The Role of Energy in the Lightweight Cryptographic Profile

Conor Patrick and Patrick Schaumont

Bradley Department of Electrical and Computer Engineering,
Virginia Tech, Blacksburg VA 24061
conorpp@vt.edu, schaum@vt.edu

Abstract. NIST's Lightweight Cryptography Project ties selected cryptographic algorithms to a profile, which captures the physical, performance and security characteristics of the target devices that run these algorithms. This contribution investigates the role of energy in the profile. For many of the important application domains of Lightweight Cryptography, the operation is intermittent, rather than continuous. Examples include devices with a low duty cycle, devices running on an energy-harvested energy source, and RFID. In this context, energy consumption becomes as important as power dissipation. We analyze recently published lightweight cryptographic algorithms from the energy perspective. We highlight a unique opportunity that exists between energy harvesting and cryptography. The bulk of the operations in a cryptographic algorithm, such as the key schedule or key-stream generation, can be completed off-line when a secure nonvolatile memory is available. This decreases the latency and complexity of the online phase, and it spreads out the energy needs of the algorithm more evenly in time. We describe how such energy-harvesting friendly operation can be captured in the profile of the algorithm, and provide several examples of this concept to symmetric-key and public-key primitives.

Keywords: Lightweight Cryptography, Energy Balance, Pre-computing, Energy Harvesting

1 Introduction

Lightweight cryptography is characterized by constraints on the implementation characteristics of the cryptography, including the resource cost, the performance characteristics, and the security. The potential applications of a cipher are thus determined by this profile. The Lightweight Cryptography Project by NIST aims to establish a portfolio of cryptographic algorithms that are tied to these profiles [1]. Among the factors currently considered in a profile are gate count (GE) for hardware implementations, memory footprint for software implementations, latency and throughput, power dissipation, security level, attack models, and implementation attack resistance.

Our contribution considers the role of energy (Joule) in the profile of a cipher. Energy equals effort (J), whereas power equals the rate of effort (J/s). While energy and power are frequently mentioned together in publications, they reflect two very different properties of an implementation. Energy reflects a *resource requirement*, while power reflects a *performance characteristic*. For example, each encryption performed by a cipher requires some energy. A given battery, which holds a finite amount of energy, can only support a limited number of encryption operations. If one knows the energy per encryption, it's possible to predict the number of encryptions on a battery charge. Based on the power consumption of the cipher alone however, one cannot tell how long the battery will last. As a second example, an electronic designer can trade off throughput and power dissipation through clock frequency scaling. This transformation, however, has no first-order impact on the energy requirement.

Traditionally, computing systems have operated under a steady-state operation modus. They run algorithms continuously from an apparently infinite energy source – the wall plug. Under these assumptions, one would optimize algorithm implementations towards minimal power dissipation and maximal throughput.

In the world of lightweight cryptography, computers may also operate from batteries or energy harvesters. Batteries provide an *almost* infinite power-delivery capacity. Ebergy harvesters provide an infinite energy delivery capacity – if you're patient. In order to decide if a given battery or harvester configuration is adequate, it's important to understand the activity profile of the application. The steady-state operation modus should be extended with an on-off modus, where computing systems perform on-demand tasks and then switch to a low-power, energy-conserving mode during periods of inactivity. This concept applies equally well to cryptography, and perhaps especially well to cryptographic protocols. Think for example about one-time password tokens, or about sensor nodes that do an occasional challenge-response authentication, while staying dormant otherwise.

In this contribution we have three objectives.

1. We describe the principle of energy-balance to identify the main sources and sinks of the energy-flow in a lightweight cryptography application. Then, we make an assessment of the energy needs for lightweight cryptography, and we estimate the energy resource-cost of some lightweight primitives mentioned in the report by NIST [1].
2. We propose how energy could be captured in the profile of a lightweight cryptographic algorithm, and what algorithm features can be highlighted to promote the suitability of a lightweight cryptographic algorithm in energy-sensitive applications.

3. We explore the potential of pre-computed key-schedules and key-streams, offered by most cryptographic algorithms, to support lightweight crytography applications. Precomputation spreads out the computations of an algorithm over time, as small computation steps. We show that this leads to significant improvements in energy-efficiency and latency of the cryptographic algorithms, at the cost of extra nonvolatile storage. We confirm that these results apply to public-key operations as well, and summarize some earlier experiments with post-quantum signatures.

We are not the first to investigate a holistic view on the energy balance of cryptographic algorithms, though we believe we are the first to specifically consider the case of energy harvesting.

