent disclosure of information to unauthorized parties. When sharing sensitive data, confidentiality is usually enabled by encryption.[4] Over the years, much work has focused on improving both computational and energy efficiencies of encryption and decryption algorithms in block ciphers (such as AES). Alternatively, new lightweight block ciphers have been proposed trying to address encryption in energy-constrained networks (e.g., wireless sensor networks). While all of these approaches have their merits, often they are not sufficient to provide the energy savings that would allow them to be implemented in many devices in the IoT and, specifically, in energy-harvesting devices.

In this paper, we propose to distribute block cipher encryption and decryption operations between a set of *trusted* energy-constrained devices. This will be shown to bring significant energy savings to a device. The advantage of such an approach is twofold: on one hand, traditional block ciphers can now be used in energy-constrained devices; on the other hand, the energy consumption of "lightweight" block ciphers can further be reduced, thus increasing a node's battery lifetime.

This paper is structured as follows. In Section 5.2, we start discussing the main challenges in providing data confidentiality in the IoT. In Section 5.3, we show how the energy consumption brought by encryption and decryption operations can be drastically reduced by distributing their computation among a trusted set of energy-constrained nodes. Our approach is validated through energy measurements performed both on an Arduino-based testbed (Section 5.4) and on simulated Zolertia Z1 motes (Section 5.5), showing promising energy savings of our distributed approach. Finally, concluding remarks are presented in Section 5.6.

5.2 Data Confidentiality in the IoT

Data confidentiality in the Internet is usually enabled by encryption. Traditional encryption algorithms, however, are too slow and energy-demanding to be applied to the IoT. In particular, the amount of energy spent in encryption and decryption is a critical factor as we move towards wearables, energy-harvesting wireless nodes, and the IoT.

Efforts in reducing block ciphers energy consumption can be divided into three main categories: hardware optimizations, software optimizations, and new "lightweight" algorithms. Let us look at each one of these.

Hardware optimizations [1–3] mainly focus on reducing the number of logic gates necessary to implement a block cipher.[5] A smaller circuit consumes less energy than a larger one: therefore, by reducing the circuit size, the block cipher consumes less energy. These approaches usually require the use of very specialized hardware, making their feasibility quite difficult if not for very peculiar applications.

[4] Other means include the use of file permissions and access control lists.

[5] Another important metric is the number of cycles required to process one block.

Software optimizations [4,5] aim at reducing both the memory footprint and the computational power of block ciphers. Using a smaller amount of computer memory and fewer CPU cycles translates to a lower energy consumption. These approaches achieve some energy savings which, unfortunately, are not sufficiently significant to satisfy the strict energy requirements that some IoT nodes will likely have. Because of this, over the years new lightweight block ciphers have been proposed [6–9] trying to address the energy requirements typical of wireless sensor networks and the IoT. The basic idea behind these block ciphers is to consume little energy while providing lower throughput and a moderate level of security. In particular, the implicit assumptions behind this idea are the facts that in a constrained environment the amount of data to encrypt will be small and an attacker will have limited computing capabilities, so that a moderate level of security is appropriate. The key length for such lightweight block ciphers is not too long, typically between 80 and 128 bits, and they usually operate on a smaller 64-bit block size of input data. Their low energy consumption makes them suitable for many resource-constrained devices at the price, however, of limited protection. On the other hand, the reduced level of protection due to the use of smaller encryption keys can be increased by introducing periodic key-changing mechanisms (whose rate strictly depends on the amount of security targeted) in order to make it more difficult to perform brute force attacks on these ciphers [10].

Distributing block ciphers computations further improves on lightweight block ciphers as constrained nodes can now compute a subset of the lightweight block cipher rounds, thus increasing the energy savings even more. Also, such greater energy savings would allow to use block ciphers in energy-harvesting devices, i.e., devices with the most stringent energy requirements. Last, a distributed computation can drastically increase, from a node's perspective, the energy savings for "traditional" block ciphers such as AES, thus enabling the use of AES in constrained environments without the use of any specialized hardware.

5.3 A Distributed Computation Approach

A large number of devices in the IoT will be energy-constrained and will often have to operate using just a few micro-joules of energy [11]. Because of their limited energy, such devices (i.e., nodes) typically form multi-hop networks. In other words, instead of sending packets directly to a far-away gateway, which would require higher transmission power, they relay each others' packets all the way to the gateway by using appropriate routing protocols [12]. In doing so, less energy is consumed in transmission as neighboring nodes are typically much closer than the gateway.

