Cryptography on a Speck of Dust*

Jens-Peter Kaps†, Gunnar Gaubatz, Berk Sunar
Worcester Polytechnic Institute

Abstract

Ubiquitous computing has become a reality in recent years. Tiny wireless sensors and RFID tags are being deployed today and will soon form an important aspect of our infrastructure. Security is a critical factor for these ultra-low power devices due to their impact on privacy, trust and control. Traditional cryptographic algorithms are considered too bulky, complex and power hungry for this application. This paper gives an overview of cryptographic primitives present in classic and emerging algorithms and analyzes their suitability for ultra-low power implementations. This analysis is supported with data obtained from our own example implementations of ultra-low power primitives of block ciphers, hash functions and even public key algorithms. Finally we use our findings to provide designers of cryptographic algorithms with guidelines for developing new power efficient schemes to enable cryptography on a speck of dust.

1 Introduction

Wireless sensor networks (WSN) [1] and Radio Frequency Identification Devices (RFIDs) belong to a new set of ultra-low power applications which make computing more pervasive. They have wide ranging applications from supply chain management, to home automation, health care and military purposes. WSN and RFIDs are quickly becoming a vital part of our infrastructure. Security is a critical factor in these applications. A good introduction to the security issues that WSN face can be found in [2]. However, these tiny, pervasive computing devices have extremely limited power resources and computational capabilities. Security engineers face the seemingly contradictory challenge of providing lightweight cryptographic algorithms for strong authentication, encryption and other cryptographic services that can perform on a speck of dust.

Current wireless sensor nodes use simple, battery powered 4-bit or 8-bit general purpose processors. They provide for secure communication using cryptographic protocols like SPINS [3], which are implemented in software. We envision that the next generation sensor nodes will operate without batteries. Instead they will harvest energy from ambient sources in their environment. The notion of removing the battery and having self-powered computing devices opens the door to a wealth of new applications, not just for wireless sensor nodes but for the whole field of ubiquitous computing. Computing devices can then be embedded in walls, in concrete, placed in inaccessible locations or deployed in hostile territory by dropping them off aircrafts. Energy scavengers are devices that harvest energy from environmental sources like light, heat, and vibration and convert it into electric power. Micro-Electro-Mechanical Systems (MEMS) based power scavengers have

*This is a pre-print, only the affiliations have been updated in 2007

†Jens-Peter Kaps is now with George Mason University

been integrated on chip and are currently capable of producing up to 8 μW of power [4]. It is conceivable that future MEMS-based scavengers will be able to deliver up to 50 μW continously.

Most RFID tags contain a transponder and a read-only memory chip that is given a unique electronic product code, also called Global Unique Identifier (GUID). More sophisticated tags have writeable memory and some even sensors, blurring the line with sensor nodes. We distinguish between active and common passive tags. Active tags carry a small battery while passive tags receive their power from the reader. The reader emits an electric field to power the tags while querying them. The amount of power a tag receives depends on the field's intensity which is governed by national and international regulations. Only about 20 μW are available for the digital part of an RFID tag.

The power available to sensor nodes and RFID tags is orders of magnitude less than what battery powered devices consume. Therefore, designing cryptographic systems for these power constrained devices is especially difficult. Scavengers and RFID reader fields do not produce enough energy to support even the simplest general purpose low power CPUs which are currently being used in Sensor Networks. This defines the ultra-low power application domain.

