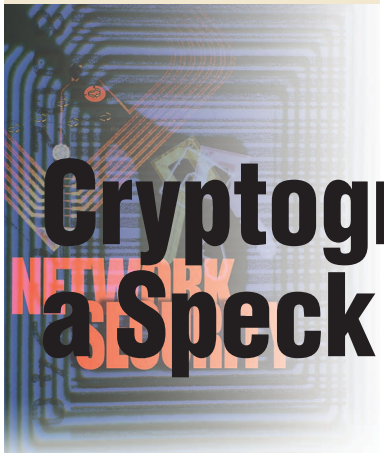

Cryptography on a Speck of Dust

Jens-Peter Kaps, George Mason University
Gunnar Gaubatz and Berk Sunar, Worcester Polytechnic Institute

As tiny wireless sensors and RFID tags become ubiquitous, they impact privacy, trust, and control. Protecting data on these devices requires new algorithms suitable for ultralow-power implementations.

Wireless sensor networks (WSNs)[1] and radio frequency identification (RFID) devices—with applications ranging from supply-chain management to home automation and healthcare—are quickly becoming a vital part of our infrastructure. Security is a critical factor in these ultralow-power applications.[2] However, these tiny, pervasive computing devices have extremely limited power resources and computational capabilities. Thus, security engineers face the seemingly contradictory challenge of providing lightweight 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 and provide secure communication using software-implemented cryptographic protocols, such as Security Protocols for Sensor Networks (Spins).[3] We envision that next-generation sensor nodes will operate without batteries, harvesting energy instead from ambient sources in the environment. The notion of self-powered computing devices opens the door to a wealth of new applications, not just for wireless sensor nodes

Ultralow-Power Application Domain

Energy scavengers harvest energy from environmental sources such as light, heat, and vibration and convert it into electric power. On-chip microelectromechanical system (MEMS)-based power scavengers can currently produce up to 8 µW of power.[1] Future MEMS-based scavengers could conceivably deliver up to 50 µW continuously.

Most RFID tags contain a transponder and a read-only memory chip that has a unique electronic product code, or *global unique identifier*. More sophisticated tags have writeable memory and some even have sensors. Active tags carry a small battery; 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, as 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 don't produce enough energy to support even the simplest general-purpose low-power CPUs currently being used in sensor networks.

Reference

1. S. Meininger et al., "Vibration-to-Electric Energy Conversion," *IEEE Trans. VLSI Systems*, Feb. 2001, pp. 64-76

Published by the IEEE Computer Society
0018-9162/07/$25.00 © 2007 IEEE

Table 1. Common elements in block ciphers.

Algorithm	No. of rounds	Block size (bits)	Feistel	Substitution-permutation (SP) network	Arithmetic S-box	Pseudo-random S-box	Fixed permu-tations	Addition (mod 2w)	Fixed shift/ rotation	Variable shift/ rotation	Modular multiplication	Multiplication with a constant
DES/ Triple DES	6/48	64	Yes			Yes	Yes		Yes			
IDEA	8	64	Yes					Yes			Yes	
RC5	12/16	32/128	Yes					Yes		Yes		
AES (Rijndael)	10/12/14	128		Yes	Yes		Yes		Yes			Yes
RC6	20	128	Yes					Yes		Yes	Yes	
MARS	2×16	128	Yes			Yes		Yes	Yes	Yes	Yes	
Serpent	32	128		Yes		Yes	Yes		Yes			
Twofish	16	128	Yes			Yes		Yes	Yes		Yes	Yes
XTEA	More than 31	32	Yes					Yes	Yes			

but for the entire ubiquitous computing field.

As the "Ultralow-Power Application Domain" sidebar describes, to provide cryptographic functions for this class of devices, designers must make power consumption their first priority. Thus far, research has concentrated on WSNs' network-specific aspects and on software implementations of cryptographic algorithms. Only recently have researchers published studies of special hardware implementations.[4-8]

A wireless sensor node's main power consumer is its RF transceiver, or radio. Hardware developed specifically for radios combines a low data rate, low power consumption, and the ability to interface directly with low-power microcontrollers. Providing adequate security for ultralow-power applications such as WSNs and RFID devices requires an approach similar to that used for radios—that is, specialized hardware combined with application-specific algorithms.

