The SIMON and SPECK lightweight block ciphers

Ray Beaulieu Douglas Shors Jason Smith
Stefan Treatman-Clark Bryan Weeks Louis Wingers

National Security Agency
9800 Savage Road
Fort Meade, MD, 20755, USA
{rabeaul,djshors,jksmit3,sgtreat,beweeks,lrwinge}@tycho.ncsc.mil

Invited

ABSTRACT

The SIMON and SPECK families of block ciphers were designed specifically to offer security on constrained devices, where simplicity of design is crucial. However, the intended use cases are diverse and demand flexibility in implementation. Simplicity, security, and flexibility are ever-present yet conflicting goals in cryptographic design. This paper outlines how these goals were balanced in the design of SIMON and SPECK.

CCS Concepts

•**Security and privacy → Block and stream ciphers;**

Keywords

SIMON, SPECK, lightweight, block cipher, Internet of Things

1. INTRODUCTION

Cryptographic design is all about compromise. For robust security, there is a desire to use components with strong cryptographic properties, and to build in a large margin of security by stepping an algorithm many more times than may seem necessary. The competing aim—efficiency—means that we want to minimize computation to the extent possible. How one balances these conflicting goals is the art of cryptography.

The problem is complicated by the fact that efficiency is not a well-defined notion. An algorithm can have efficient realizations in dedicated hardware (e.g., on an ASIC), but sacrifice performance on 8-bit microcontrollers. *Or* it could admit high-throughput implementations on 64-bit desktop processors, but require a large amount of code. *Or* it could be designed to optimize performance on a particular processor (by taking full advantage of the instruction set for that processor). *Or* it could be tailored for high performance on a particular FPGA (for example by basing it on LUTs of a certain size). Or it could have very high-throughput pipelined ASIC implementations, but offer no compact realization suitable for constrained devices.

Whether or not an algorithm is secure may seem clearer, but there are plenty of questions here as well. A fundamental security goal is that the most efficient way to recover the secret key should be to try all the keys (in an intelligent way, of course, minimizing computation along the way as much as possible). But what is the attack model? It's accepted that a hypothetical attacker should be given access to loads of matched plaintext/ciphertext pairs, and even be allowed to choose plaintexts and ciphertexts in an adaptive manner. But how much data? The easy answer is "all of it", but there are real-world costs associated with blocking *theoretical* attacks that in practice might require more energy than we would ever be able to produce.[1]

In addition, more liberal attack models have been proposed. Should we care about related-key attacks, whereby the attacker is allowed to manipulate secret keys (without learning what they are)? What sort of manipulations should be allowed? Bit-flipping only? Or something more general? (If it's too general then it's been shown that security is impossible to achieve!) What about attacks in the open-key model, where the attacker knows (or even chooses) the key, and then seeks to discover certain sorts of nonrandom properties of the cipher?

Finally, regarding security, the current state of our knowledge means that there are no guarantees in the world of symmetric-key cryptography. The typical—and reasonable—approach of cryptographers is to prove that certain standard attacks don't work when applied to their algorithms. But the collection of standard attacks considered relevant today is certain to be a proper subset of the analogous collection 20 years from now.

Given this state of affairs, how much of a buffer should we build in to our algorithms to guard against future cryptanalytic advances? The answer to this depends on whether we expect these advances to be incremental (say, improved techniques for implementing linear cryptanalytic attacks), or revolutionary (such as a demonstration that P = NP). There's not much we can do about the latter, where we currently have no insight, beyond perhaps designing hugely

[1]See [2]: the energy necessary to try all 2^{128} AES-128 keys would boil all the oceans on earth about 16000 times over. And this is the small AES key size. Doing the same for AES-256 would boil the oceans about 10^{42} times.

DAC '15 June 07 - 11, 2015, San Francisco, CA, USA

ACM ISBN 978-1-4503-3520-1/15/06…$15.00
DOI: 10.1145/2744769.2747946

over-engineered and wildly inefficient algorithms. But such an approach does not seem to strike the right balance between our current needs and the future threat. There is hope, however, that we can gain some understanding of what incremental advances might look like by examining what has happened in the past. It is encouraging that the cryptanalysis of the most-studied algorithms (e.g. AES) seems to have stabilized pretty early in their lives (except, of course, when they are subjected to analysis using assumptions previously not considered).