In order to provide cryptographic functions for this class of devices, their power consumption has to be made the first design priority. So far research has concentrated on the network specific aspects of WSN and on software implementations of cryptographic algorithms. Only recently studies were published on special hardware implementations [5, 6, 7, 8]. The main power consumer of a wireless sensor node is its RF-transceiver or radio for short. In the past years specialized hardware has been developed for radios which combine low-data rate, low-power consumption, and the ability to interface directly with low-power micro controllers. In this paper we propose that in order to provide adequate security for ultra-low power applications like WSN and RFID, one has to employ a similar approach as for the radios, i.e. specialized hardware in conjunction with application specific algorithms. Other factors also play a role. In spite of the advances in radio design, transmission power is still very costly compared to computations. Hence, the transmission overhead incurred by applying security has to be kept to an absolute minimum. Furthermore, most cryptographic algorithms currently in use are designed for high performance, mostly in software on 32-bit microprocessors. On sensor nodes and RFIDs the computation time is not very critical. It is more important to consume little power and space. To this end, we introduce useful techniques for implementors of cryptographic algorithms as well as guidelines for cryptographers to design new cryptographic algorithms specifically for ultra-low power applications.

2 Survey of Cryptographic Algorithms

As a first step we examine current cryptographic algorithms. We limit ourselves to current popular algorithms and also include some that we found to be particularly interesting for this purpose. The interested reader can find a good introduction to cryptography and descriptions of many of these algorithms in [9] and other books.

2.1 Block Ciphers

We start our discussion by identifying common functions and structures that are shared by a wide variety of established block cipher designs. Our list of algorithms consists of classic block ciphers (DES/3DES, IDEA, RC5), the AES finalists, and the extended tiny encryption algorithm XTEA.

Table 1 contains a summary for the ciphers under consideration.

	No. of Rounds	Block Size (bits)	Feistel	SP-Network	Arithmetic S-Box	Pseudorandom S-Box	Fixed Permutations	Addition (mod 2^w)	Fixed Shift/Rotation	Var. Shift/Rotation	Mod. Multiplication	Const. Multiplication
DES/3DES	16/48	64	•			•	•		•			
IDEA	8	64	•					•			•	
RC5	12/16	32/128	•					•		•		
AES (Rijndael)	10/12/14	128		•	•		•		•			•
RC6	20	128	•					•		•	•	
MARS	2×16	128	•			•		•	•	•	•	
Serpent	32	128		•		•	•		•			
Twofish	16	128	•			•		•	•		•	•
XTEA	≥ 32	32	•					•	•			

Table 1: Common Elements in Block Ciphers

Round Structure Virtually all modern block ciphers are iterated product ciphers, i.e. the encryption process consists of repeated application of a round function. The round function is composed of multiple layers of transformations that perform substitution and permutation, more generally also called confusion and diffusion layers. In and by itself the round function is not considered secure, but each additional round adds to the security level.

Popular round structures are variations of the Feistel network (DES, RC5, MARS, etc.) in which the round function typically only modifies part of the round data. Therefore, some publications refer to rounds in Feistel ciphers as half-rounds. Substitution and Permutation Networks (SPN) on the other hand typically modify the entire dataset in each round, examples are IDEA, Rijndael (AES), etc.

Substitution Functions All product ciphers employ substitution functions, so called S-Boxes, in one form or another for introducing non-linearity into the encryption process. Various techniques range from look-up table based pseudo-random substitutions to non-linear arithmetic functions of high degree.

Permutation Product ciphers combine various transformations for confusion and diffusion. The latter is achieved either through fixed permutations (e.g. IP/IP^{-1} and P-Box in DES) or through data dependent (variable) shifts and rotations (RC5/6 and MARS).

Key Mixing Almost all block ciphers add subkeys into the round data using XOR operations, which are fast and introduce virtually no overhead in both software and hardware implementations.

Several algorithms also use a blend of XORs and regular integer addition. This has the effect that the resulting addition is no longer commutative, thereby complicating cryptanalysis.

Arithmetic Operations Certain arithmetic operations are useful for combining diffusion and non-linear mixing of round data with key bits. A good example is truncated integer multiplication which is used in MARS.

2.2 Stream Ciphers

Stream ciphers can be built in two different ways. The predominant method is to use a pseudo random number generator (PRNG) to generate a key stream and XOR its output with the datastream. The other possibility is to use a dedicated stream cipher like RC4.