Several authors have analyzed the energy balance between wireless communications and cryptographic computations and they demonstrate the relatively high cost of wireless communications as opposed to cryptographic operations. de Meulenaer *et al.* demonstrate a symmetric-key based Kerberos protocol on a MICAz sensor node. A single Kerberos key exchange uses 96% of the energy for wireless communications, and only 4 % for symmetric key cryptography [2]. A similar experiment, by Singelee *et al.*, using an ISO 9798-2 authentication protocol based on AES, and a low-cost radio, concludes that 94% of the energy per authentication is used for wireless communications, and only 6% for cryptography [3]. However, when public-key cryptography is considered, the energy is much more evenly distributed, and the tendency is towards 50/50 between communication and computation. Similar data is presented by Struik [4].

Trappe *et al.* present a different view on the energy needs of cryptography within the generic context of the Internet of Things. They argue that energy-harvesting is incapable of delivering sufficient energy to power cryptographic solutions, and suggest to investigate alternative scenario's for security at the physical-layer level [5]. We confirm that the traditional, always-on model of cryptographic computations cannot be supported with the current generation of energy-harvesting technologies. However, we will show that the combination of cryptography and energy harvesting still presents an important opportunity. Therefore, we argue that the two mechanisms – physical layer security and lightweight cryptography – are both significant in the secure Internet of Things, and that neither is able to replace the other.

The opportunity to combine cryptography and energy harvesting has been identified in prior works. Pelissier *et al.* suggests using stream ciphers, and point out that for such algorithms, the key stream can be generated before the data is known or available [6]. They use this capability to generate

key streams during periods when energy is plentiful. The pre-computing technique was also proposed for public-key algorithms running on harvested energy, by Ateniese *et al.* for elliptic-curve signatures [7], and by Aysu *et al.* for hash-based signatures and lattice-based sigatures [8,9]. In this paper, we argue that the ability to express a cryptographic algorithm or mode of operation in small, atomic steps is a crucial benefit for pre-computation techniques.

The remainder of this paper is organized as follows. The next section clarifies the difference between power and energy, and argues that both are relevant for lightweight cryptography. In Section 3 we examine the energy needs of several lightweight crypto-algorithms in further detail, and examine the energy-balance of a typical application using lightweight cryptography. In Section 4 we discuss the pre-computing technique in further detail, and show how this leads to an overall increase in energy efficiency. In Section 5, we conclude by suggesting how energy can be integrated as a factor within a profile for Lightweight Cryptography.

2 Preliminaries

In this section we clarify the terminology and definitions used in this paper. The central topic of the paper is energy, which is a measure of effort and which is expressed in Joules. Power is a measure of rate of effort and is expressed in Joules/second or Watt. Cryptographic computations are iterations of a cryptographic kernel, using a mode of operation. We call the energy needed by a cryptographic device to complete a single iteration, the energy quantum (J).

Energy quanta are application dependent, and they can measure energy to complete the computations for signatures, ciphertext blocks, message digests for a fixed message length, and so on. Quanta are discrete – it's not helpful to compute only half a signature or only half of the number of rounds in a block cipher. The average power consumed by a cryptographic device equals the energy quantum divided by the time needed to complete the quantum. A high energy quantum does not imply a high power dissipation or vice versa. For example, complex algorithms may need a longer time to complete on a simple cryptographic device, so a very low power dissipation can still result in a high energy quantum [1].

With these definitions, we can describe the operation of a battery-powered system and of an energy-harvested system as shown in Figure 1. In

[1] We believe that for this reason, the profile of a lightweight cryptographic algorithm should mention power as well as energy. Latency in cycles, and power in Watt, is insufficient to derive the energy quantum when the device clock frequency is unknown.

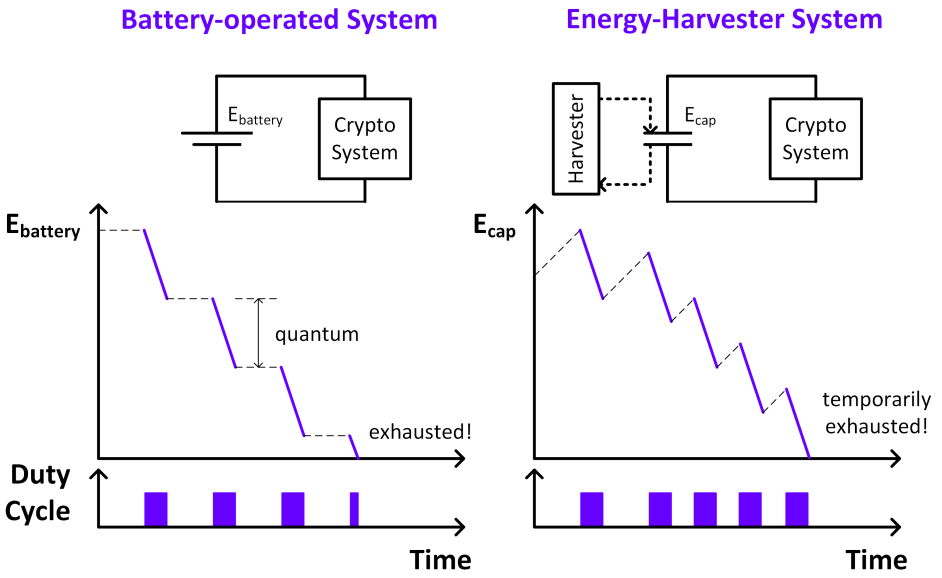

Fig. 1: *Energy Dissipation of a battery-powered (left) and an energy-harvester powered (right) system.*