Other factors also play a role. For example, despite advances in radio design, transmission power is still costly compared to computations. Hence, we must keep the transmission overhead that applying security incurs to an absolute minimum.

Furthermore, most current cryptographic algorithms are designed for high performance, mostly in software on 32-bit microprocessors. On sensor nodes and RFID devices, computation time isn't as critical as power and space conservation. To this end, we introduce useful techniques for cryptographic algorithm implementers as well as guidelines for designing cryptographic algorithms for ultralow-power applications.

SURVEY OF CRYPTOGRAPHIC ALGORITHMS

As a first step, we examine current popular cryptographic algorithms as well as some we have found particularly interesting for use in this domain.[8]

Block ciphers

A wide variety of established block cipher designs share similar functions and structures. Our list of algorithms consists of classic block ciphers (Data Encryption Standard/Triple DES, International Data Encryption Algorithm [IDEA], and RC5), the Advanced Encryption Standard (AES) finalists, and the Extended Tiny Encryption Algorithm (XTEA). Table 1 summarizes their features.

Round structures. Virtually all modern block ciphers are *iterated product ciphers*—that is, the encryption process consists of repeated applications of a round function. The round function consists of multiple layers of transformations (also called confusion and diffusion layers) that perform substitution and permutation. The round function itself isn't considered secure, but each additional round increases the security level.

Popular round structures—used by DES, RC5, MARS, and so on—are variations of the Feistel network, in which the round function typically modifies only part of the round data. Some publications therefore refer to rounds in Feistel ciphers as half-rounds. Substitution and permutation networks used by IDEA, Rijndael (AES), and so on typically modify the entire data set in each round.

Substitution functions. All product ciphers use some form of substitution function, or *S-box*, to introduce nonlinearity into the encryption process. Techniques range from lookup-table-based pseudorandom substitutions to high-degree nonlinear arithmetic functions.

Permutation. Product ciphers combine various transformations for confusion and diffusion. They achieve diffusion through fixed permutations (for example, initial permutation [IP], inverse IP [IP^{-1}], and permutation [P] in DES) or data-dependent (variable) shifts and rotations (used by RC5/6 and MARS, for example).

Key mixing. Most block ciphers add subkeys to the round data using XOR operations, which are fast and

Table 2. Summary of hash function characteristics.

Hash function	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		Yes	Yes
SHA-1	4	512	160	16	20		32		Yes	Yes
MD4	3	512	128	16	16		32		Yes	Yes
MD5	4	512	128	16	16		32		Yes	Yes
NH	N/A	64	64			64	64			
WH	N/A	64	64					Yes	Yes	

introduce virtually no overhead in either software or hardware implementations. Several algorithms use a blend of XORs and regular integer addition. However, the resulting addition is no longer commutative, thereby complicating cryptanalysis.

Arithmetic operations. Certain arithmetic operations are useful for combining diffusion and nonlinear mixing of round data with key bits. A good example is truncated integer multiplication, which MARS uses.

Stream ciphers

The predominant method for building stream ciphers uses a pseudorandom number generator (PRNG) to generate a key stream and XOR its output with the data stream. The other method uses a dedicated stream cipher such as the proprietary RC4, which uses integer addition modulo 256 and a dynamic S-box with 256 8-bit entries. The S-box is a lookup table in which the entries change with each encrypted byte depending on the key.

Developers can build PRNGs from block ciphers, hash functions, modular exponentiators, or linear feedback shift registers (LFSRs) based on stop-and-go generators. Using block ciphers, hash functions, or modular exponentiators has a big advantage over using a dedicated stream cipher because these algorithms can maintain their original function and adding the stream cipher functionality comes at a minimal extra cost.

Hash functions

A hash function produces a short fixed-size digest of a long message that's useful for checking the message's integrity. *Message authentication codes* are hash functions that use a secret key and hence provide both message authentication and integrity. Universal hash function families provide provable security, and can be used to build provable secure MACs.[9] In other words, we can prove bounds on an attacker's success probability independent of the applied computational power.

In addition to the most popular hash functions, we focus on two universal hash function families: NH[10] and WH.[7] Table 2 summarizes some hash function parameters.