The RC4 is a proprietary stream cipher. It uses integer addition modulo 256 and employs a dynamic S-Box with 256 8-bit entries. This S-Box is a lookup table where the entries change with each encrypted byte depending on the key.

PRNGs can be built from block ciphers, hash functions, modular exponentiators, or linear feedback shift register (LFSR) based stop-and-go generators. The first three have a big advantage over a dedicateds tream cipher, namely, the base function can also be used for its original purpose. The stream cipher functionality comes at minimal extra cost.

2.3 Hash Functions

A hash function produces a short fixed size digest of a long message. This digest can be used to check the integrity of a message. Hash functions that employ a secret key are called message authentication codes (MAC) and hence provide message authentication as well as integrity. Universal hash function families provide provable security and can be used to build provable secure MACs [10], i.e. bounds on the success probability of an attacker independent of the computational power applied can be proven. For this article we selected some of the most popular hash functions and additionally two universal hash function families: NH [11] and WH [8]. Table 2 summarizes some hash function parameters.

Each hash function has a fixed input size. Longer input strings are split, shorter ones are padded. The hash size specifies the length of the resulting hash value, independent of the length of the input string. The hash functions MD4, MD5, and those described in the Secure Hash Standard (SHS) SHA-1, SHA-256, etc. all belong to the same group of hash algorithms. MD2, MD4, MD5, and SHA-1 share similar functional blocks with minor differences in the parameters. In spite of recent advances in attacking SHA-1, it is still in wide spread use, however MD4 and MD5 are considered compromised.

NH was introduced as a new hash function family for an authentication code called UMAC and is based on modular integer multiplication and summation. WH is a hash function family with even stronger security properties than NH and can be used in place of NH. It is specifically designed for implementation in ultra-low power hardware and is based on modular polynomial multiplication and summation. NH and WH are defined for any fixed block size. For our table we assume a block size of 64 bits as this was also the size chosen in [8] for the implementation.

	No. of Rounds	Input Size(bits)	Hash Size(bits)	Constants (bytes)	Variables (bytes)	Integer Multiplication (bits)	Integer Addition (bits)	Polynomial Multiplication	Simple Logic Functions	Fixed Shift/Rotation
MD2	16	128	128	256	48		32		•	•
SHA-1	4	512	160	16	20		32		•	•
MD4	3	512	128	16	16		32		•	•
MD5	4	512	128	16	16		32		•	•
NH	n/a	64	64			64	64			
WH	n/a	64	64					•	•	

Table 2: Summary of Hash Function Characteristics

2.4 Public Key Cryptosystems

The security of classic public key algorithms typically relies on the hard problem of integer factorization or finding the discrete logarithm (DL) in a finite field (RSA, ElGamal, etc.). The dominating arithmetic operation of these algorithms is modular exponentiation, which is typically implemented using modular multiplication and squaring operations. A variant of RSA is Rabin's Scheme, in which the public key exponent is fixed to the value 2. This significantly reduces the complexity of the encryption operation to modular squaring and allows for a compact implementation, whereas the general case of RSA would be too large. The security of Rabin's Scheme relies on the integer factorization problem just like RSA. Therefore, there is no compromise in terms of security.

Elliptic curve cryptography (ECC) uses a variant of the DL problem that is defined over the additive group of points on an elliptic curve. Points are repeatedly added (or doubled) until a scalar 'multiple' of the originating point is obtained. A single point "addition" consists of a heterogeneous variety of finite field operations such as addition/subtraction, multiplication and in some cases inversion.

Hyperelliptic curve cryptography (HECC) has been proposed as a generalization of ECC. Its advantage is that operands can be even shorter than for ECC while an equivalent level of security is maintained. The problem of HECC is its arithmetic structure which is even more complex than that of ECC. While efficient explicit expressions exist for the group operations [12], they still contain diverse arithmetic primitives that may prove to be too complex for ultra-low power implementations.