As mentioned in Section 5.1, we propose to distribute block cipher operations among a set of trusted nodes. In particular, this approach is motivated by two considerations. The first one is the multi-hop nature of communication for energy-constrained nodes: as such nodes relay each other's packets, they can thus easily apply incremental computations on them. The second one is the iterative structure of block ciphers, usually either a Feistel network or a Substitution-Permutation net-

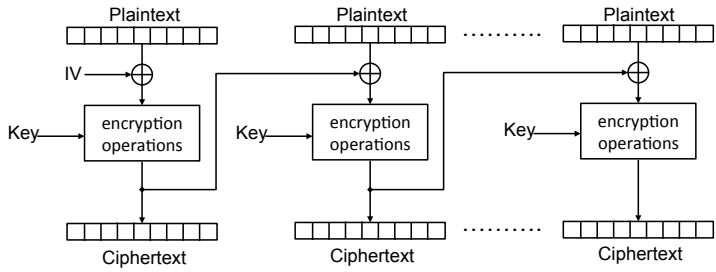


Figure 5.1: Example of block cipher iterative structure in Cipher Block Chaining (CBC) mode.

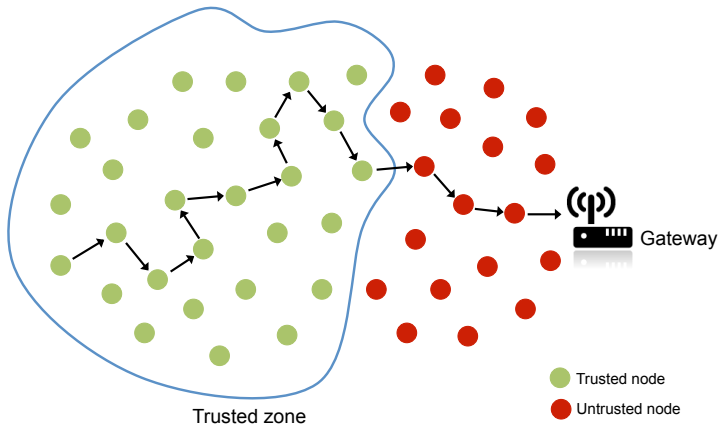


Figure 5.2: Multi-hop network: distributing block-ciphers computations among the trusted nodes.

work: an illustrative representation is shown in Fig. 5.1. For both block cipher structures, the same operations are repeated multiple times in different stages or rounds. Because of this iterative computation and because of the multi-hop nature of the communication protocol between nodes, it is natural to think to distribute the rounds of encryption of a given block cipher (e.g., 10 rounds in AES128) among a selected set of nodes in the multi-hop network.

The nodes participating to the distributed computation need to be *trusted*. More precisely, they must be positioned in such a way that: (i) their communication cannot be eavesdropped and (ii) they cannot be hijacked or tampered with. In Fig. 5.2, an illustrative representation of such a scenario is shown. For as long as all the required rounds of encryption are completed within the trusted zone, everything is fine. If, however, a packet reaches the boundary of the trusted zone before all rounds of encryption have been performed, then the content of the packet, as well as the secret key used by the block cipher, may be compromised.

A multi-hop network with a trusted set of nodes, as the one just described, could be any network of smart objects deployed in hard-to-reach places, such as under-sea or underground networks. Saving energy and prolonging battery lifetime in such networks is very important, given the difficulty in replacing the batteries, if at all possible. For example, a critical infrastructure in the city of New York is the underground network of steam pipes which runs all throughout the city and is meant to keep water pipes temperatures high during winter so to prevent them from freezing and bursting. In an IoT scenario, one can imagine having sensors and actuators attached to these underground steam pipes. The sensors would monitor the temperature level of the water pipes and communicate with nodes above ground. Such "external" nodes would monitor weather conditions and pipes temperatures in order to "tell" the actuators how much steam to push into the underground system: this would allow efficient management of the pipes and, therefore, economic savings. Clearly, the sensors monitoring the underground water pipes and the actuators cover a very vast area and collect sensitive information—in fact, misuse of such information may cause severe damage. It is, therefore, important that their communication to the external nodes is kept confidential (i.e., encrypted) while, at the same time, using the least amount of energy. In such a scenario, sensors and actuators can distribute block ciphers computations making sure that, by the time packets are sent to nodes above ground, encryption has correctly completed. Nodes operating underground fit our definition of "trusted."

Clearly, the way packets are routed through the multi-hop network is very critical. As part of future work, one attractive research direction is the design of a routing protocol, operating in the trusted zone of the multi-hop network, able to maximize the lifetime of the set of trusted nodes while making sure that packets have been fully encrypted by the time they leave the trusted zone. This opens interesting research questions about possible cooperation strategies among trusted nodes.

5.4 Arduino-based Experimental Analysis

5.4.1 Testbed Setup

In order to measure a node's energy savings when distributing encryption and decryption computations, we use the Arduino testbed shown in Fig. 5.3. This testbed includes two Arduino UNO R3 boards equipped with an Atmel ATMEGA-328p microcotroller and an Adafruit INA219 board. In particular, encryption and decryption are performed by one of the Arduino UNO boards (indicated as, namely, the Arduino UNO (2) in Fig. 5.3). This board was powered by a 9 Volt battery. Voltage and current used by the board were measured by the Adafruit INA219. The Adafruit INA219 was powered and driven by another Arduino board (indicated as, namely, the Arduino UNO (1) in Fig. 5.3) which would also collect all the measurements received from the Adafruit INA219 and would then send them to a laptop via USB. On the laptop, the measurement readings would be cleaned up and further processed using some Python scripts in order to compute the actual energy consumed by the Arduino board.