a battery-operated design, the energy in a battery decreases monotonically as the cryptographic device completes quanta, and eventually the battery is exhausted. In an energy-harvester based system, the harvester will continuously recharge an energy store such as a supercap. Hence, the energy store will be recharged even as energy quanta are taken out. As long as the energy influx in the energy harvester is larger than the amount of energy quanta consumed, the lifetime of the energy-harvester cryptosystem is unlimited. A critical factor in determining this balance is the system duty-cycle, which is the relative on-time of the cryptosystem. The energy capacity of both the battery and the energy store of an energy harvester are limited. A higher store capacity increases the autonomy of the cryptosystem, but also its cost.

3 Energy Needs for Lightweight Cryptography

In this Section, we evaluate lightweight cryptographic designs from an energy perspective. Our objective is to assess the relative magnitude of energy capacities and energy needs for contemporary lightweight cryptographic algorithms. This helps to identify open challenges.

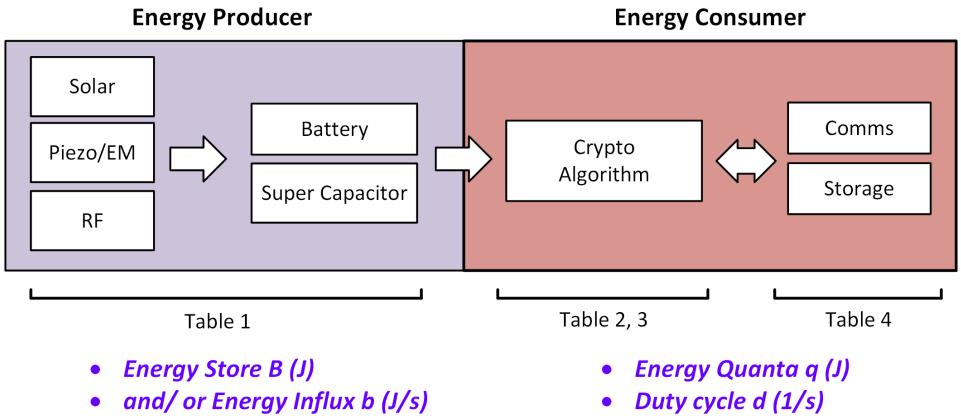

Fig. 2: *A lightweight cryptographic device as part of an embedded system. Overall lifetime assessments are made by considering the energy balance between the sources of energy and the consumers of energy.*

3.1 Energy Balance

Figure 2 illustrates the energy balancing problem faced by an engineer of a lightweight device. There are two design strategies: use a large enough battery that will fuel the device over an extended period; or use energy harvesting and carefully schedule energy consuming activities.

At the energy producer side, energy is held in a store of B Joules, and when an energy-harvester is used, that store is replenished at a rate of b Joules/second. At the energy consumer side, there's a lightweight cryptographic algorithm powered by energy quanta of q Joules. The data processed by the algorithm will either be forwarded to a communications unit or else stored in long-term memory or non-volatile memory. The duty cycle of the cryptographic device is d per second, reflecting the usage of this lightweight device.

Because this system must be in balance, there is a limit on the number of energy quanta utilized. With a fixed-size energy source, the amount of crypto-operations available is $\lfloor B/q \rfloor$. With an energy-harvester, the amount of crypto-operations may become infinite provided that there is a net positive energy balance, i.e. $b > (q.d)$. Otherwise, the duty cycle of the energy harvester will be bound by (b/q). Hence, the argument of energy balance is easy to make once the quantities B, b, q and d are known. In the remainder of this section, we provide some typical values for contemporary technologies.

Table 1: Capacity of energy sources.

Sources	Power	Energy
RF sources (per cm^2) [10]	$10^{-3}..10^{-1}$ μW	3.6..360 μJ*
Vibration Piezo/EM (per cm^3) [11]	$5.10^{-1}..2.10^{2}$ μW	1.8..720 mJ*
Optical (per cm^2) [12]	$5..5.10^{4}$ μW	$18..1.8.10^{5}$ mJ*
AAA battery (NiMH)		$3,500$ J
10F, 3V supercap		45 J

* Assumes one hour of energy harvesting

3.2 Energy Production

Table 1 illustrates typical energy budgets that are encountered for producers and consumers of energy. The first row collects three harvesting sources; the second row lists the energy capacities of energy stores. The table clearly indicates a wide variation in energy budget between battery-powered and energy-harvested systems. In the next Section, we investigate the energy balance of these sources against the needs of existing lightweight cryptosystems.