NtruEncrypt is based on a completely different hard problem namely the Shortest Vector Problem (SVP) in high dimension lattices. The central arithmetic primitive of NtruEncrypt is multiplication in a truncated polynomial ring. The operation itself can be subdivided into individual computations of polynomial coefficients through accumulation of partial products. Table 3 compares the algorithm parameters of Rabin's Scheme, Ntru and Elliptic Curve of equivalent security

levels.

	Encryption	Signature	Message Payload (bits)	Ciphertext (bits)	Signature Length (bits)	Integer Multiplication (bits)	EC Point Addition (bits)	Polynomial Coefficients (bits)
Rabin's Scheme	•	•	< 512	512	512	512		
NtruEncrypt	•		< 265	1,169				8
NtruSign		•			1,169			8
EC-MV	•		< 200	400			169	
EC-DSA		•			200		169	

Table 3: Comparison of PKC Functions

3 Analysis

In this section we analyze how the structure, functional primitives and storage requirements of cryptographic algorithms relate to their energy consumption. From this we can devise recommendations for future algorithms tailored toward ultra-low power implementations. For our example implementations, we used the TSMC 0.13μm ASIC library, which is characterized for power, and the Synopsys tools Design Compiler and Power Compiler for synthesis. Modelsim was used for simulation and capturing of switching activity which was used by Power Compiler for power estimation. We noticed that at a clock frequency of 500 kHz, which is commonly found in sensor nodes, the static power consumption P_{Leak}, caused by leakage, outweighs the dynamic power consumption P_{Dyn}, caused by switching activity.

3.1 Algorithm Structure

The algorithm structure can tell us how well each algorithm lends itself to both parallelization and serialization. The latter ties in directly with minimization of circuit area and therefore static power consumption.

Scalability refers to the possibility of scaling an algorithm between bit serial and highly parallelized realizations in an efficient manner. In certain contexts, such as Public Key Cryptography, scalability may also encompass the re-use of existing processing elements for higher precision operands than originally intended, i.e. using a 1024 bit modular exponentiator hardware for 2048 bit operands.

The iterative round structure of most block ciphers is by itself an indicator for a reasonable degree of scalability, provided that all rounds are the same. Then only one instance of the round

function needs to be implemented. Serialization is possible with RC5 and RC6 and, ignoring a slightly modified last round, also for most other ciphers.

In contrast to block ciphers, public key schemes are for the most part based on arithmetic over large integer or polynomial fields, which usually lends itself well to serialization, even though certain operations like modular reduction introduce additional complexity which hinders serialization beyond a certain point. The biggest problem with serial implementations is the running time that is cubic in the operand size for most public key algorithms. Insofar it is important to evaluate the trade-off between area usage and a certain degree of parallelization.

Modularity is closely related to the concept of scalability, but more in the sense that simple processing elements can be replicated easily for further parallelization of a task when higher performance is necessary. We will give our implementation of the NtruEncrypt algorithm as an example [6]. NtruEncrypt has good modularity. Its basic operation can be subdivided into computations using 8-bit long polynomial coefficients. By intelligently arranging memory accesses to these coefficients, it is possible to perform computation of multiple coefficients of the result in parallel. Since storage of operands is by far the largest portion of the circuit, scaling up the number of parallel arithmetic units (AUs) has little effect ($< 50\%$ increase) on the overall area and power consumption. At the same time the number of clock cycles can be reduced dramatically, as can be seen from Table 4a.

Regularity describes the degree of similarity between modules at different levels of parallelization. At the logic level highly regular designs allow for efficient parameterization and reuse, while irregular circuits often require manual design changes. At the algorithmic level a high degree of regularity expresses the uniformity of operations necessary to perform a task, while very complex tasks consisting of many different atomic operations are characterized by their irregularity. Rabin's Scheme, NH and WH are good examples for algorithms that have high regularity. They all have one simple underlying function. In block ciphers even a single round of a block cipher can take up considerable amounts of chip area. Serialization of the round function is therefore necessary. Ciphers with a homogeneous round function (e.g. AES) have a high degree of regularity and therefore seem better suited for serialization than others of heterogeneous structure (e.g. DES).