And so far we haven't even mentioned side-channel attacks. While such attacks exploit properties of implementations rather than of algorithms *per se*, choices the designer of a block cipher makes can affect the ease with which side-channel mitigations can be made.

With all this in mind, the cryptographer must think carefully about his or her goals before embarking on a design effort.

2. SIMON AND SPECK

In recent years, computational tasks have progressively been pushed down to smaller and smaller devices. It has been recognized that traditional cryptography is not always particularly well-suited to the needs of this emerging reality.

Lightweight cryptography seeks to address this, by proposing algorithms and protocols designed specifically to perform well on these constrained platforms. In this paper we discuss our contribution, the lightweight block cipher families SIMON and SPECK, and we attempt to explain some of the thinking behind the designs.

We would like to briefly discuss the history of, and the motivation for, this project.

SIMON and SPECK were proposed publicly in June 2013 [10] by a group of researchers in the US National Security Agency's Research Directorate. Design work began in 2011, and a significant amount of cryptanalysis was done (not just by the designers, but by many others across the enterprise) in the time leading up to the publication. The result of this is that we believe the algorithms are secure.

Why did we do this? After all, many lightweight block ciphers have been proposed, and many perform well on various constrained platforms. The simple answer is that we thought we could use the considerable experience we have in cryptographic design (extending back many decades) to improve on what was available, and thus facilitate security in the new Internet of Things era.

The major issue we saw was that the lightweight encryption algorithms that had been proposed tended to be somewhat lacking in flexibility. That is, they typically targeted a particular sort of platform (most often ASICs) and the designer was usually able to achieve good performance on that platform. But too often little consideration was given to performance on platforms other than the intended platform. We believe that flexibility will be crucial to developers in the coming years, when heterogeneous networks connecting disparate, tiny devices will require cryptography that performs adequately on all the devices. Therefore, we don't believe it makes sense to optimize for a specific platform; rather, it's better to make algorithms that are so simple that they will perform well just about anywhere. After all, we don't know what sort of new devices will exist in, say, 2025. But we can be pretty sure that if they can do any computation, then they will support simple operations such as addition, `AND`, `XOR`, etc.

A case in point here is PRESENT [16], which is perhaps the leading lightweight block cipher. It does what it was intended to do, admirably: it has extremely compact ASIC implementations. But it comes up a little short if it has to be implemented on a constrained software platform[2] or on an FPGA.[3] —which is not a surprise, because such performance was not the primary design goal.

A further limitation we saw was that existing lightweight block ciphers tended to have a fixed block size, and one, or at most two, key sizes. We wanted to provide the additional flexibility to tailor a block and key size to the application at hand. To that end, SIMON and SPECK each have multiple instantiations, supporting block sizes of 32, 48, 64, 96, and 128 bits, and with up to three key sizes to go along with each block size. Each family provides ten algorithms in all. Table 1 lists the different block and key sizes, in bits. We defer a further (but brief) description of the algorithms to Section 4. A full description can be found in [10].

block size	key sizes
32	64
48	72, 96
64	96, 128
96	96, 144
128	128, 192, 256

Table 1: Simon and Speck parameters.

Our aim was that SIMON and SPECK should be flexible enough to perform well on the full spectrum of constrained platforms, and this motivated us to choose the simplest components possible. But the simplicity of the designs had an added benefit: we ended up with algorithms that have exceptional performance on high-end platforms as well. As far as we are aware, SPECK has the highest throughput on 64-bit processors of *any* block cipher implemented in software,[4] and amongst block ciphers in the literature, SIMON appears to have the highest throughput per unit area for pipelined ASIC implementations.

3. DETAILS

Flexibility imposes stringent limitations on the sort of operations that we could consider for SIMON and SPECK, as each operation had to have highly efficient realizations on a variety of hardware and software platforms. There is a relatively short list of such operations: `XOR`, `AND`, `OR`, `NOT`,

[2]See [20], where scores are assigned to algorithms based on their performance on 8-, 16-, and 32-bit microcontrollers, with smaller numbers being better: for use in authentication protocols, SPECK and SIMON finished first and second, respectively, out of the 13 algorithms considered, with scores of 3.9 and 6.1. PRESENT was 12th, with a score of 47.3.

