

INFORMATION SOCIETY TECHNOLOGIES (IST) PROGRAMME

Project n°: FP6-IST-033563

SMEPP



SMEPP

Secure Middleware for Embedded Peer-to-Peer Systems

WP5.1

***D5.1.1 Specification of Secure
Instruction Sets for EP2P Devices***

Author(s): TUG

Status - Version: Final Revision

Date: December 10, 2007

Distribution - Confidentiality: Public

Code: SMEPP – D5.1.1 Specification of Secure Instruction Sets
for EP2P Devices

Disclaimer

This document contains material, which is the copyright of certain SMEPP contractors, and may not be reproduced or copied without permission. All SMEPP consortium partners have agreed to the full publication of this document. The commercial use of any information contained in this document may require a license from the proprietor of that information.

The SMEPP Consortium consists of the following companies:

Participant no.	Participant name	Participant short name	Country
1 (Co-ordinator)	Universidad de Málaga	UMA	Spain
2	Tecnatom, S. A.	TEC	Spain
3	Technische Universität Graz	TUG	Austria
4	Siemens AG	SIEM	Germany
5	Valtion Teknillinen Tutkimuskeskus	VTT	Finland
6	Università di Pisa	UPI	Italy
7	Telefónica I+D	TID	Spain
8	Institute for Infocomm Research	I2R	Singapore

The information in this document is provided “as is” and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

Document Revision History

Date	Issue	Author/Editor/Contributor	Summary of main changes
August 16, 2007	0.1	Stefan Tillich	Initial version
August 17, 2007	0.2	Stefan Tillich	Implementation Options for Cryptographic Algorithms
August 19, 2007	0.3	Stefan Tillich	Instruction Set Extensions for Cryptography, Proposed Extensions for EP2P Systems, Protection against Side-Channel Attacks
August 29, 2007	0.4	Stefan Tillich	Cleanup of document
December 6, 2007	0.5	Stefan Tillich	Support for AES on 8-bit architectures
December 10, 2007	0.6	Stefan Tillich	Extend section on support for AES on 8-bit architectures

Contents

1	Overview	5
2	Implementation Options for Cryptographic Algorithms	5
2.1	Application-Specific Integrated Circuit (ASIC)	7
2.2	Application-Specific Instruction Set Processors (ASIPs)/Application Domain-Specific Processors (ADSPs)	8
2.3	General-Purpose Processor with Coprocessor (GPP+COP)	8
2.4	General-Purpose Processor with Instruction Set Extensions (GPP+ISE)	9
2.5	General-Purpose Processor (GPP)	10
3	Instruction Set Extensions for Cryptography	11
3.1	Support for Public-Key Cryptography	11
3.1.1	Integer Modulo Arithmetic	12
3.1.2	Non-Integer Arithmetic	12
3.2	Support for Secret-Key Cryptography	14
3.2.1	General Permutations	14
3.2.2	Broad Algorithm Support	14
3.2.3	Specific Algorithm Support for the AES	15
3.2.4	Synergies of Support of Secret-Key Algorithms and Public-Key Algorithms	16
4	Proposed Extensions for EP2P Systems	18
4.1	Cryptography Instruction Set Support for 32-bit Microprocessors	18
4.1.1	Instruction Set Extensions for ECC	18
4.1.2	Instruction Set Extensions for AES	19
4.2	Cryptography Instruction Set Support for 8-bit Microcontrollers	21
4.2.1	Instruction Set Extensions for ECC	21
4.2.2	Instruction Set Extensions for AES	22
5	Protection against Side-Channel Attacks	26

1 Overview

As described in the SMEPP deliverables D1.2 “Security Requirements of EP2P Applications” and D4.2 “Security services and primitives for EP2P systems”, the basis of SMEPP middleware security will be laid by strong cryptographic algorithms. This deliverable deals with a rather new implementation approach for cryptographic algorithms focused on the hardware/software co-design paradigm. The basic idea of this approach is to equip a microcontroller/microprocessor with custom instructions and architectural enhancements to ease the burden of cryptographic processing. This solution of extending the original instruction set architecture (ISA) of the microcontroller/microprocessor is hence denominated *Instruction Set Extension (ISE)*.

In Section 2 we provide an overview of the principal approaches regarding the implementation of cryptographic primitives. Subsequently, we will discuss the most important research results regarding ISE for cryptography in Section 3. The extensions deemed most beneficial for implementation on EP2P devices will then be described in detail in Section 4. A short discussion on side-channel attacks in relation to cryptography ISE is provided in Section 5.

2 Implementation Options for Cryptographic Algorithms

This section will give an overview of the general approaches for implementing cryptographic algorithms. Subsequently, all options will be evaluated in regard to constrained embedded systems, which constitute the primary target devices for the SMEPP middleware.

Implementations of cryptographic algorithms can be classified into five general categories. The following list cites them with decreasing amount of dedicated hardware.