Power - Energy Tradeoff Energy equals the amount of power dissipated over time. Increasing the degree of parallelism increases the power consumption, but at the same time decreases the computation time. The trade-off depends on the overall structure of the architecture, since certain elements may have constant size. The point of optimality can be found by modeling the energy consumption as a function of the degree of parallelism. A useful metric for this is Energy per Bit Encrypted. It describes the amount of energy necessary to encrypt a single bit of the message. This metric can be used to compare the energy efficiency of cryptosystems at an equivalent level of security. It is independent of the actual operand length.

3.2 Functional Primitives

In the following we describe the characteristics of each group of primitives we encountered in Section 2 and their suitability for ultra-low power implementation.

Simple Logic Functions This group contains logic functions where the output is dependent on only a small, fixed set of inputs. This includes functions like XORs of two bitstrings (AddRoundKey in AES), bit-multipliers and multiplexers. The number of logic gates scales linearly with the width of the data path.

Fixed Shifts and Permutations Fixed in this context means that the shifts and permutations are not data dependent. Permutations and expansions are used in block ciphers (DES, Serpent, AES) as non-linear diffusion elements. Fixed shifts and rotations serve the same purpose and are frequently used in block ciphers (AES, Mars, Serpent, Twofish) and hash functions (MD2, SHA-1, MD4, MD5). Common to all these functions is that their implementation introduces virtually no cost as they require only wiring resources and no logic. They are perfect for any hardware implementation.

Data Dependent Shifts Also called "variable shifts" for short, are used in a couple of block ciphers (RC5, RC6, Mars) due to their resistance against differential cryptanalysis. Implementations frequently use barrel shifters to support all possible shifts or rotations. The delay for one shift operation is proportional to $\log_2 n$ but its area scales with $n \log_2 n$. For situations in which the shift or rotation is followed by a register, it may be more power efficient to instead implement the register as a shift register with parallel load and combine it with some additional control logic and a counter. Due to the relatively high area cost variable shifts are not well suited for ultra-low power implementations unless they can be combined with existing registers.

Integer Arithmetic Integer arithmetic primitives such as addition and multiplication are frequently the most costly functions in a cryptographic algorithm. Although they can often be implemented in a bit serial fashion (e.g. multiplication), the efficient propagation of carries presents itself as a major problem. The simplest form of an adder, the carry propagate or ripple carry adder, scales linearly with the word size n, but glitches in the carry chain cause high dynamic power consumption. Various alternatives exist in the literature, but they go along with a penalty in terms of area and therefore static power consumption. Because of these costs integer arithmetic should be avoided for new ultra-low power algorithms.

Arithmetic primitives are frequently combined with modular reduction steps. In trivial cases the modulus is chosen as 2^k, which means that only k bits of the result are kept, excess bits are truncated. Finite field arithmetic with a non-trivial modulus adds a fair amount of complexity to the circuit. Simple implementations perform conditional subtractions of the modulus from the result, depending on the value of its most significant bit. For certain classes of public key algorithms which heavily depend on modular arithmetic, the use of residue number system arithmetic has proven to be effective for efficient implementation.

Polynomial Arithmetic Polynomial arithmetic, i.e. arithmetic in extension fields, is preferable for ultra-low power implementation due to limited carry propagation and improved regularity. Therefore, several algorithms are specifically tailored towards arithmetic in $GF(2^k)$ (e.g. AES). Additions in fields of characteristic two can simply be implemented by an XOR. For ultra-low power applications multiplication may be implemented in a bit-serial fashion. This is facilitated by the simplicity of the addition and reduction steps.