[3]SIMON 128/128 can be implemented in 36 slices on a SPARTAN 3 FPGA [7] (we have reduced this to 28), and *all* 10 versions require just 90 slices on the SPARTAN 3 [23]. The smallest implementation of PRESENT-128 we know of requires 117 slices on this FPGA [29].

[4]Remarkably, on an Intel Haswell processor, SPECK 128/256 encrypts at 1.36 cycles/byte, which is *nearly* as fast as a hardware AES-NI implementation (1.02 cycles/byte).

rotations, end-off shifts, modular addition and subtraction, and perhaps a few others.

In particular, we eschewed Sboxes: 8-bit Sboxes, for example, can be quite strong cryptographically, but can also be relatively heavy in hardware and constrained software. It's less clear whether to avoid 4-bit Sboxes, which can be pretty compact. But they're not as compact as SIMON and SPECK's nonlinear functions, and are less amenable to software implementations. 3-bit Sboxes can be very compact, but the fact that 3 does not divide 32, 64, and 128 (block sizes we support) complicates their use.

If we're not going to use Sboxes, then our block ciphers will not be substitution-permutation networks (SPNs). The nice thing about SPNs is that they typically enable relatively simple security arguments. But we would maintain that it's not crucial that evaluation be easy—just that it should be possible, in order to gain a good understanding of security. Indeed, it's probably better to spend more time on the one-time work of evaluation if the benefit is that you get to spend less on the every-time work of encryption and decryption.

Rather than being SPNs, the SIMON and SPECK round functions are based on Feistel permutations, which can provide a good balance between linear diffusion and nonlinear confusion operations.

Speaking of diffusion, we note that care must be taken with bit permutations. In ASIC implementations, bit permutations are essentially free (requiring no gates; just wires). But a bit permutation not chosen with software in mind can result in greatly increased complexity for software implementations.[5] The bit permutations we use are *rotations*. While they don't have quite the same diffusion properties as the bit permutations in, say, PRESENT or PRINTcipher [25], they are cheap in hardware *and* software, and this more than makes up for the fact that we'll require additional rounds for sufficient diffusion.

Anticipating that there could be applications that will require highly optimized software or hardware algorithms, our design effort bifurcated, and we produced two families of algorithms. While both were meant to perform very well across the range of hardware and software platforms, when there was a design choice to be made, we made it to favor hardware over software for SIMON, and vice versa for SPECK.

For SIMON, the nonlinear function is the composition of two rotations and a bitwise `AND`, which requires minimal hardware. It's also easily computed, but a bit less so, in software (there is some price to be paid for doing multiple operations on the same word). But it's relatively weak cryptographically, and we pay the price by having to do a relatively large number of rounds.

SPECK uses modular addition for its nonlinearity. This function is cryptographically stronger than SIMON's `AND`, and is well suited to software implementations. But it can't be implemented quite as efficiently in hardware as an `AND`, because of the need to propagate carries.

In the next section we describe the algorithms, and then in Section 5 we discuss some of our additional considerations which caused the algorithms to look the way they do.

[5]Efficient *bit-sliced* implementations may be still be possible, but it doesn't seem wise to rely on these, as they have drawbacks—including relatively expensive data transpose operations on the plaintext and ciphertext, and the inability to efficiently encrypt single plaintext blocks (and single encryptions will be necessary for many lightweight communication and authentication protocols). In addition, the code size tends to be large, making such implementations unsuitable for some lightweight applications.

4. THE SIMON AND SPECK ALGORITHMS

We briefly describe the SIMON and SPECK algorithms here, but refer the reader to [10] for complete details.

The SIMON block cipher with an n-bit word (and hence a $2n$-bit block) is denoted SIMON$\,2n$, where n is required to be 16, 24, 32, 48, or 64. SIMON$\,2n$ with an m-word (mn-bit) key will be referred to as SIMON$\,2n/mn$. For example, SIMON$\,64/128$ refers to the version of SIMON acting on 64-bit plaintext blocks and using a 128-bit key. The notation for SPECK is analogous.

We use the following notation for operations on n-bit words, where the value of n should be understood from the context:

- bitwise `XOR`, $\oplus$,
- bitwise `AND`, $\&$,
- mod 2^n addition, $+$,
- left circular shift, S^j, by j bits.