3.3 Energy Consumption

To properly design a secure lightweight device, we suggest to consider the following metric: Joules per byte. If a lightweight primitive implementation can be expressed in Joules per byte, then an engineer can decide if a primitive will meet the energy requirements. It can be applied in two distinct scenarios:

- **Battery powered**: The device has a limited energy supply and must last for at least N years. The engineer can consider the amount of information that must be processed in N years and multiply it by the Joules/byte factor to get the minimum capacity of the battery.
- **Energy harvester**: The device has unlimited energy spread over time. The engineer considers the smallest unit of time before a device must process a minimum amount of information. He multiplies the information by the Joules/byte factor and checks if that much energy can be harvested at a regular rate.

In Table 2, we provide a survey of the energy consumption for lightweight primitives implemented in software. These are the same implementations noted by NIST [1]. We approximated the Joules/byte metric $E_n = P.T/n$ for each primitive implementation by following the datasheet for each respective platform to determine the expected power dissipation P, determining the

Table 2: Energy consumption per byte for software implementations.

Software Implementations			
Primitive	**Type**	**Platform**	E_n **(nJ/byte)**
Chaskey fast [13]	MAC	ARM M0/STM32F030R8	21.4
Chaskey compact [13]	MAC	ARM M0/STM32F030R8	19.8
Speck 64 bit block [14]	block cipher	ATtiny45	214
Speck 128 bit block [14]	block cipher	ATtiny45	252
Simon 64 bit block [14]	block cipher	ATtiny45	394
Simon 128 bit block [14]	block cipher	ATtiny45	604
AES-128 fast [15]	block cipher	AT90USB162	1,031
AES-128 compact [15]	block cipher	AT90USB646	1,114
DESXL [16]	block cipher	ATmega128	8,830
PRESENT-80 [16]	block cipher	ATmega128	11,099
Lesamnta-LW [17]	hash	8 bit Renesas H8	14,948
D-QUARK [18]	hash	ATtiny45	39,919
PHOTON-160 [18]	hash	ATtiny45	43,560
SPONGENT-160 [18]	hash	ATtiny45	75,050

absolute latency (run-time) of the primitive (T), and by normalizing the resulting energy estimate over the block length or digest length n.

It's worth noting that DESXL and PRESENT do poorly in software implementations likely because they were designed for hardware implementations.

In Table 3, we provide a similar survey for hardware implementations. The energy consumption of a hardware module can be approximated as before by $E_n = P.T/n$. The power consumption P includes a static part and a dynamic part. In the following, we will focus on the dynamic portion of the power consumption. The dynamic power can be estimated by $P = GE.C.V^2.f/2$, with GE the number of gates, C average switched capacitance per gate per cycle, V the voltage supply and f the clock frequency. The average switched capacitance per gate depends on the technology (45nm, 90nm, etc) as well as on the activity of the design.

We tried to define an energy-metric that can be computed from the existing body of work in crypto-hardware designs, which typically mention gate count (GE), throughput T_p in cycles, and maximum clock-frequency (f_{max}). Using a technology constant $k_{tech} = C.V^2/2$ (J/gate), we can write $E_n = k_{tech}.GE.T_p/n$. In this expression, T_p is the latency of the algorithm in clock cycles, GE the number of equivalent gates, n the number of bytes in a block, and k_{tech} the technology constant. Table 3 lists this metric for various designs without the factor k_{tech}.

This table is only a first-order approximation which ignores static power dissipation. We verified that k_{tech} for 0.18 μm designs varies between $1.7.10^{-8}$ and $1.9.10^{-8}$ (J/gate). Note that PRESENT has a very good energy factor

Table 3: Relative energy consumption per byte for hardware implementations. k_{tech} is a technology factor in Joule/gate.

Hardware Implementations			
Primitive	**Type**	**Technology**	E_n/k_{tech}
PRESENT-80 [19]	block cipher	0.18 µm	236
DESL [20]	block cipher	0.18 µm	1,463
Simon 128 bit block [14]	block cipher	0.13 µm	1,725
Simon 64 bit block [14]	block cipher	0.13 µm	1,796
Speck 64 bit block [14]	block cipher	0.13 µm	2,450
Speck 128 bit block [14]	block cipher	0.13 µm	3,461
LED [21]	block cipher	0.18 µm	6,140
Photon-160 fast [22]	hash	0.18 µm	3,176
Spongent-160 fast [23]	hash	0.13 µm	3,695
Lesamnta-LW [17]	hash	90 nm	3,708
Spongent-128 fast [23]	hash	0.13 µm	4,428
Photon-160 compact [22]	hash	0.18 µm	15,511
D-QUARK [24]	hash	0.18 µm	22,493
Spongent-128 compact [23]	hash	0.13 µm	93,529
Spongent-160 compact [23]	hash	0.13 µm	99,675
TuLP-128 fast [25]	MAC	0.18 µm	5,772
TuLP-128 compact [25]	MAC	0.18 µm	8,577
Grain [26]	stream cipher	0.13 µm	514
Trivium [26]	stream cipher	0.13 µm	805

but its key is only 80 bits and does not meet the 112 bit key requirement from NIST [1].