The vast difference in power consumption between integer and polynomial arithmetic is demonstrated in [8], where we describe the universal hash function families NH and PH. PH is a redefinition of NH which uses polynomials over $GF(2)$ instead of integers. We would like to emphasize that both hash functions are 2^{-w}-almost universal, hence both provide the same level of security. The differences in area, speed and power consumption, however, are impressive. The results of our implementation at 500 kHz are summarized in Table 4b.

Substitution Functions (S-Box) can be implemented using various techniques. While look-up tables are fast and easy to implement, the size of the tables is often prohibitively expensive. If performance is secondary to low power consumption and an arithmetic description for the S-Box exists, then a circuit realization of the underlying arithmetic operation may be preferable. We performed a case study on different AES S-box architectures for ultra-low power implementations. For one circuit we implemented the S-box as an arithmetic function using its inherent algebraic structure. For our other circuit we implemented a 256×8-bit look-up table in combinational logic. The results are summarized in Table 4c. Even though the combinational implementation uses only 30% of the dynamic power of the arithmetic implementation, due to its size its total power consumption is two times higher. This shows that it is advantageous if the content of the S-box can be described algebraically, which additionally gives an opportunity for serialization.

	Implementation	**Power (μW)**			**Area**[1]	**Delay**	**Clock**	**PDP**
		$\mathbf{P_{Dyn}}$	$\mathbf{P_{Leak}}$	**Total**		**ns**	**Cycles**[2]	**ns**$\times\mu$**W**
a)	NtruEncrypt: 1 AU	4.03	15.1	19.1	2,850	0.69	29,225	13.18
	NtruEncrypt: 8 AUs	5.00	22.5	27.5	3,950	0.69	3,682	18.96
b)	NH (integer)	5.47	28.1	33.6	5,291	9.92	64	333.31
	PH (polynomial)	3.41	12.1	15.5	2,356	1.35	64	20.93
c)	AES S-Box: Logic	0.42	7.67	8.10	1,397	1.61	1	13.04
	AES S-Box: Algebraic	1.39	2.68	4.07	431	4.68	1	19.05

Table 4: Implementation Comparisons (0.13μ ASIC library, 500 kHz)

3.3 Storage Requirements

Storage requirements of cryptographic algorithms are manifold. All constants and variables used by an algorithm as well as implementation specific storage elements add to it. Constants are comprised of fixed setup parameters, precomputed constants, and static S-Boxes. Fixed parameters and precomputed constants can be implemented in combinational logic. Strategies for larger sets of constants i.e. S-Boxes are described above in Section 3.2.

Variables, as well as variable S-Boxes (RC4) and temporary data has to be stored in registers or RAM. Pipelining techniques require additional storage elements. Since storage elements typically impose significant area and power penalties, they should be used conservatively in ultra-low power implementations.

[1] Area is given in terms of equivalent two input NAND gates

[2] Number of clock cycles to complete one operation

3.4 Implementation Considerations

Here we want to mention additional considerations that go beyond looking at the structure and elementary functions of a cryptographic algorithm.

Multi-encryption and Multi-hashing Multi-encryption /-hashing are two related concepts for increasing the security of an algorithm by applying it repeatedly. The Triple Data Encryption Algorithm (TDEA) also known as Triple DES is probably the best known example of multi-encryption. It applies DES three times in a row, using either two or three different keys depending on the keying option. It was originally developed out of the need to prolong the lifetime of DES until a new, more secure standard was found. However, in the light of ultra-low power cryptography, multi-encryption and multi-hashing can be seen as an enabling technology. It makes it possible to use block ciphers or hash functions that consume very little power but have a small security margin, and run them several times in series, thus obtaining a more secure overall cipher or hash function.

Fixed or Constant Parameters in cryptosystems can help to alleviate the problem of large storage requirements, and even simplify certain computations. This is highly dependent on the intended application context. For example, an Internet server will typically have to change keys and associated key parameters frequently, such that keeping them constant is not possible. In embedded applications, where communication is typically limited to links between sensor nodes and a base station, fixing parameters such as the public key helps to reduce the storage requirements significantly.