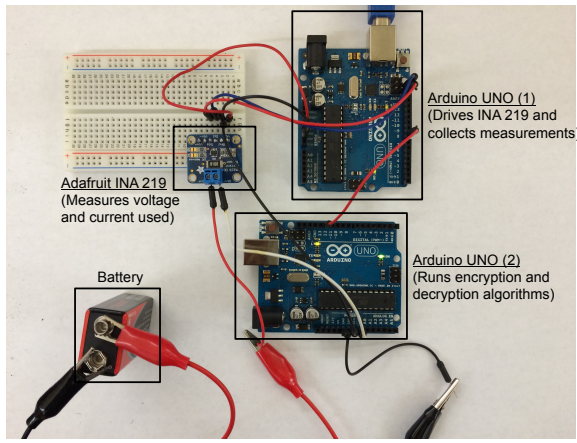


Figure 5.3: Arduino testbed used for the experiments.

From a software standpoint, we modified Davy Landman's AESLib library [13] in order to let the Arduino node perform a custom number of rounds of encryption and decryption per each packet sent or received.

5.4.2 Experimental Measurements

We measure the energy consumption and savings of the AES block cipher as it is widely used in the Internet (e.g., Transport Layer Security (TLS) protocol [14]). AES is a block cipher that operates on 128-bit blocks of plaintext and outputs 128-bit blocks of ciphertext. The key length in AES can be 128, 196 and 256 bits, which correspond to 10, 12, and 14 rounds, respectively. We focus on AES with a key length of 128 bits and, thus, 10 rounds. Furthermore, there are several modes of operation for AES. Throughout our experiments we consider the Cipher Block Chaining (CBC) mode, as it is one of the most used. In CBC mode, an Initialization Vector (IV) must be used in order to make each message unique. Furthermore, as we can see from Fig. 5.1, at each round a block of plaintext is first XORed with the previous block of ciphertext and then encrypted.

5.4.2.1 Energy Measurements

Our first objective is to measure how much energy a node would save when using distributed block cipher computations. In order to do this, we measure the energy consumed by a node running AES128-CBC and compare it with the energy spent by the same node running one single round of AES128-CBC. As shown in [15], the encryption operations in our testbed use, on average, 1.24 mW of power, while decryption operations use, on average, 1.79 mW of power. In order to measure the energy consumed, knowing the used power, one needs to measure how long encryption and decryption operations take.

# of Rounds	Encryption			Decryption		
	Duration (ms)	**Energy (μJ)**	**Savings (%)**	**Duration (ms)**	**Energy (μJ)**	**Savings (%)**
1	3.25	4.05	73	2.66	4.77	81
2	4.22	5.26	65	3.92	7.02	72
3	5.19	6.47	56	5.17	9.27	63
4	6.16	7.68	48	6.43	11.52	54
5	7.13	8.89	40	7.68	13.77	45
6	8.1	10.1	32	8.94	16.02	36
7	9.07	11.31	24	10.19	18.27	27
8	10.04	12.52	16	11.43	20.48	18
9	11.01	13.73	8	12.70	22.77	9
10	11.98	14.93	0	13.96	25.02	0

Table 5.1: Average time and energy spent for encryption and decryption of 1024 bytes for a different number of rounds of AES128-CBC. The average savings of a node performing a distributed computation of AES128-CBC are also shown.

Table 5.1 shows, among various performance metrics, the duration of encryption and decryption operations for different numbers of rounds for a 1024-byte packet size. One can first observe that while for both one-round and two-round computations encryption takes longer than decryption, the opposite is true for a number of rounds larger than two. In particular, for full AES128 (i.e., 10 rounds), encryption takes, on average, 11.98 ms, while decryption takes, on average, 13.96 ms. On the other hand, for a single round of AES128, encryption takes, on average, 3.25 ms, while decryption takes, on average, 2.66 ms. Interestingly, for both encryption and decryption, the time required to run full AES128 (i.e., 10 rounds) is considerably shorter than ten times the time required to run one round of AES128. Consequently, the overall energy spent by running the non-distributed version of AES128 is much lower than the overall energy spent by running one round of AES128 ten times. As we confirmed in our extended measurements [15], this difference is due to the initial overhead of AES128 caused by initialization operations such as key expansion. In particular, this initial overhead is responsible for an additional average consumption of 2.84 μJ for encryption and of 2.53 μJ for decryption. In the Arduino AES library used in our testbed, such overhead is present at each round of the distributed version of AES128, whereas it is present only in the first round of the non-distributed version of AES128. This difference in the initial cost is the reason why running one round of AES128 ten times (i.e., distributed AES128) is much more costly, in terms of energy, than running the non-distributed version of AES128. This clearly shows that one of the optimizations required in distributing block cipher computations is to make sure that any initialization cost incurs only once, at the first round, and does not get propagated to subsequent rounds of computation. By taking this into account, the total amount of energy spent by running distributed block cipher computations must match the amount of energy spent when running the non-distributed version of the same block cipher.