Table 4 complements our hardware implementation approximations with an overview of results from [27,28]. Outlined are better estimates for Joule/B results from hardware implementations. Both [27] and [28] used different transistor technologies which shows their Joule/byte metrics are far off from each other. But because they used three common ciphers, AES, PRESENT, and PRINCE, we can see that they are consistent in the relative amount of energy consumed. PRESENT, SIMON, and LED are common and consistent with our approximations as well. Thus it is important to use relative metrics. Midori [27] should be taken note of because the cipher was designed to be energy efficient.

In Table 5, we provide representative energy data to capture the cost of storing ciphertext in non-volatile memory, or else transmitting it using a low-power wireless technology. As expected, non-volatile memory storage (NVM) is orders of magnitude cheaper than wireless transmission. On the other hand, the energy cost of a modern low-energy communication technology such as Bluetooth LE is in the same ballpark as the cryptographic computation cost. Therefore, when establishing energy quanta, the designer

Table 4: Absolute energy consumption per byte for block cipher hardware implementations from [27, 28].

Block Cipher Hardware Implementations	
STM 90 nm, 10 MHz [27]	
Block Cipher	E_n **(pJ/byte)**
Midori-128	11.7
PRINCE	18.1
NOEKEON	21.1
PRESENT	21.5
AES	21.9
SIMON 128/128	41.5
UMC 0.130 µm, 100 KHz [28]	
Block Cipher	E_n **(pJ/byte)**
KLEIN-parallel	105.9
PRINCE	170.4
PRESENT	189.5
LED	477.6
CLEFIA	566.2
KATAN-64	793.7
AES	389.0 - 2315.8

cannot ignore the cost of communication in addition to cryptographic computation.

Future work on lightweight crypto should consider providing a Joules/byte metric.

3.4 Achieving Balance

With the right metrics, an engineer can design a system multiple ways. For a given platform or algorithm requirement, the engineer can determine the energy needs and subsequently the energy balance. For a given energy constraint, the engineer can determine the suitable algorithms. In an appropriate energy balancing effort, one should always consider the complete platform including the sources of energy, the cryptography, and the data links that integrate the cryptography.

For example, if a system is constrained to use a AAA battery and must write results to FRAM, then we could afford to run Chaskey on 117 GB of data, Speck-128 on 9.5 GB of data, or D-QUARK on 59.5 MB of data. Now when we consider energy harvesting, we can assume the system can use a small battery and we now choose ciphers based on the rate of energy harvesting. If we use a $1cm^2$ solar cell and harvest an average of $200J$ per day, then we could afford to send roughly a 15th of what we could using a AAA battery *each day*. This would require encryptions be spread out over

Table 5: Energy consumption per byte for non-volatile memory and communications. We caution that this table only shows the energy needed for a typical scenario, and is not meant as a performance comparison between different technologies.

Technology	Energy nJ/byte
FRAM [29]	10 *
EEPROM [29]	13,210 *
916.5 MHz radio transmit [30]	10,080
2.4 GHz WirelessHART (IEC 62951) [4]	320
2.4 GHz ANT [31]	5,680
2.4 GHz Bluetooth LE [31]	1,224
IrDA [31]	385,600

* Under 3.3 supply voltage and continuous writes to 64KB memory size.

time. Piezo or RF based energy harvesters would likely not support software hashing, EEPROM, or radios but rather only support some kilobits of data per day using block ciphers or MACs and FRAM.

Our examples fill out Figure 2 with details from Tables 1, 5, 2, and 3. Ultimately some form of this is what secure lightweight design comes down to.

4 Balancing energy with pre-computing

Crypto algorithms achieve secrecy by mixing secret key material with plaintext, in order to achieve ciphertext. The key is usually preprocessed using various mechanisms, before it is combined with the plaintext. We will use the term *key-expansion* to describe all of the following key handling cases.

- The key may be expanded into *roundkeys* by means of a keyschedule. The schedule often looks like a simplified encryption round (eg. AES, PRESENT [19]), but some lightweight crypto proposals have simplified the key expansion to padding (eg. LED [21]) or even reuse (eg. PRINCE [32]).
- The key may be inserted inline with plaintext, such as in the case of computing a MAC or a sponge construction [33]. In this case, the key is absorbed into the state of the cipher through separate rounds.
- In the case of stream ciphers, the cryptographic algorithm itself is a fully specialized, dedicated key generation algorithm, and the encryption is reduced to a simple XOR operation.
- In the case of public-key cryptography, the key pair is generated separately in a compute-intensive specialized computational step.