The SIMON$\,2n$ round function is the map

$$(x, y) \mapsto (y \oplus f(x) \oplus k,\ x),$$

where x and y are n-bit quantities, $f(x) = (Sx \,\&\, S^8x) \oplus S^2x$, and k is a round key. The round keys are generated by a key schedule (see below), and the round function is composed a number of times that is a function of block and key size, shown in Table 2.

block size $2n$	key size mn	Simon rounds	Speck rounds
32	64	32	22
48	72	36	22
	96	36	23
64	96	42	26
	128	44	27
96	96	52	28
	144	54	29
128	128	68	32
	192	69	33
	256	72	34

Table 2: Simon and Speck rounds.

The SIMON key schedules employ round constants to eliminate slide properties; we omit discussion of the constants here (see [10] for details). There are three related key schedules, depending on whether the number of key words is two (the 96/96 and 128/128 sizes), three (48/72, 64/96, 96/144, and 128/192), or four (32/64, 48/96, 64/128, and 128/256).

The key schedules produce round keys $k_0, k_1, k_2, \ldots$ from a key value $k_0, \ldots, k_{m-1}$, where $m = 2$, 3 or 4. The two-, three-, and four-word key schedules, respectively, generate round keys as follows:

$$k_{i+2} = k_i \oplus (I \oplus S^{-1})S^{-3}k_{i+1} \oplus C_i,$$
$$k_{i+3} = k_i \oplus (I \oplus S^{-1})S^{-3}k_{i+2} \oplus D_i,$$
$$k_{i+4} = k_i \oplus (I \oplus S^{-1})(S^{-3}k_{i+3} \oplus k_{i+1}) \oplus E_i,$$

where C_i, D_i, and E_i are (round-dependent) constants.

The SPECK $2n$ round function is the map

$$(x, y) \mapsto ((S^{-\alpha}x + y) \oplus k,\ S^{\beta}y \oplus (S^{-\alpha}x + y) \oplus k)$$

where x and y are n-bit quantities, and k is a round key. The parameters α and β are 8 and 3, respectively, except in the case of SPECK 32/64, where they're 7 and 2.

Like SIMON, SPECK has two-, three-, and four-word key schedules. SPECK's key schedules are based on its round function, as follows. We let m be the number of words of key, and we write the key K as $(\ell_{m-2}, \ldots, \ell_0, k_0)$. We then generate two sequences of words k_i and ℓ_i by

$$\ell_{i+m-1} = (k_i + S^{-\alpha}\ell_i) \oplus i \quad \text{and}$$
$$k_{i+1} = S^{\beta}k_i \oplus \ell_{i+m-1}.$$

The value k_i is the ith round key, for $i \geq 0$. Note the round counter i here which serves to eliminate slide properties.

5. FURTHER CONSIDERATIONS

Now that we've described the algorithms, we can discuss in a bit more detail the considerations that went into the design choices we made.

Simplicity

SIMON and SPECK use *simple* round functions, iterated as many times as necessary for security. This contrasts with other algorithms (such as AES) which use relatively complex round functions but require fewer rounds.

The use of simple round functions means the algorithms have compact realizations, and are well suited for use on constrained platforms.

Uniformity

We wished to have one set of parameters for all the versions of SIMON, and similarly for SPECK. (In the end we backed off from this only for the smallest version of SPECK.) The reasons for this are

- it makes for a more succinct description of the algorithm families,
- it facilitates the analysis,
- and it allows for smaller joint implementations of all the versions, both in software and in hardware [23]. This has the added benefit that a simpler description makes coding errors less likely.

Moves

For software efficiency, it is best to avoid the need to make copies of particular words. The intent is to enable software implementations that require no moves.

We've noted that for SIMON, which requires multiple operations on the same word, we don't achieve this, and this is part of the reason that SPECK outperforms SIMON in software.[6] SPECK can be done entirely with in-place operations, and so moves are unnecessary, as is seen in this snippet of pseudocode for a round of SPECK:

```
x = RCS(x, α)
x = x + y
x = x ⊕ k
y = LCS(y, β)
y = y ⊕ x
```

[6]The other primary reason is that SPECK's nonlinear function is stronger, so fewer rounds are required.

Encrypt/decrypt symmetry

To enable compact joint implementations of the encryption and decryption algorithms, it's best to make encryption look like decryption (with the round keys reversed, of course). SIMON decryption can be accomplished by swapping ciphertext words, reading round keys in reverse order, and then swapping the resulting plaintext words.