- Application-specific Integrated Circuit (ASIC)
- Application-specific Instruction Set Processor (ASIP) and Application-domain-specific processor (ADSP)
- General Purpose Processor with Co-processor (GPP+COP)
- General Purpose Processor with Instruction Set Extensions (GPP+ISE)
- General Purpose Processor (GPP)

An *application-specific integrated circuit (ASIC)* consists only of hardware dedicated to one or several specific algorithms. *Application-specific instruction set processors (ASIPs)* and *application domain-specific processors (ADSPs)* are processors with an instruction set architecture (ISA) specifically tuned for a concrete application or application domain, which may also feature additional optimizations of the microarchitecture. A *coprocessor (COP)* can be attached to a *general-purpose processor (GPP)* in order to enhance the performance of specific algorithms. *Instruction set extensions (ISE)* feature an even tighter coupling of application-specific hardware and the general-purpose processor by providing new instructions in addition to the original ISA. The last design option is an unmodified general-purpose processor where all algorithms are executed using the defined ISA. All these approaches have different characteristics regarding *performance, energy efficiency, flexibility, and cost*.

The border between co-processor and instruction set extension is not completely sharp as co-processor solutions might also be accompanied by the addition of custom instructions (e.g. for controlling the coprocessor's functionality). A good criterion to categorize a specific solution is to look at the control mechanism of the cryptographic computations. If this control is mainly external to the processor pipeline, the solution is rather a coprocessor. A typical pattern for a coprocessor usage just consists of a few simple steps: Configure coprocessor functionality (e.g. select encryption or decryption), load input data, start operation, and retrieve results. On the other hand, if the pipeline retains fine-grain control over the cryptographic computations, the solution is rather an instruction set extension. Typically, the custom instructions perform relatively small steps of the algorithm (e.g. part of a round transformation of a block cipher).

On desktop systems, performance is measured in terms of throughput and is traditionally the main criteria in cryptographic systems (i.e. the faster the better). This stance is especially typical for cryptographic software for mainly two reasons. Firstly, resources like working memory, program memory, and available energy are practically unbounded. Secondly, although the speed and execution parallelism of desktop processors have been increasing steadily, the performance of cryptographic software continues to lag behind the performance of network connections. For example, the throughput for the AES block cipher on a modern desktop CPU currently peaks in the 1-2 Gbit/s range [82, 85], while Ethernet has already reached throughputs of 10 Gbit/s [72], with work already commencing on speeds up to 100 Gbit/s [71].

However, in embedded systems throughput is not the single goal of optimization. These systems normally have very limited memory and energy resources which need to be used scarcely by a cryptographic implementation (and in fact by any application running on the embedded device). The most important goal is to reach an "adequate" throughput sufficient for the particular application at hand with minimal use of memory and energy resources. Furthermore, as most embedded devices must be inexpensive, the additional cost for cryptography must henceforth be low. Another important aspect for embedded systems is flexibility and scalability of cryptographic processing.

Flexibility refers to a system's capability to process a variety of tasks from an application domain, e.g. support of different ECC parameters, or even from different applications domains, e.g. multimedia, digital signal processing, and cryptography. The required degree of flexibility is highly dependent on the intended application of the system. While some embedded systems with fixed processing requirements (e.g. within household appliances) have a limited need for flexibility, other systems (e.g. multi-purpose smart-cards, cellphones, PDAs) may require a very high flexibility.

Scalability is concerned with the provision of different variants of a specific cryptographic algorithm. A typical example is a block cipher with a varying key size and number of rounds, as it is offered by most modern block ciphers. A scalable solution can be adapted to the use of different key sizes, which offer a varying degree of security.

Energy efficiency refers to the amount of processed data in relation to the required energy. As a general rule, energy efficiency becomes better with a rising degree of dedicated hardware. However, this is only true within the domain for which the system is built. If, e.g. an ASIP is forced to do general-purpose processing, its energy efficiency may fall well below that of a GPP. Energy efficiency is strongly related to performance, as fast processing of an algorithm also reduces the total amount of energy spent on the task [49].

When considering cost, one has to distinguish different contributing factors; mainly design cost

and implementation cost. Design cost refers to the cost for building or purchasing the solution. While GPPs, GPPs with ISE and GPPs with co-processors may be readily available as intellectual property (IP) cores, ASIP/ADSPs and ASICs normally require a greater design effort. Implementation cost refers to the cost for realizing the system. Solutions involving GPPs may already be available as relatively cheap chips, whereas application-specific solutions will most likely require custom implementation by the system manufacturer.

In summary, embedded devices require cryptographic implementations which offer good properties in the following regards:

- Throughput
- Memory usage (both working memory and program memory)
- Flexibility/Scalability
- Energy consumption
- Cost

Finding a satisfying mix of all these properties is a very difficult task. Figure 1 gives a rough overview of the design space for cryptographic systems. The five key characteristics are also shown for the different design approaches. It is however important to note, that this assessment is only a typical trend and is not an absolute measure. It may well be that a specific GPP with ISE may outperform a GPP with coprocessor or that the cost of a specific ASIP is higher than that of an ASIC.