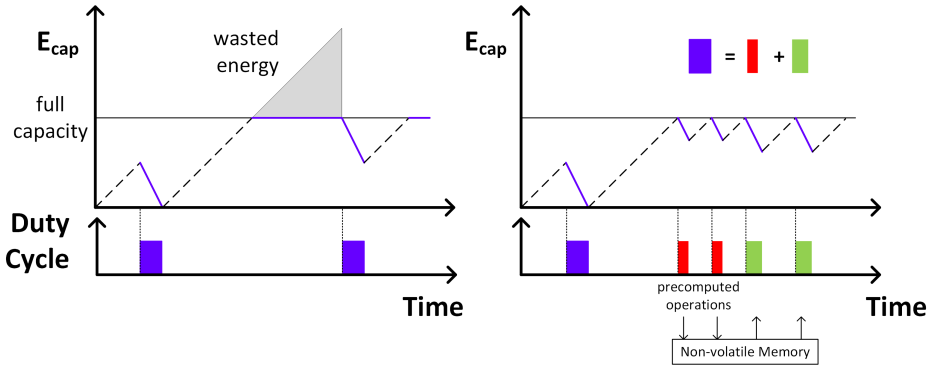

Fig. 3: Energy levels on the energy store of a harvester. Without pre-computing (left), energy harvested energy about the store capacity is wasted. With pre-computing (right), portions of the cryptographic algorithm are pre-computed and stored in non-volatile memory. The overall amount of energy quanta completed with pre-computing is higher than without, indicating that the pre-computed system has a higher energy efficiency.

The bottom line of these observations is that a significant portion of cryptographic computations do not depend on input data, but only on the knowledge of a secret key or a seed. This leads to two different key-expansion strategies: the *online* strategy, which processes the key when input data is available or when output data is required, and the *offline* strategy, which processes the key beforehand, and which stores intermediate values such as round-keys in a secure non-volatile memory. The offline scenario reduces the complexity of the online encryption phase, and it presents a unique opportunity for the energy-harvested scenario.

Figure 3 illustrates two scenarios of lightweight cipher use in combination with an energy harvester. The left side uses an online key-schedule, and will expand keys only when the actual crypto-operation is required. The right side uses an off-line key-schedule. It will initiate the key-schedule when the energy-store is full. This prevents wasting of harvested energy. The key-schedule part can potentially be executed by the device in a low-power operation mode, without real-time requirements. The generated keys are stored in a secure non-volatile memory. The key expansion process continues as long as there is excess energy, and as long as there is storage available to store expanded keys. When the actual encryption operation is required, the cipher uses a pre-generated set of roundkeys, and therefore will have lower latency and higher throughput as compared to the case with an online key-schedule.

The concept of precomputation is powerful, and a potential game-changer for cryptography in constrained context. We illustrate this with the following example. Assume an energy harvester that delivers 100 μW, the power available from a good piezo harvester. In an experimental setup with a wireless node, we measured a 91mJ requirement for a NIST P-256 ECDSA based authentication protocol [34]. Hence, an online scheme would need 910 seconds to collect the required energy to perform this authentication – clearly an unacceptable latency. Nevertheless, over the course of 24 hours, the same harvester is able to collect sufficient energy for 95 such authentications, which would be sufficient for many applications. With a suitable pre-computation scheme, ECDSA may be a feasible candidate in energy harvesting context. Several authors have made exactly this point for public-key cryptography [7, 9].

5 Acknowledgments

This research was supported in part by the NSF CyberCorps SFS Program.

6 Conclusion

In this paper we evaluated lightweight cryptography from an energy perspective. We observe that energy becomes important when one considers that a cryptographic device can have multiple power states (on and off, for example), and one wants to make assertions on the lifetime of the device.

We propose the metric Joules/byte as a measure of the energy requirement of a lightweight cryptographic primitive, and we suggest that this metric be added to the lightweight cryptographic profiles as a performance metric. There are two application scenario's that require knowledge of the Joules/byte metric. In the battery operated applications, energy available to the cryptography is limited by the capacity of the battery. Hence, the Joules/byte metric helps to determine the expected lifetime of the cryptographic device. Second, in energy-harvested applications, the power available to the cryptography is limited by the harvesting efficiency. In this case, the Joules/byte metric helps to establish an expected duty cycle for a lightweight cryptographic module.

We also point out an opportunity for cryptography in the context of energy harvesting. It's possible to convert all of the harvested energy to useful computations, by partitioning the cryptographic algorithm in small atomic steps. The likely candidate for such partitioning is in computing key schedules or keystreams. Preliminary results have shown great promise of this concept for public-key crypto, but the case of symmetric-key crypto

remains unexplored. In particular, for MAC and authenticated encryptions, efficient partitioning techniques are yet to be defined.