We note that SIMON beats SPECK in this regard (SPECK decryption requires modular subtraction, and one of the rotations is reversed), *because* its Feistel stepping performs all operations on one word, which is precisely why its software implementation required moves.

Rotations

Many microcontrollers only support shifts by a single bit, which means that k-bit rotations must be implemented as a composition of k 1-bit rotations. On 8-bit microcontrollers, however, a rotation by 8 can often be achieved for free, as a simple relabeling of registers. These two facts mean that a rotation by k on an 8-bit microcontroller is most efficient when k is close to a multiple of 8.

This was an important consideration that constrained our exploration of the parameter spaces for SIMON and SPECK: we wanted rotation amounts to be as close to multiples of 8 as possible, subject to the constraint the the algorithms be secure for a not-unreasonable number of rounds.

Having opted to use rotations by 8, in particular, in both SIMON and SPECK, an important question is what sort of impact this had on performance on 16-bit, 32-bit, and 64-bit processors. It turns out 8-bit rotations can be done efficiently on most such platforms, because support is often provided for byte-oriented operations. For instance, on the MSP430 16-bit microcontroller, a rotation by 8 on a 64-bit value (four 16-bit words) can be accomplished in 12 cycles; it takes 5 cycles for rotation by one, so this is certainly a big improvement over doing eight 1-bit rotations. On current Intel processors, a rotation by 8 can be done using AVX2 byte shuffle commands, e.g., `_mm256_shuffle_epi8`.

There is a final consideration involving rotation amounts. We wanted SIMON and SPECK each to have a small bit-serial ASIC implementation. For SIMON, the rotation amounts have little impact on area (they just end up affecting the control bits in `MUX`es [9]. For SPECK, bit-serial implementations are a little more complicated than for SIMON, and there's a need to use a total of $|\alpha - \beta|$ `MUX`ed flip-flops to "freeze" the registers for $|\alpha - \beta|$ steps. To keep the area small, $|\alpha - \beta|$ should be small, and it's 5 for SPECK.

Key schedules

SPECK's reuse of the round function for key scheduling is a software feature, allowing code reuse, and it enables implementations with extremely small code size.

Because SIMON was optimized for hardware, it does not

take advantage of this software reuse idea. Instead, it uses a key schedule which was designed to be a little lighter than the round function.

Of course it is possible to have key schedules even simpler than the ones we have used for SIMON and SPECK; for example, one can produce round keys simply by cycling through key words. This leads to the possibility of "hardwiring" the key in an ASIC implementation, thereby saving considerably on area by eliminating any flip-flops needed for holding the key. But such an approach, when used together with very simple round functions, can lead to related-key issues, and we therefore avoided it.

We believe the ability to use hardwired key is of limited utility, and it violates our flexibility goal by optimizing for a particular sort of use (perhaps to the detriment of other uses in the form of increased numbers of rounds or cryptanalytic weaknesses).

Our key schedules do the minimal mixing that we thought would eliminate the threat of related-key attacks.

As a final point, we omit plaintext and ciphertext key whitening operations, as such operations would increase circuit sizes. This means that the first and last rounds of the algorithms do nothing cryptographically, beyond introducing the first and last round keys.

Constants

In order to eliminate slide issues, SPECK uses a one-up counter in its key schedule.

For SIMON, we wished to reduce circuit size as much as possible, and ended up using a sequence of 1-bit constants generated by a 5-bit linear register. In software the constants can be packed into words and stored, but this is a little less efficient than it would have been had we just used a one-up counter.

We note that this is a good example of how we made choices to benefit software over hardware for SPECK, and vice versa for SIMON.

6. OTHERS' WORK

While we have not been able to talk about our cryptanalytic efforts with respect to SIMON and SPECK, many researchers have published their own cryptanalytic examinations of the algorithms. See, for example, [3, 13, 4, 15, 19, 2, 6, 5, 1, 14, 26]

To date, all published "attacks" on SIMON and SPECK are of the reduced-round variety. The goal of this sort of analysis is to determine the maximal number of rounds that would be susceptible to a theoretical attack (i.e., anything better than an exhaustive key search); all block ciphers are subject to reduced-round attacks. A measure of security is the number of rounds that have been attacked, as a percentage of the total. So far no published attack makes it more than about 70% of the way through any version of SIMON or SPECK. (The best are 48 of 69 rounds for SIMON 128/128 (69.6%) [1] and 19 of 27 rounds for SPECK 64/128 (70.3%) [21]. Compare this to PRESENT-128, which, at 31 rounds, has a 26-round attack (83.9%) [24].) Of course it is possible to reduce this percentage for any block cipher by increasing the number of rounds, but this comes at a cost to efficiency. Based on our analysis, we believed we had set our stepping to appropriately balance efficiency and security, and we continue to believe this in light of the published analysis.