Each hash function has a fixed input size. The algorithm splits longer input strings and pads shorter ones. The hash size specifies the resulting hash value's length independent of the input string's length. The hash functions MD4, MD5, and those described in the secure hash standard (SHS), such as SHA-1 and SHA-256, 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. SHA-1 is widely used despite recent advances in attacking it; however, MD4 and MD5 are considered compromised.

Introduced as a new hash function family for the MAC using universal hashing (UMAC), NH is based on modular integer multiplication and summation. WH, a hash function family with even stronger security properties than NH, can be used in place of NH. WH is specifically designed for implementation in ultralow-power hardware and is based on modular polynomial multiplication and summation. NH and WH are defined for any fixed block size. For Table 2, we assume a block size of 64 bits because this was the size we used in our implementation.[7]

Public-key cryptosystems

Classic public-key algorithm security (such as that used by RSA and ElGamal) typically relies on the hard problem of integer factorization or on finding the discrete logarithm (DL) in a finite field. The dominating arithmetic operation is modular exponentiation, which is typically implemented using modular multiplication and squaring operations.

Rabin's scheme, a variant of RSA, fixes the public-key exponent to 2. This significantly reduces the encryption operation's complexity to modular squaring and allows for a compact implementation, whereas the general RSA case would be too large. Because security in both Rabin's scheme and RSA relies on integer factorization, we assume that there is no compromise in terms of security.

Elliptic curve cryptography (ECC) uses a variant of the DL problem defined over the additive group of points on an elliptic curve. The algorithm repeatedly adds (or doubles) points until it obtains a scalar multiple of the originating point. A single point addition consists of a heterogeneous variety of finite field operations such as addition/subtraction, multiplication, and, in some cases, inversion.

Table 3. Comparison of public-key cryptography functions.

Implementation	Encryption	Signature	Message payload (bits)	Ciphertext (bits)	Signature length (bits)	Integer multiplication (bits)	EC point addition (bits)	Polynomial coefficients (bits)
Rabin's scheme	Yes	Yes	Less than 512	512	512	512		
NtruEncrypt	Yes		Less than 265	1,169				8
NtruSign		Yes			1,169			8
Elliptic Curve MV	Yes		Less than 200	400			169	
Elliptic Curve DSA		Yes			200		169	

Researchers have proposed hyperelliptic curve cryptography (HECC) as a generalization of ECC. In HECC, operands can be even shorter than in ECC while maintaining an equivalent security level. HECC's arithmetic structure is even more complex than ECC's, however. Although efficient explicit expressions exist for the group operations,[11] they contain diverse arithmetic primitives that might prove too complex for ultralow-power implementations.

The Ntru encryption algorithm is based on the shortest vector problem in high-dimension lattices. NtruEncrypt's central arithmetic primitive is multiplication in a truncated polynomial ring. We can subdivide the operation itself into individual computations of polynomial coefficients through accumulation of partial products.

Table 3 compares the algorithm parameters for Rabin's scheme, Ntru, Elliptic Curve Menezes Vanstone (ECMV) encryption, and Elliptic Curve Digital Signature Algorithm (ECDSA).

ANALYSIS

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 for ultralow-power implementations.

For our example implementations, we used the TSMC 0.13-µm application-specific integrated circuit (ASIC) library, which is characterized for power, and the Synopsys (www.synopsys.com) Design Compiler and Power Compiler tools for synthesis. We used ModelSim (www.model.com) to simulate and capture switching activity, which Power Compiler uses to estimate power. We observed that at a 500-kHz clock frequency, which is common in sensor nodes, the static power consumption P_{Leak}, caused by leakage, outweighs the dynamic power consumption P_{Dyn}, caused by switching activity.

Algorithm structure

An algorithm's structure indicates how well it lends itself to parallelization and serialization. The latter is directly related to minimizing circuit area and therefore static power consumption.

Scalability refers to the possibility of efficiently scaling an algorithm between bit serial and highly parallelized realizations. In certain contexts, such as public-key cryptography, scalability can also encompass the reuse of existing processing elements for higher-precision operands than originally intended—for example, using 1,024-bit modular exponentiator hardware for 2,048-bit operands.