References

1. McKay, K., Bassham, L., Turan, M., Mouha, N.: Report on Lightweight Cryptography. NIST DRAFT NISTIR 8114 (2016) http://csrc.nist.gov/publications/drafts/nistir-8114/nistir_8114_draft.pdf.
2. de Meulenaer, G., Gosset, F., Standaert, F.X., Pereira, O.: On the Energy Cost of Communication and Cryptography in Wireless Sensor Networks. In: 2008 IEEE International Conference on Wireless and Mobile Computing, Networking and Communications. (2008) 580–585 http://dx.doi.org/10.1109/WiMob.2008.16.
3. Singelée, D., Seys, S., Batina, L., Verbauwhede, I.: The Energy Budget for Wireless Security: Extended Version. IACR Cryptology ePrint Archive **2015** (2015) 1029 http://eprint.iacr.org/2015/1029.
4. Struik, R.: Cryptography for Highly Constrained Networks. Workshop on Cryptography for Emerging Technologies and Applications (2011) http://csrc.nist.gov/groups/ST/CETA_2011/presentations/struik.pdf.
5. Trappe, W., Howard, R., Moore, R.S.: Low-Energy Security: Limits and Opportunities in the Internet of Things. IEEE Security Privacy **13** (2015) 14–21 http://dx.doi.org/10.1109/MSP.2015.7.
6. Pelissier, S., Prabhakar, T.V., Jamadagni, H.S., VenkateshaPrasad, R., Niemegeers, I.: Providing security in energy harvesting sensor networks. In: 2011 IEEE Consumer Communications and Networking Conference (CCNC). (2011) 452–456 http://dx.doi.org/10.1109/CCNC.2011.5766511.
7. Ateniese, G., Bianchi, G., Capossele, A., Petrioli, C.: Low-cost Standard Signatures in Wireless Sensor Networks: A Case for Reviving Pre-computation Techniques? In: 20th Annual Network and Distributed System Security Symposium, NDSS 2013, San Diego, California, USA, February 24-27, 2013. (2013)
8. Aysu, A., Schaumont, P.: Precomputation Methods for Hash-Based Signatures on Energy-Harvesting Platforms. IEEE Transactions on Computers **65** (2016) 2925–2931 http://dx.doi.org/10.1109/TC.2015.2500570.
9. Aysu, A., Yuce, B., Schaumont, P.: The Future of Real-Time Security: Latency-Optimized Lattice-Based Digital Signatures. ACM Trans. Embed. Comput. Syst. **14** (2015) 43:1–43:18 http://doi.acm.org/10.1145/2724714.
10. Raju, M., Grazier, M.: ULP meets energy harvesting: A game-changing combination for design engineers. Texas Instruments White Paper (2008) http://www.ti.com/lit/wp/slyy018a/slyy018a.pdf.
11. Mitcheson, P.D., Yeatman, E.M., Rao, G.K., Holmes, A.S., Green, T.C.: Energy Harvesting From Human and Machine Motion for Wireless Electronic Devices. Proceedings of the IEEE **96** (2008) 1457–1486 http://dx.doi.org/10.1109/JPROC.2008.927494.
12. Penella-López, M.T., Gasulla-Forner, M. In: Optical Energy Harvesting. Springer Netherlands, Dordrecht (2011) 81–123 http://dx.doi.org/10.1007/978-94-007-1573-8_5.
13. Mouha, N., Mennink, B., Herrewege, A.V., Watanabe, D., Preneel, B., Verbauwhede, I.: Chaskey: An Efficient MAC Algorithm for 32-bit Microcontrollers. Cryptology ePrint Archive, Report 2014/386 (2014) http://eprint.iacr.org/2014/386.
14. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK Families of Lightweight Block Ciphers. Cryptology ePrint Archive, Report 2013/404 (2013) http://eprint.iacr.org/2013/404.