As shown in Table 5.1, in our measurements we have observed that the node running the first round of distributed encryption consumes, on average, 4.05 μJ of energy. However, nodes performing subsequent rounds of encryption spend, on aver-

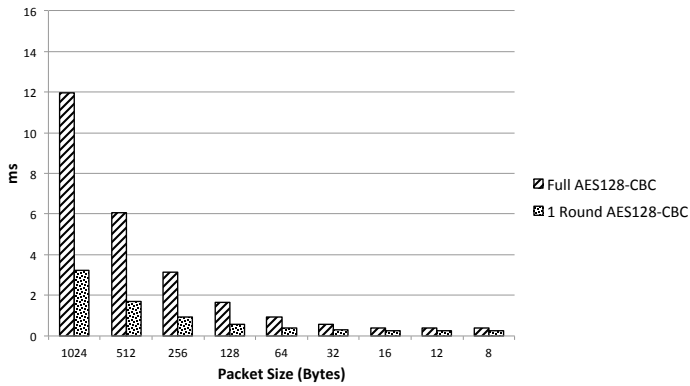


Figure 5.4: Encryption time as a function of the packet size.

age, 1.21 μJ per round of encryption due to the lack of initialization costs. Similarly, for decryption, the node running the first round of decryption would consume, on average, 4.77 μJ, while nodes running subsequent rounds would consume, on average, 2.24 μJ.

Naturally, energy savings reduce as nodes run an increasing number of rounds of block-cipher computations. In Table 5.1, we also show the energy savings of a node when performing a different number of rounds. As one can see, from energy savings of 73% in encryption and of 81% in decryption, when performing 1-round computation, a node reduces its energy savings to 48% in encryption and 54% in decryption when performing 4-round computations of the same block cipher. Such savings drop down to 8% in encryption and 9% in decryption when performing 9-round computations. As expected, the larger the number of computed rounds, the lower the energy savings—obviously reaching zero for 10 rounds (i.e., full AES128).

We now investigate the impact of the packet size on AES128 power consumption. In Fig. 5.4 and Fig. 5.5, the encryption and decryption times are shown, respectively, as functions of the packet size. In both cases, full AES128 (i.e., 10 rounds) and one single round of AES128 are considered. In particular, we measure the time required by encryption and decryption operations for packet sizes between 8 bytes and 1024 bytes. It can be observed that the duration of encryption and decryption decreases linearly with the decrease of packet size. However, below 16 bytes (i.e., 128 bits), encryption and decryption times tend to remain constant. This behavior is due to the fact that in AES128-CBC packets smaller than 16 bytes are padded to the block size of 16 bytes.

In Fig. 5.6, the energies consumed by (a) encryption and (b) decryption are shown as functions of the packet size, considering full AES128 and one round of AES128. In both encryption and decryption, one can see that by using a distributed block cipher, the highest energy savings are obtained with large packet sizes. As the packet size decreases, so do the energy savings, down to a point where performing full AES128 takes almost the same amount of energy as performing a single round

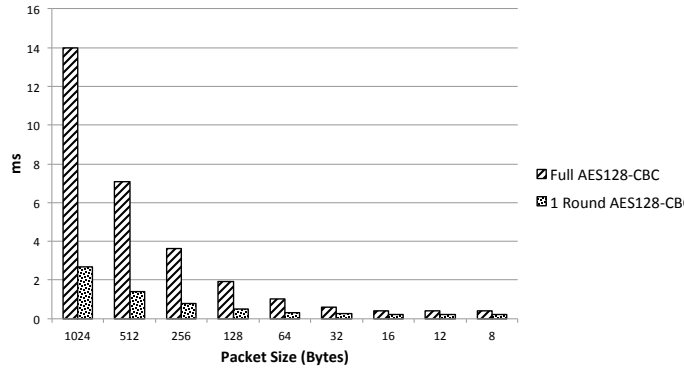


Figure 5.5: Decryption time as a function of the packet size.

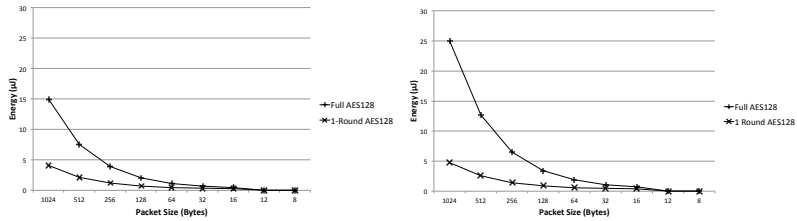
Size (Bytes)	**Encryption (%)**	**Decryption (%)**
1024	73	81
512	72	80
256	69	78
128	65	74
64	59	68
32	49	58
16	36	45
12	36	45
8	36	45

Table 5.2 Average energy savings (in both encryption and decryption), for different packet sizes, when using AES128-CBC as a distributed block cipher and performing 1 round.

of AES128. This result is very surprising, as one would expect the savings to be roughly the same, in relative terms, for any packet size. By comparing Fig. 5.6a and Fig. 5.6b, it can also be concluded that such savings are greater for decryption than for encryption.