The iterative round structure of most block ciphers indicates a reasonable degree of scalability, provided that all rounds are the same. Then, we need only implement one instance of the round function. Serialization is possible with RC5 and RC6 and, ignoring a slightly modified last round, most other ciphers.

Unlike block ciphers, public-key schemes are based primarily on arithmetic over large integer or polynomial fields. This usually lends itself well to serialization, even though certain operations (modular reduction, for example) introduce additional complexity, which hinders serialization beyond a certain point. The biggest problem with serial implementations is running time, which is cubic in the operand size for most public-key algorithms. It's important to evaluate the tradeoff between area usage and a certain degree of parallelization.

Modularity is closely related to scalability in the sense that we can easily replicate simple processing elements for further task parallelization to improve performance. Our implementation of the NtruEncrypt algorithm provides an example.[5]

We can subdivide NtruEncrypt's basic operation into computations using 8-bit-long polynomial coefficients. By intelligently arranging memory accesses to these coefficients, we can compute multiple coefficients of the result in parallel. Because operand storage is the largest portion of the circuit, scaling up the number of parallel arithmetic units (AUs) has little effect (less than a 50 percent increase) on the overall area and power consumption. At the same time, this process can reduce the number of clock cycles dramatically, as the first two rows in Table 4 on the next page show.

Regularity describes the degree of similarity between modules at different levels of parallelization:

- At the logic level, highly regular designs allow efficient parameterization and reuse, while irregular cir-

Table 4. Comparison of implementations using a 0.13-µm application-specific integrated circuit library and a 500-kHz clock frequency.

Implementation	Power (µW) P_{Dyn}	Power (µW) P_{Leak}	Power (µW) Total	Area[1]	Delay (ns)	Clock cycles[2]	PDP (ns × µW)
NtruEncrypt (1 arithmetic unit)	4.03	15.1	19.1	2,850	0.69	29,225	13.18
NtruEncrypt (8 arithmetic units)	5.00	22.5	27.5	3,950	0.69	3,682	18.96
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
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

[1] Area is given in terms of equivalent two-input NAND gates
[2] Number of clock cycles to complete one operation

cuits 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 irregularity characterizes very complex tasks consisting of many atomic operations.

Rabin's scheme, NH, and WH are examples of algorithms with high regularity. Each has one simple underlying function. Block ciphers require serialization of the round function because even a single round can take a considerable amount of chip area. Ciphers with a homogeneous round function, such as AES, have high regularity and therefore seem better suited for serialization than ciphers with a heterogeneous structure, such as DES.

Energy equals the amount of power dissipated over time. Increasing the degree of parallelism increases power consumption, but it also decreases computation time. Because certain elements can have constant size, the power-energy tradeoff depends on the architecture's overall structure. We can find the point of optimality by modeling the energy consumption as a function of the degree of parallelism. Energy per bit encrypted describes the amount of energy needed to encrypt a single message bit. We can use this metric, which is independent of the actual operand length, to compare the energy efficiency of cryptosystems at an equivalent security level.

Functional primitives

Each group of primitives has specific characteristics and suitability for ultralow-power implementation.

Simple logic functions. In this group, the logic function output depends on a small, fixed set of inputs. This includes functions such as XORs of two bit strings (`AddRoundKey` in AES), bit multipliers, and multiplexers. The number of logic gates scales linearly with the data path's width.

Fixed shifts and permutations. In this article's context, *fixed* means that the shifts and permutations aren't data-dependent. Block ciphers such as DES, Serpent, and AES use permutations and expansions as nonlinear diffusion elements. Fixed shifts and rotations serve the same purpose and are frequently used in both block ciphers (such as AES, MARS, Serpent, and Twofish) and hash functions (such as MD2, SHA-1, MD4, and MD5). Common to all of these functions is that their implementation introduces virtually no cost because they require only wiring resources and no logic. They are perfect for any hardware implementation.

Data-dependent shifts. Because of their resistance to differential cryptanalysis, data-dependent shifts (or variable shifts) are used in block ciphers such as RC5, RC6, and MARS. 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 a register follows the shift or rotation, implementing the register as a shift register with parallel load and combining it with additional control logic and a counter might be more power-efficient. Because of the relatively high area cost, variable shifts are poorly suited for ultralow-power implementations unless they're combined with existing registers.