15. Osvik, D.A., Bos, J.W., Stefan, D., Canright, D.: Fast Software AES Encryption. In: Fast Software Encryption: 17th International Workshop, FSE 2010, Seoul, Korea, February 7-10, 2010, Revised Selected Papers, Berlin, Heidelberg, Springer Berlin Heidelberg (2010) 75–93 http://dx.doi.org/10.1007/978-3-642-13858-4_5.
16. Eisenbarth, T., Kumar, S., Paar, C., Poschmann, A., Uhsadel, L.: A Survey of Lightweight-Cryptography Implementations. IEEE Des. Test **24** (2007) 522–533 http://dx.doi.org/10.1109/MDT.2007.178.
17. Hirose, S., Ideguchi, K., Kuwakado, H., Owada, T., Preneel, B., Yoshida, H. In: A Lightweight 256-Bit Hash Function for Hardware and Low-End Devices: Lesamnta-LW. Springer Berlin Heidelberg, Berlin, Heidelberg (2011) 151–168 http://dx.doi.org/10.1007/978-3-642-24209-0_10.
18. Balasch, J., Ege, B., Eisenbarth, T., Grard, B., Gong, Z., Gneysu, T., Heyse, S., Kerckhof, S., Koeune, F., Plos, T., Pppelmann, T., Regazzoni, F., Standaert, F.X., Assche, G.V., Keer, R.V., van Oldeneel tot Oldenzeel, L., von Maurich, I.: Compact Implementation and Performance Evaluation of Hash Functions in ATtiny Devices. Cryptology ePrint Archive, Report 2012/507 (2012) http://eprint.iacr.org/2012/507.
19. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C. In: PRESENT: An Ultra-Lightweight Block Cipher. Springer Berlin Heidelberg, Berlin, Heidelberg (2007) 450–466 http://dx.doi.org/10.1007/978-3-540-74735-2_31.
20. Leander, G., Paar, C., Poschmann, A., Schramm, K.: New Lightweight DES Variants. In: Fast Software Encryption: 14th International Workshop, FSE 2007, Luxembourg, Luxembourg, March 26-28, 2007, Revised Selected Papers, Berlin, Heidelberg, Springer Berlin Heidelberg (2007) 196–210 http://dx.doi.org/10.1007/978-3-540-74619-5_13.
21. Jian Guo, Thomas Peyrin, A.P., Robshaw, M.: The LED Block Cipher. Cryptology ePrint Archive, Report 2012/600 (2012) http://eprint.iacr.org/2012/600.
22. Guo, J., Peyrin, T., Poschmann, A.: The PHOTON Family of Lightweight Hash Functions. Cryptology ePrint Archive, Report 2011/609 (2011) http://eprint.iacr.org/2011/609.
23. Bogdanov, A., Knežević, M., Leander, G., Toz, D., Varıcı, K., Verbauwhede, I. In: spongent: A Lightweight Hash Function. Springer Berlin Heidelberg, Berlin, Heidelberg (2011) 312–325 http://dx.doi.org/10.1007/978-3-642-23951-9_21.
24. Aumasson, J.P., Henzen, L., Meier, W., Naya-Plasencia, M.: Quark: A Lightweight Hash. Journal of Cryptology **26** (2013) 313–339 http://dx.doi.org/10.1007/s00145-012-9125-6.
25. Gong, Z., Hartel, P., Nikova, S., Tang, S.H., Zhu, B.: TuLP: A Family of Lightweight Message Authentication Codes for Body Sensor Networks . Journal of Computer Science and Technology **29** (2014) 53–68 http://dx.doi.org/10.1007/s11390-013-1411-8.
26. Good, T., Chelton, W., Benaissa, M.: Review of stream cipher candidates from a low resource hardware perspective. SASC 2006 Stream Ciphers Revisited **125** (2006)
27. Banik, S., Bogdanov, A., Isobe, T., Shibutani, K., Hiwatari, H., Akishita, T., Regazzoni, F.: Midori: A block cipher for low energy (extended version). Cryptology ePrint Archive, Report 2015/1142 (2015) http://eprint.iacr.org/2015/1142.
28. Batina, L., Das, A., Ege, B., Kavun, E.B., Mentens, N., Paar, C., Verbauwhede, I., Yalcin, T.: Dietary recommendations for lightweight block ciphers: Power, energy and area analysis of recently developed architectures. Cryptology ePrint Archive, Report 2013/753 (2013) http://eprint.iacr.org/2013/753.

29. Medu, M.: Energy Comparison of Cypress F-RAM and EEPROM. Cypress White Paper (2014) http://www.cypress.com/file/46746/download.
30. Hill, J.L., Culler, D.E.: Mica: A Wireless Platform for Deeply Embedded Networks. IEEE Micro **22** (2002) 12–24
31. Smith, P.: Comparing Low-Power Wireless Technologies. CSR PLC Whitepaper (2011) http://www.digikey.com/en/articles/techzone/2011/aug/comparing-low-power-wireless-technologies.
32. Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E.B., Knezevic, M., Knudsen, L.R., Leander, G., Nikov, V., Paar, C., Rechberger, C., Rombouts, P., Thomsen, S.S., Yalçin, T.: PRINCE - A low-latency block cipher for pervasive computing applications (full version). IACR Cryptology ePrint Archive **2012** (2012) 529 http://eprint.iacr.org/2012/529.
33. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: Sponge functions. Ecrypt Hash Workshop 2007 (2007)
34. Schaumont, P., Yuce, B., Pabbuleti, K., Mane, D.: Secure authentication with energy-harvesting: A multi-dimensional balancing act . Sustainable Computing: Informatics and Systems (2015) http://dx.doi.org/10.1016/j.suscom.2015.10.002.