Table 5.2 shows the energy savings that a node would have by using a distributed block cipher and computing a single round of AES128 as opposed to running full AES128. As can be seen, for larger packet sizes one can save up to 73% and up to 81% of the energy usually spent during encryption and decryption, respectively. For decreasing packet sizes, the energy savings decrease as well, although they still remain quite significant. For packet sizes smaller than or equal to 16 bytes, the savings remain constant due to padding.

In many IoT applications, packet sizes tend to be quite small. There is, however, a vast literature on the use of data fusion [16], data aggregation [17, 18] and other



(a) Encryption (b) Decryption

Figure 5.6: Comparison, in terms of consumed energy as a function of packet size, considering full and 1-round AES128-CBC.

techniques [19, 20], in the IoT, to combine multiple packets together, before sending them to the gateway, in order to save transmission energy. The reason for combining packets together is that while sending a short packet consumes less energy than sending a large one, sending many small packets is less efficient due to the packet headers that, overall, reduce the goodput. Therefore, the following observations hold:

- just by looking at the packet size, more energy is spent to transmit large packets than smaller ones;
- considering the transmission energy spent per bit of payload (i.e., excluding headers content), the opposite is true and more energy can be saved by sending a few larger packets.

A data fusion approach thus goes well together with our findings of higher energy savings, in terms of encryption and decryption energy costs, for larger packet sizes. By using data fusion together with a distributed block cipher computation, transmission energy costs, as well as encryption and decryption energy costs, can be simultaneously reduced.

5.4.2.2 Lifetime Increase: a Single Node's Perspective

As we discussed earlier and shown in Table 5.1, a node performing one round of encryption can save 73% of the energy normally spent in encryption. For an Arduino UNO board powered by a 9 Volt alkaline battery, this means about 9 minutes of extra battery lifetime, which roughly corresponds to 3% of the total battery lifetime. While this may seem like a modest energy saving, one needs to keep in mind that we are reducing the energy consumption of encryption and decryption operations which use between one and two milliwatts of power, while the Arduino UNO board has an *operational power* of hundreds of milliwatts, that is, two orders of magnitude higher than both the encryption and decryption powers. Given this huge difference in power consumption, a 3% battery lifetime increase achieved by reducing only the energy consumption of encryption operations is quite impressive. For energy-harvesting devices, where the power used in cryptographic operations has the same order of magnitude of the operational power, significantly greater savings are expected.

Another cause of significant energy consumption is *transmission power*. In our measurements, a Digi XBee S1 802.15.4 RF module, connected to an Arduino UNO board, used about 71 mW of power to send a 1024 byte packet at 9.6 kbps. This power is roughly one order of magnitude higher than what is required by encryption and decryption operations. In the IoT, however, a large class of devices will be energy harvesting devices. For such devices, operation and communication costs, in terms of energy, become comparable to computational cost and, as such, to the cost of cryptographic operations. For example, Energy Harvesting Active Networked Tags (EnHANTs) [11] use a few microjoules of energy to transmit a packet. For such devices, great savings in encryption and decryption translate to great savings in overall node energy consumption, as the energy spent in encryption and decryption represents a large part of the node's energy budget.

5.4.2.3 Lifetime Increase: a Multi-hop Network Perspective

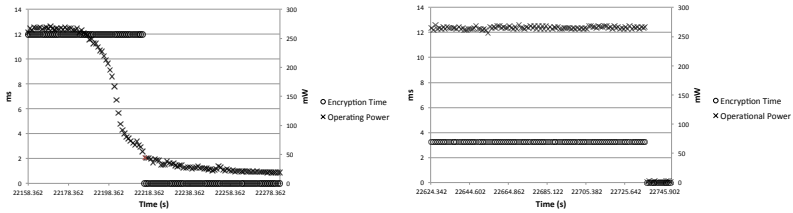
Up to this point we have discussed energy savings from a single node's point of view, without taking into account network dynamics. When we consider not just one node but the whole multi-hop network of trusted nodes (i.e., trusted zone in Fig. 5.2), some important observations can be carried out.