Integer arithmetic. Integer arithmetic primitives, such as addition and multiplication, are often the most costly functions in a cryptographic algorithm. Although we can often implement them in a bit-serial fashion (such as multiplication), the efficient propagation of carries presents a major problem. The carry propagate, or ripple carry adder (the simplest adder form), scales linearly with the word size n, but glitches in the carry chain cause high dynamic power consumption. Various alternatives exist, but they carry a penalty in terms of area and therefore static power consumption. Because of these costs, new ultralow-power algorithms should avoid integer arithmetic.

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

Polynomial arithmetic. Polynomial arithmetic—arithmetic in extension fields—is preferable for ultralow-power implementations because of limited carry propagation and improved regularity. Therefore, several algorithms, including AES, are specifically tailored for arithmetic in $GF(2^k)$. We can implement additions in fields of characteristic two

by using XORs. The simplicity of the addition and reduction steps facilitates a bit-serial implementation of multiplication for ultralow-power applications.

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

Multiencryption and multihashing increase an algorithm's security by applying it repeatedly.

Substitution functions. Various techniques can implement substitution functions (S-box). Although implementing lookup tables is fast and easy, their size often makes them prohibitively expensive. If performance is secondary to low-power consumption and an arithmetic description for the S-box exists, a circuit realization of the underlying arithmetic operation might be preferable.

The last two rows in Table 4 summarize the results of a case study using different AES S-box architectures for ultralow-power implementations. For one circuit, we implemented the S-box as an arithmetic function using its inherent algebraic structure. For the other circuit, we implemented a 256×8-bit lookup table in combinational logic. Even though the combinational implementation uses only 30 percent of the arithmetic implementation's dynamic power, its size makes its total power consumption two times higher. Thus, describing the S-box's content algebraically is advantageous and allows serialization.

Storage requirements

Cryptographic algorithms have manifold storage requirements. The constants and variables that an algorithm uses as well as implementation-specific storage elements add to these requirements. Constants consist of fixed setup parameters, precomputed constants, and static S-boxes. The first two—fixed parameters and precomputed constants—can be implemented in combinational logic. However, devices must store variables, including variable S-boxes (RC4), and temporary data in registers or RAM. Pipelining techniques require additional storage elements. Because storage elements typically impose significant area and power penalties, ultralow-power implementations should use them conservatively.

Implementation considerations

Additional considerations go beyond a cryptographic algorithm's structure and elementary functions.

Multiencryption and multihashing increase an algorithm's security by applying it repeatedly. Triple DES is probably the best-known example of multiencryption. It applies DES three times in a row, using either two or three different keys depending on the keying option. Researchers originally developed Triple DES to prolong DES's lifetime until they established a new, more secure standard. However, multiencryption and multihashing can also enable ultralow-power cryptography. We can use block ciphers or hash functions that consume 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 can help alleviate cryptosystems' storage problem and can even simplify certain computations. This is highly dependent on the intended application context. For example, because Internet servers typically must change keys and associated key parameters frequently, constant parameters are impossible. 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 significantly reduce the storage requirements.

Precomputation is a powerful method for solving latency problems. It's especially important for low-power nodes in which 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, processing the input data will require only a small number of computations and latency might be virtually eliminated.

DESIGN RECOMMENDATIONS FOR NEW ALGORITHMS

The most important requirement for a new cryptographic algorithm is scalability. Implementers should be able to scale the algorithm from a bit-serial implementation to a highly parallel implementation depending on the desired maximum power consumption and speed.

An algorithm's scalability depends on its regularity. New cryptographic algorithms should be regular and contain only a limited number of different primitives. To further improve scalability, the basic functions should be serializeable. The implementer can then trade speed for power on a fine level of granularity, not just on the algorithmic level.

Assuming clock speed is held constant, serializing an algorithm slows its operation. However, in environments in which data must be sent quickly but infrequently, computing most steps offline ahead of time is desirable. When the data becomes available, only a simple, fast computation should be required to complete the operation—for example, adding data to the key.

WSN and RFID security requirements vary by application, ranging from high-risk applications such as mil-

itary target tracking, where attacks on devices are likely, to low-risk applications such as passive environmental monitoring. To support such a range of applications, the new algorithms should operate with various key lengths. Multihashing and multiencryption can also increase the security level without increasing the footprint.