In summary, both algorithms continue to enjoy a healthy security margin, and there seems to be a consensus amongst those who have studied the algorithms that they are secure.

Many authors have studied implementations of SIMON and SPECK. There have been a number of results demonstrating record-breaking FPGA implementations [8, 23], some with side channel mitigations [12, 28], and highly efficient microcontroller implementations [11, 17, 20]. Novel uses have been proposed, including explorations showing the suitability of SIMON in the decidedly non-lightweight realm of homomorphic encryption [27, 18].

On a related note, some authors have begun exploring the design space for the algorithms and asking whether better parameters could have been chosen (see [26], in particular, which considers the SIMON family). The authors find no problems with the rotation amounts used by SIMON, but correctly note that there are other parameter choices worthy of consideration. They put forward three alternatives, and the design criteria outlined here make clear why we chose as we did: one of the alternatives has rotation amounts far from multiples of 8, leading to poor performance on microcontrollers; one has *very* slow diffusion, meaning the numbers of rounds would need to increase significantly, again leading to poor performance; and the final set of parameters yields a design which, though it has better diffusion through the initial rounds, is shown to be cryptographically weaker than SIMON.

7. REFERENCES

[1] M. A. Abdelraheem, J. Alizadeh, H. A. Alkhzaimi, M. R. Aref, N. Bagheri, P. Gauravaram, and M. M. Lauridsen. Improved linear cryptanalysis of reduced-round SIMON. Cryptology ePrint Archive, Report 2014/681, 2014. http://eprint.iacr.org/.

[2] F. Abed, E. List, S. Lucks, and J. Wenzel. Differential and linear cryptanalysis of reduced-round Simon. Cryptology ePrint Archive, Report 2013/526, 2013. http://eprint.iacr.org/.

[3] F. Abed, E. List, S. Lucks, and J. Wenzel. Differential cryptanalysis of round-reduced Simon and Speck. In *Fast Software Encryption, FSE 2014*, LNCS. Springer, 2014.

[4] J. Alizadeh, H. AlKhzaimi, M. R. Aref, N. Bagheri, P. Gauravaram, A. Kumar, M. M. Lauridsen, and S. K. Sanadhya. Cryptanalysis of SIMON variants with connections. In N. Saxena and A. Sadeghi, editors, *Radio Frequency Identification: Security and Privacy Issues - RFIDSec 2014*, volume 8651 of *LNCS*, pages 90–107. Springer, 2014.

[5] J. Alizadeh, N. Bagheri, P. Gauravaram, A. Kumar, and S. K. Sanadhya. Linear cryptanalysis of round reduced Simon. Cryptology ePrint Archive, Report 2013/663, 2013. http://eprint.iacr.org/.

[6] H. A. Alkhzaimi and M. M. Lauridsen. Cryptanalysis of the SIMON family of block ciphers. Cryptology ePrint Archive, Report 2013/543, 2013. http://eprint.iacr.org/.

[7] A. Aysu, E. Gulcan, and P. Schaumont. SIMON Says, Break Area Records of Block Ciphers on FPGAs. *Embedded Systems Letters, IEEE*, 6(2):37–40, June 2014.

[8] A. Aysu, E. Gulcan, and P. Schaumont. SIMON says,

break the area records for symmetric key block ciphers on FPGAs. Cryptology ePrint Archive, Report 2014/237, 2014. http://eprint.iacr.org/.

[9] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers. The SIMON and SPECK Block Ciphers on ASICs. To appear.

[10] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers. The SIMON and SPECK families of lightweight block ciphers. Cryptology ePrint Archive, Report 2013/404, 2013. http://eprint.iacr.org/.

[11] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers. The SIMON and SPECK block ciphers on AVR 8-bit microcontrollers. In Eisenbarth and Öztürk [22].