We have seen that a single node performing one round of encryption can save 73% of the energy normally spent in encryption operations. If, however, this node performs multiple rounds of encryption, then its energy savings decrease as per Table 5.1. In a multi-hop network, a node can generate its own packets but can also relay other nodes' packets. For example, if a node generates one packet and receives nine packets from other trusted nodes, since on each of these ten packets one round of AES128 encryption[6] is performed, it will then consume an amount of energy equal to ten rounds of encryption. This is the same amount of energy the node would have consumed by just encrypting its own packet with "traditional" AES128 (i.e., in a non-distributed manner). Therefore, apparently, there would be no savings in using a distributed computation approach in such a scenario. To make things worse, if the node receives packets from twenty trusted nodes and performs one round of encryption on each one of those, it would spend twice the amount of energy that it would have spent by encrypting only its own packets with "traditional" AES128. More generally, if the trusted zone of the multi-hop network generates and processes a given amount of packets, it will consume, overall, the same amount of energy with or without distributing encryption computations. So, why using a distributed approach for encryption and decryption?

The energy savings obtained by distributing encryption and decryption computations, at a network level, depend on two important factors: battery residual energy distribution among the nodes in the network and network topology. Taking into account those factors, various challenges emerge.

In an IoT application including wireless sensors, for example, nodes will detect events, thus generating traffic, with distributions that are not spatially uniform. Because of this, some nodes will generate a large amount of traffic, while others

[6]The first, more energy demanding, round for its own packet and a subsequent round for other nodes' packets.



(a) Full AES128-CBC encryption (b) One round AES128-CBC encryption

Figure 5.7: Tail of the battery discharging profile.

may not generate any traffic at all, but act only as relays. This asymmetry in traffic generation contributes to creating a non-uniform battery residual energy distribution among the nodes in the network—under the implicit assumption that all nodes have the same initial battery energy. Furthermore, if the nodes are energy harvesting nodes, the different rate at which each node can harvest energy will further accentuate the difference in the available energy among nodes. In such a scenario, a routing protocol, as the one described at the end of Section 5.3, would make it possible to route packets so that nodes with a larger amount of available energy would perform more rounds of encryption while nodes with less energy could save it for routing purposes. In doing so, the overall lifetime of the network would increase as nodes with a small amount of energy would prolong their lifetime by delegating encryption to nodes with a higher energy budget. In other words, a routing protocol using distributed encryption computations would increase the network lifetime by distributing encryption operations in a non-uniform way among nodes, following the non-uniform energy distribution. In multi-hop networks, network topology is very important. In particular, because of various reasons (e.g., physical obstructions, nodes layout), there could be a few nodes that have to relay a large amount of traffic between different parts of the network or between the network and the gateway. This is especially true in network topologies considered in IoT scenarios, such as those created by the use of the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) [12]. Such nodes are extremely important since, if they die, large parts of the network, if not the whole network, are likely to remain isolated and packets will not reach the gateway any longer. Therefore, it is important, for such nodes, to live as long as possible regardless of their initial energy budget. In such a scenario, by distributing encryption computations, the routing protocol could take this into account and would not assign any encryption computations to such nodes in order to maximize their lifetime and, thus, the lifetime of the network of trusted nodes.

5.4.2.4 Battery Discharging Profile

As shown above, nodes participating in a distributed block cipher will be able to encrypt and decrypt packets at a much lower energy level. Because of this, a battery used by such nodes will reach a discharged state with a residual energy lower than that of a node operating the full block cipher.

Fig. 5.7 shows the tails of the discharging profiles of (a) a battery powering a node performing full AES128 encryption and (b) a battery powering a node performing one round of distributed AES128 encryption. In both cases, the same battery type was used (i.e., 9 Volt alkaline battery) and nodes were encrypting a 1024 byte packet every 10 ms. When the encryption time goes to zero, the node has no longer enough energy to work properly and cannot encrypt or decrypt packets correctly. As can be seen, while discharging for full AES128 is slow and gradual, in the case of 1-round AES128 more energy can be "squeezed out" of the battery, which lasts longer but dies abruptly.

5.5 Zolertia-based Simulation Analysis

5.5.1 Simulator Setup

While measurements on the Arduino testbed have already proven the very high energy savings our approach can provide, we want to explore what kind of energy savings can be expected when distributing computations of lightweight block ciphers that have been engineered specifically for low-power devices. In particular, in the following simulations we focus on the energy spent during encryption by the Scalable Encryption Algorithm (SEA) [6] and the Tiny Encryption Algorithm (TEA) [9]. Furthermore, we compare such savings with the ones achieved by AES128-CBC.