Precomputation is a powerful method for solving latency problems and is especially important for low-power nodes where intensive computations must be spread over time to reduce the power consumption below the maximum tolerable level. If the algorithm allows precomputation of intermediate results and thus only a small number of computations are necessary for the processing of the data, then latency may be virtually eliminated.

4 Recommendations for Designing new Algorithms

In this section we outline recommendations for designers from an implementers point of view. Note that designing cryptographic algorithms requires a unique skill-set and many years of experience and therefore should be attempted only by professional cryptographers. We use the terminology defined in the previous section to describe the key features that a new algorithm should have. The goal is to obtain an ultra-low power, scalable, secure algorithm.

The most important requirement is scalability. It should be possible to scale the algorithm from a bit serial implementation to a highly parallel implementation depending on the desired maximum power consumption and speed. The extent to which an algorithm is scalable depends on its regularity. New cryptographic algorithms should be regular and contain only a limited amount of different primitives. In order to further improve the scalability, the basic functions themselves should be serializeable. Then the implementor can trade off speed with power on a fine level of granularity and not just on the algorithmic level (e.g. round function).

Serializing an algorithm slows down its operation assuming the clock speed is held constant. However, in environments where data has to be sent quickly but only once in a while, it would be desirable if most steps could be computed "offline" ahead of time. When the data becomes available only a simple, fast computation should be required to complete the operation (e.g. addition of data to the key).

Applications for WSN and RFIDs have a variety of security requirements ranging from high risk applications e.g. military target tracking, where the devices are likely to being attacked to low risk applications e.g. passive environmental monitoring. This would be best supported if the new algorithms could operate with various key lengths (e.g. AES). Another method to increase the security level without increasing its footprint is multi-hashing / multi-encryption.

We would like to briefly summarize the implementation considerations for elementary functions.

- Lookup tables are costly. An algebraic representation can be more efficient.
- Polynomial arithmetic in $GF(2^k)$ is well suited for hardware implementation.
- Integer arithmetic has an inherent high power consumption.
- Data dependent shifts and rotations are costly unless they can be combined with existing registers.
- Fixed shifts and rotations are well suited for hardware implementation.

Messages in WSN and used by RFIDs are usually very small and average between 30 bits to 100 bits in length. The power consumed by transmitting a bit is high compared to the power consumed by computation. A new algorithm should therefore have a compact representation of cipher I/O. Encryption functions should not cause any message expansion and use a small block size. Hash functions should result in small digests as they are transmitted in addition to the original data. Ideally, the digest size should not affect the collision probability, i.e. security.

The challenge of future research is to find an algorithm that at its core contains a a simple, scalable primitive. It would be highly desirable if this simple primitive can be used as a common element for secret and public key functions. This would make it possible to provide both types of functions for ultra-low power applications, which in turn will enable simpler and efficient security protocols.

5 Conclusion

Current wireless sensor nodes and RFID tags struggle with the load of cryptographic algorithms implemented in software. In this paper we provide guidelines on how to implement current cryptographic algorithms in hardware to enable sufficiently strong cryptography for these devices. The next generation nodes will be MEMS powered and therefore their power constraints will be even more severe. Future RFID tags will incorporate more functionality like writeable memory and sensors. This will further decrease the amount of power available for cryptography. To address this issue we presented recommendations targeted to cryptographers for new ultra-low power algorithms. This paper gives an insight into the challenges that lie ahead for the cryptographic community to enable cryptography on a speck of dust and shows the first steps in this direction.

References

[1] D. Culler, D. Estrin, and M. Srivastava, "Guest editors' introduction: Overview of sensor networks," *Computer*, vol. 37, no. 8, pp. 41–79, Aug. 2004.

[2] A. Perrig, J. Stankovic, and D. Wagner, "Security in wireless sensor networks," *Commun. ACM*, vol. 47, no. 6, pp. 53–57, Jun. 2004.