[12] S. Bhasin, T. Graba, J. Danger, and Z. Najm. A look into SIMON from a side-channel perspective. In *Hardware-Oriented Security and Trust, HOST 2014*, pages 56–59. IEEE Computer Society, 2014.

[13] A. Biryukov, A. Roy, and V. Velichkov. Differential analysis of block ciphers SIMON and SPECK. In *Fast Software Encryption, FSE 2014*, LNCS. Springer, 2014.

[14] A. Biryukov, A. Roy, and V. Velichkov. Differential analysis of block ciphers SIMON and SPECK. Cryptology ePrint Archive, Report 2014/922, 2014. http://eprint.iacr.org/.

[15] A. Biryukov and V. Velichkov. Automatic search for differential trails in ARX ciphers. In J. Benaloh, editor, *Topics in Cryptology - CT-RSA 2014*, volume 8366 of *LNCS*, pages 227–250. Springer, 2014.

[16] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT: An ultra-lightweight blockcipher. In *Cryptographic Hardware and Embedded Systems - CHES 2007*, volume 4727 of *LNCS*, pages 450–466. Springer, 2007.

[17] B. Buhrow, P. Riemer, M. Shea, B. Gilbert, and E. Daniel. Block cipher speed and energy efficiency records on the MSP430: System design trade-offs for 16-bit embedded applications. Cryptology ePrint Archive, Report 2015/011, 2015. http://eprint.iacr.org/.

[18] B. Carmer and D. W. Archer. Block ciphers, homomorphically. Galois, Inc. Blog, December 2014. http://galois.com/blog/2014/12/block-ciphers-homomorphically/.

[19] N. Courtois, T. Mourouzis, G. Song, P. Sepehrdad, and P. Susil. Combined algebraic and truncated differential cryptanalysis on reduced-round Simon. In M. S. Obaidat, A. Holzinger, and P. Samarati, editors, *SECRYPT 2014*, pages 399–404. SciTePress, 2014.

[20] D. Dinu, Y. L. Corre, D. Khovratovich, L. Perrin, J. G. schädl, and A. Biryukov. Triathlon of lightweight block ciphers for the internet of things. Cryptology ePrint Archive, Report 2015/209, 2015. http://eprint.iacr.org/.

[21] I. Dinur. Improved differential cryptanalysis of round-reduced Speck. In A. Joux and A. M. Youssef, editors, *Selected Areas in Cryptography - SAC 2014*, volume 8781 of *LNCS*, pages 147–164. Springer, 2014.

[22] T. Eisenbarth and E. Öztürk, editors. *Lightweight Cryptography for Security and Privacy - LightSec 2014*, volume 8898 of *LNCS*. Springer, 2014.

[23] E. Gulcan, A. Aysu, and P. Schaumont. A flexible and compact hardware architecture for the SIMON block cipher. In Eisenbarth and Öztürk [22].

[24] J. N. Jr., P. Sepehrdad, B. Zhang, and M. Wang. Linear (hull) and algebraic cryptanalysis of the block cipher PRESENT. In J. A. Garay, A. Miyaji, and A. Otsuka, editors, *Cryptology and Network Security, CANS 2009*, volume 5888 of *LNCS*, pages 58–75. Springer, 2009.

[25] L. Knudsen, G. Leander, A. Poschmann, and M. J. B. Robshaw. PRINTCIPHER: A Block Cipher for IC Printing. In *Cryptographic and Embedded Systems - CHES 2010*, volume 6225 of *LNCS*, pages 16–32. Springer, 2010.

[26] S. Kölbl, G. Leander, and T. Tiessen. Observations on the SIMON block cipher family. Cryptology ePrint Archive, Report 2015/145, 2015. http://eprint.iacr.org/.

[27] T. Lepoint and M. Naehrig. A comparison of the homomorphic encryption schemes FV and YASHE. In D. Pointcheval and D. Vergnaud, editors, *AFRICACRYPT 2014*, volume 8469 of *LNCS*, pages 318–335. Springer, 2014.

[28] A. Shahverdi, M. Taha, and T. Eisenbarth. Silent Simon: A Threshold Implementation under 100 Slices. Cryptology ePrint Archive, Report 2015/172, 2015. http://eprint.iacr.org/.

[29] P. Yalla and J.-P. Kaps. Lightweight Cryptography for FPGAs. In *Reconfigurable Computing and FPGAs, ReConFig '09*, pages 225–230, December 2009.