Simulations have been conducted on Contiki OS-based constrained devices [21], using the Cooja simulator [22]. Contiki OS is a specialized low-power operating system for constrained devices that provides implementations of the IPv6 stack and RPL routing, as well as several MAC protocols. The version of Contiki OS used is 2.7. The experimental platform is based on Zolertia Z1 nodes, with nominal 92 kB ROM (when compiling with 20-bit architecture support) and an 8 kB RAM. In practice, the compilation with the Z1 nodes has been performed with a 16-bit target architecture, which lowers the amount of available ROM to roughly 52 kB. The energy consumption of the CoAP server has been evaluated using Powertrace, a tool for network-level power profiling of low-power wireless networks. Powertrace determines the energy consumption by estimating the amount of time each operation mode of the device is being used. We refer to the current consumption of each component indicated in the Z1 datasheet. In particular, the MSP430f2617 microcontroller consumes 0.515 mA in active mode (CPU) and 0.5 μA in low-power mode (lpm), respectively. Similarly, the CC2420 radio transceiver consumes 17.4 mA in transmitting (TX) mode and 18.8 mA in receiving (RX) mode. In order to obtain the total consumed energy for the node, the following conversion formula has been used:

$$E = \sum_{j \in \mathscr{M}} i_j \cdot v \cdot \Delta t_j \tag{5.1}$$

where: $\mathscr{M}$ is the set of all operation modes of the node (active, lpm, TX, and RX); v is the nominal voltage of the node; and Δt_j is the time the node was in the j-th operation mode j.

A linear topology has been used to deploy simulated nodes.

	Full Encryption		1-Round Encryption		
Block Cipher	**Duration (ms)**	**Energy (μJ)**	**Duration (ms)**	**Energy (μJ)**	**Energy Savings (%)**
AES	18.68	28.87	4.9	7.58	73.74
SEA	20.58	31.79	3.1	4.79	84.93
TEA	20.04	30.96	1.9	2.88	90.69

Table 5.3 Average time and energy spent by AES, SEA and TEA for full encryption and 1-round encryption of 16-byte, 12-byte and 8-byte payload packets, respectively. The average energy savings of a node performing a distributed computation are also shown.

5.5.2 Simulation Results

Table 5.3 shows the results of our simulations. As we can see, when doing a full encryption, AES performs much better than SEA and TEA. This is due to the fact that SEA and TEA have been designed to be extremely efficient when implemented using dedicated hardware. In software, however, they do not perform very well and AES represents a better option. On the other hand, by distributing block cipher computations, a single round of SEA and TEA performs better, energy-wise, than a single round of AES. This apparent contradiction is due to the fact that while each round of either SEA or TEA consumes less energy, the total numbers of rounds for SEA and TEA are 51 and 32, respectively: these values are much larger than the 10 rounds of AES. Moreover, in this case we can see that the energy spent by a block cipher when performing the full encryption is much lower than the product of (i) the energy spent by a single round of encryption and (ii) the number of rounds. This, as discussed for the Arduino measurements, is due to a higher energy cost of the first round of encryption, which takes into account an initialization phase not present in subsequent rounds. As mentioned earlier, when distributing block cipher computations, we have to make sure to incur in this initialization cost only once so that the total amount of energy spent when distributing the computations equals the amount of energy spent in the non-distributed case.

For all three block ciphers, we have very large energy savings. In particular, from a single node's perspective, with TEA the energy saving for a single node reaches 90.69%, followed by SEA with saving equal to 84.93%, and, lastly, by AES with 73.74%. We remark that, for AES, our simulation results perfectly match, in terms of energy savings, our experimental results. At the network level, however, things are different since the larger number of rounds of SEA and TEA translates into a larger amount of energy spent by the network as a whole. Moreover, routing becomes more challenging given that a larger number of rounds means that a packet may need to be routed through a larger number of nodes (i.e., longer paths). In such cases, there are tradeoffs between route complexity, nodes energy savings and overall network energy level.

Table 5.4 shows the average energy consumption of AES, SEA, and TEA for packet TX and RX. Given that the packets sent and received by the three block ciphers have different sizes, we cannot compare their TX and RX energy consump-

Block Cipher	RX Energy (mJ)	TX Energy (mJ)
AES	1.12	0.37
SEA	0.94	0.5
TEA	3.24	0.88

Table 5.4 Average energy spent in RX and TX

tions. However, we can see how, for all three block ciphers, RX energy is higher than TX energy. This is consistent with the observations in [11]. In particular, this is due to the fact that in low-power applications, such as those in the IoT, for packet transmission the radio needs to stay on only for the duration of the transmission; at the opposite, for packet reception the radio needs to stay on for a longer amount of time. Since perfect node synchronization consumes much energy and is, therefore, not implemented, the exact arrival time of a packet is, in fact, unknown

5.6 Conclusions and Future Work

After a short discussion on the challenges in provisioning data confidentiality in the IoT and after presenting some existing solutions, we have introduced a novel approach to greatly reduce the energy spent by a node during encryption and decryption operations. In particular, given the iterative structure of block ciphers and the multi-hop nature of communication protocols in the IoT, a node can save a large amount of energy by distributing block ciphers computations. We have experimentally measured node's energy savings, for a distributed version of AES128 in CBC mode, of up to 81% and of up to 73% for decryption and encryption, respectively. We have also measured (through simulations) energy savings for lightweight block ciphers (namely, SEA and TEA), showing encryption energy savings up to 84.93% and 90.59% for SEA and TEA, respectively.