Considerations for implementing elementary functions include:

- An algebraic representation can be more efficient than a costly lookup table.
- Polynomial arithmetic in $GF(2^k)$ and fixed shifts and rotations are well suited for hardware implementations.
- Integer arithmetic consumes a high degree of power.
- Data-dependent shifts and rotations are costly unless they're combined with existing registers.

WSN and RFID messages are usually small, ranging from 30 to 100 bits in length. Transmitting a bit consumes more power than computation. New algorithms should therefore have a compact representation of cipher I/O. Encryption functions shouldn't cause message expansion and should use a small block size. Hash functions should result in small digests because they're transmitted in addition to the original data. Ideally, the digest size won't affect the collision probability.

The challenge of future research is to find an algorithm that has at its core a simple, scalable primitive that could serve as a common element for secret and public-key functions. Such an algorithm could provide both types of functions for ultralow-power applications, which in turn would enable simple and efficient security protocols.

References

1. D. Culler, D. Estrin, and M. Srivastava, "Overview of Sensor Networks," *Computer*, Aug. 2004, pp. 41-49.
2. A. Perrig, J. Stankovic, and D. Wagner, "Security in Wireless Sensor Networks," *Comm. ACM*, June 2004, pp. 53-57.
3. A. Perrig et al., "SPINS: Security Protocols for Sensor Networks," *Wireless Networks*, Sept. 2002, pp. 521-534.
4. M. Feldhofer, S. Dominikus, and J. Wolkerstorfer, "Strong Authentication for RFID Systems Using the AES Algorithm," *Proc. Cryptographic Hardware and Embedded Systems* (CHES 04), LNCS 3156, Springer, 2004, pp. 357-370.
5. G. Gaubatz, J.-P. Kaps, and B. Sunar, "Public-Key Cryptography in Sensor Networks—Revisited," *Proc. 1st European Workshop Security in Ad-Hoc and Sensor Networks* (ESAS 04), LNCS 3313, Springer, 2004, pp. 2-18.
6. G. Gaubatz et al., "State of the Art in Ultralow-Power Public-Key Cryptography for Wireless Sensor Networks," *Proc. 3rd IEEE Int'l Conf. Pervasive Computing and Comm. Workshops*, IEEE CS Press, 2005, pp. 146-150.
7. J.-P. Kaps, K. Yüksel, and B. Sunar, "Energy Scalable Universal Hashing," *IEEE Trans. Computers*, Dec. 2005, pp. 1484-1495.
8. K. Yüksel, J.P. Kaps, and B. Sunar, "Universal Hash Functions for Emerging Ultra-Low-Power Networks," *Proc. Conf. Comm. Networks and Distributed Systems Modeling and Simulation* (CNDS 04), Soc. for Modeling and Simulation Int'l (SCS), 2004, in press.
9. M. Wegman and L. Carter, "New Hash Functions and Their Use in Authentication and Set Equality," *J. Computer and System Sciences*, June 1981, pp. 265-279.
10. J. Black et al., "UMAC: Fast and Secure Message Authentication," *Proc. Advances in Cryptology* (Crypto 99), LNCS 1666, Springer, 1999, pp. 216-233.
11. J. Pelzl et al., "Hyperelliptic Curve Cryptosystems: Closing the Performance Gap to Elliptic Curves," *Proc. Cryptographic Hardware and Embedded Systems* (CHES 03), LNCS 2779, Springer, 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 candidate in the Department of Electrical and Computer Engineering at Worcester Polytechnic Institute 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. Gaubatz received an MS in electrical engineering from Worcester Polytechnic Institute. He is a student member of the IEEE Computer Society and the International Association for Cryptologic Research. Contact him at gaubatz@ieee.org.*

Berk Sunar*, an associate professor in the Department of Electrical and Computer Engineering at Worcester Polytechnic Institute, is the head of the Cryptography and Information Security Laboratory. His research interests include finite fields, elliptic curve cryptography, low-power cryptography, and computer arithmetic. Sunar received a PhD in electrical and computer engineering from Oregon State University. He is a member of the IEEE Computer Society, the ACM, and the International Association of Cryptologic Research. Contact him at sunar@ece.wpi.edu.*