[3] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "SPINS: security protocols for sensor networks," *Wireless Networks*, vol. 8, no. 5, pp. 521–534, Sep. 2002.

[4] S. Meininger, J. Mur-Miranda, R. Amirtharajah, A. Chandrakasan, and J. Lang, "Vibration-to-electric energy conversion," *IEEE Trans. VLSI Systems*, vol. 9, no. 1, pp. 64–76, Feb. 2001.

[5] M. Feldhofer, S. Dominikus, and J. Wolkerstorfer, "Strong authentication for RFID systems using the AES algorithm," in *CHES 2004*, LNCS 3156. Springer, Aug. 2004, pp. 357–370.

[6] G. Gaubatz, J.-P. Kaps, and B. Sunar, "Public key cryptography in sensor networks—revisited," in *1st European Workshop Security in Ad-Hoc and Sensor Networks (ESAS 2004)*, LNCS 3313. Springer, Aug. 2004, pp. 2–18.

[7] G. Gaubatz, J.-P. Kaps, E. Öztürk, and B. Sunar, "State of the art in ultra-low power public key cryptography for wireless sensor networks," in *Third IEEE Int'l Conf. Pervasive Computing and Communications Workshops, Workshop Pervasiv Computing and Comminucations Security–PerSec'05.* IEEE Computer Society, Mar. 2005, pp. 146–150.

[8] K. Yüksel, J.-P. Kaps, and B. Sunar, "Universal hash functions for emerging ultra-low-power networks," in *Proc. Communications Networks and Distributed Systems Modeling and Simulation Conf. (CNDS)*, Jan. 2004.

[9] B. Schneier, *Applied Cryptography*, 2nd ed. John Wiley & Sons, 1996.

[10] M. Wegman and L. Carter, "New hash functions and their use in authentication and set equality," *J. Computer and System Sciences*, vol. 22, no. 3, pp. 265–279, Jun. 1981.

[11] J. Black, S. Halevi, H. Krawczyk, T. Krovetz, and P. Rogaway, "UMAC: Fast and secure message authentication," in *CRYPTO '99*, LNCS 1666. Springer, 1999, pp. 216–233.

[12] J. Pelzl, T. Wollinger, J. Guajardo, and C. Paar, "Hyperelliptic curve cryptosystems: Closing the performance gap to elliptic curves," in *CHES 2003*, LNCS 2779. Springer, Sep. 2003, pp. 351–365.

Jens-Peter Kaps is an assistant professor of electrical and computer engineering in the Volgenau School of Information Technology and Engineering at George Mason University. His research interests include ultralow-power cryptographic hardware design, computer arithmetic, efficient cryptographic algorithms, and computer and network security. Kaps received a PhD in electrical and computer engineering from Worcester Polytechnic Institute. He is a member of the IEEE Computer Society and the International Association of Cryptologic Research. Contact him at jpkaps@computer.org.

Gunnar Gaubatz is a PhD candiate in the Department of Electrical and Computer Engineering at Worcester Polytechnic Institute, Massachusetts, and a research assistant with the Cryptography and Information Security Laboratory. His research interests are in fault-tolerant cryptography, computer arithmetic and low-power digital circuits. He received an MS in electrical engineering from Worcester Polytechnic Institute and is a student member of the IEEE Computer Society and the International Association for Cryptologic Research. Contact him at gaubatz@ieee.org.

Berk Sunar is an associate professor in the Department of Electrical and Computer Engineering at Worcester Polytechnik Institute. He is currently heading the Cryptography and Information Security Laboratory (CRIS) and his research interests include finite fields, elliptic curve cryptography, low-power cryptography, and computer arithmetic. Sunar received Ph.D. degree in Electrical and Computer Engineering (ECE) from Oregon State University. He is a member of the IEEE Computer Society, the ACM, and the International Association of Cryptologic Research (IACR). Contact him at sunar@ece.wpi.edu.