Given the sensitive nature of block-cipher computations, such computations should be distributed among a set of *trusted* nodes. One of our current research activities consists of the design of a new routing protocol aimed at maximizing the network lifetime while, at the same time, guaranteeing encryption distribution among the set of *trusted* nodes. Finally, we plan to apply the same distribution principle to the computation of cryptographic hashes.

References

[1] Satoh A, Morioka S, Takano K, et al. A compact Rijndael hardware architecture with S-box optimization. In: Springer, editor. Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT) – Advances in Cryptology. Gold Coast, Australia; 2001. p. 239–254.

[2] Feldhofer M, Wolkerstorfer J, Rijmen V. AES implementation on a grain of sand. IEE Proceedings – Information Security. 2005 November;152(1):13–20.

[3] Wolkerstorfer J, Oswald E, Lamberger M. An ASIC implementation of the AES S-Boxes. In: Springer, editor. Topics in Cryptology (CT-RSA). San Jose, CA, USA; 2002. p. 67–78.

[4] Tillich S, Großschädl J. Instruction set extensions for efficient AES implementation on 32-bit processors. In: Springer, editor. Proc. of the 8th International Workshop on Cryptographic Hardware and Embedded Systems (CHES). Yokohama, Japan; 2006. p. 270–284.

[5] Oswald E, Schramm K. An efficient masking scheme for AES software implementations. In: Springer, editor. Information Security Applications; 2006. p. 292–305.

[6] Standaert FX, Piret G, Gershenfeld N, et al. SEA: A scalable encryption algorithm for small embedded applications. In: International Conference on Smart Card Research and Advanced Applications. Springer; 2006. p. 222–236.

[7] Bogdanov A, Knudsen LR, Leander G, et al. PRESENT: An Ultra-Lightweight Block Cipher. In: Springer-Verlag, editor. Proceedings of the 9th International Workshop on Cryptographic Hardware and Embedded Systems (CHES). Vienna, Austria; 2007. p. 450–466.

[8] Wu W, Zhang L. LBlock: a lightweight block cipher. In: Springer, editor. Proc. of the 9th International Conference on Applied Cryptography and Network Security (ACNS). Nerja, Spain; 2011. p. 327–344.

[9] Wheeler DJ, Needham RM. TEA, a tiny encryption algorithm. In: Springer, editor. Proceedings of the 2nd International Workshop on Fast Software Encryption. Leuven, Belgium; 1995. p. 363–366.

[10] Veltri L, Cirani S, Busanelli S, et al. A novel batch-based group key management protocol applied to the Internet of Things. Ad Hoc Networks. 2013;11(8):2724–2737.

[11] Gorlatova M, Margolies R, Sarik J, et al. Prototyping Energy Harvesting Active Networked Tags (EnHANTs). In: Proc. of the 32nd IEEE International Conference on Computer Communications (INFOCOM) – mini conference. Turin, Italy; 2013. p. 585–589.

[12] Winter T, Thubert P, Brandt A, et al.. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. IETF; 2012. RFC 6550 (Proposed Standard). Available from: http://www.ietf.org/rfc/rfc6550.txt.

[13] Landman D. Arduino AESLib; 2011–2013. Available from: https://github.com/DavyLandman/AESLib.

[14] Dierks T, Rescorla E. (IETF) IETF, editor. RFC 5246 – The Transport Layer Security (TLS) Protocol Version 1.2; 2008. Available from: http://tools.ietf.org/html/rfc5246.

[15] Forte AG, Ferrari G. Towards distributing block ciphers computations. In: Wireless Communications and Networking Conference Workshops (WCNCW), 2015 IEEE. IEEE; 2015. p. 41–46.

[16] Alam F, Mehmood R, Katib I, et al. Data Fusion and IoT for Smart Ubiquitous Environments: A Survey. IEEE Access. 2017;5:9533–9554.

[17] Lu R, Heung K, Lashkari AH, et al. A Lightweight Privacy-Preserving Data Aggregation Scheme for Fog Computing-Enhanced IoT. IEEE Access. 2017;5:3302–3312.

[18] Stojmenovic I. Machine-to-Machine Communications With In-Network Data Aggregation, Processing, and Actuation for Large-Scale Cyber-Physical Systems. IEEE Internet of Things Journal. 2014 April;1(2):122–128.

[19] Luong NC, Hoang DT, Wang P, et al. Data Collection and Wireless Communication in Internet of Things (IoT) Using Economic Analysis and Pricing Models: A Survey. IEEE Communications Surveys Tutorials. 2016 Fourthquarter;18(4):2546–2590.

[20] Plageras AP, Psannis KE, Stergiou C, et al. Efficient IoT-based sensor BIG Data collection-processing and analysis in smart buildings. Future Generation Computer Systems. 2018;82:349 – 357.

[21] The Contiki Operating System;. Available from: http://www.contiki-os.org.

[22] Osterlind F, Dunkels A, Eriksson J, et al. Cross-Level Sensor Network Simulation with COOJA. In: Proceedings. 2006 31st IEEE Conference on Local Computer Networks; 2006. p. 641–648.