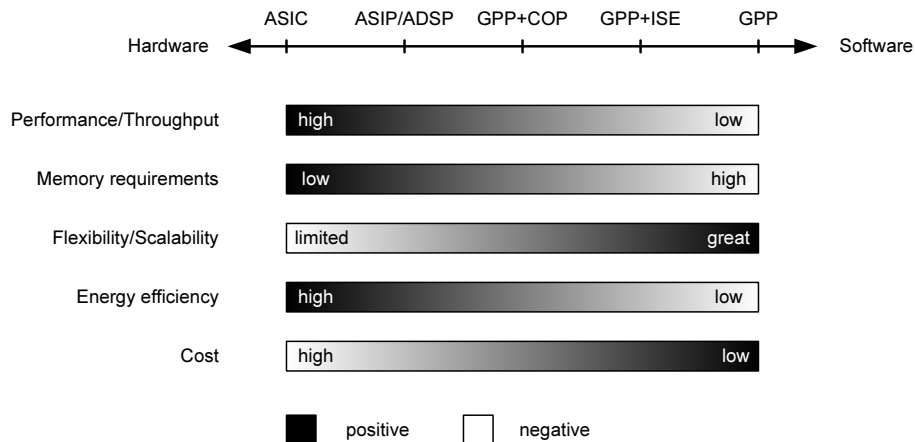


Figure 1: Hardware/software systems for cryptographic algorithms

2.1 Application-Specific Integrated Circuit (ASIC)

ASICs are custom-designed and custom-built chips which comprise a dedicated functionality. They are usually added to a system to satisfy performance demands and/or to increase the energy efficiency for a specific workload. There have been numerous publications about ASIC implementation of cryptographic algorithms. Fabricated ASICs for public-key algorithms (e.g., [8])

and secret-key algorithms (e.g., [61, 93, 113]) have been presented. Furthermore there exist high-speed implementations of hash functions (e.g., [24]). In addition, some ASICs have been designed to cater for complete security protocols like IPSec [62].

For embedded systems the prospect of increased energy efficiency is a very big advantage of ASIC implementation. However, the stringed development and implementation cost of cryptographic ASICs is contrary to the low-cost criterion. Furthermore, an additional chip could lead to an unacceptable enlargement of the embedded device. Multi-chip packaging could be used to circumvent space restrictions, but this would again increase the cost of the device. Therefore, cryptographic ASICs are no optimal choice for most embedded devices.

2.2 Application-Specific Instruction Set Processors (ASIPs)/Application Domain-Specific Processors (ADSPs)

Two prominent examples of ASIP architectures for symmetric cryptography are the Cryptonite processor [16, 99] and the CryptoManiac processor [137, 132]. For asymmetric algorithms there have also been proposed architectures [27]. In [77], a processor for both symmetric and asymmetric cryptography is presented. A methodology for designing ASIPs through gradual refinement has been presented in [109, 110].

Goodman et al. presented a reconfigurable cryptographic processor which integrated polynomial operations into an integer modular multiplier [43, 44]. The processor was denoted *Domain Specific Reconfigurable Cryptographic Processor (DSRCP)*. It includes a bit-serial multiplier for operands of up to 1,024 bits, which can be adapted in 32-bit steps to the actual operand size. The design of the DSRCP is described in detail in Goodman's Ph.D. thesis [45].

Another interesting architecture is the PAX processor [34, 39]. Although labeled "cryptographic" processor, PAX also features a number of general-purpose RISC instructions in addition to instructions for cryptography. A unique feature of PAX is its datapath scalability, which can be configured to a 32-bit, 64-bit, or even a 128-bit width.

ASIPs normally offer greater flexibility than ASICs and there is also a broader tool support available for their development. However, compared to a general-purpose processor, the scope of applications is still narrow. The goal of the SMEPP project is to facilitate development of secure embedded peer-to-peer applications by means of a suitable middleware framework. As diverse applications are to be supported, it can be assumed that the targeted devices must be equipped at least with a general-purpose processor (GPP). Therefore, it is necessary to solve the problem of cryptographic processing in conjunction with the GPP, which stresses solutions like GPP+COP, GPP+ISE, and GPP (pure software).

2.3 General-Purpose Processor with Coprocessor (GPP+COP)

A large number of designs and implementations of cryptographic coprocessors have been reported in the literature. As practically every cryptographic design can be used as coprocessor by adding some control logic, these solutions can vary considerably in their approach. The spectrum of functionality ranges from basic modular arithmetic to full-blown ASIPs in the form of a coprocessor. Normally the designs are specialized to either symmetric cryptography [69, 129, 32, 68, 104] or asymmetric cryptography [139, 31, 1].

Some publications have shown that the use of coprocessors can suffer from a serious disadvantage. The processing speed of the coprocessor can be severely limited by the data rate of its connection to the general-purpose processor. In [70] and [111] it was shown for an AES coprocessor that even dedicated high-speed interfaces may well be nearly two orders of magnitude slower than the coprocessor itself. This result provides a strong argument for a tighter coupling of the custom hardware to the processor's pipeline whenever there is the need for a substantial interaction between GPP and cryptographic hardware.

2.4 General-Purpose Processor with Instruction Set Extensions (GPP+ISE)

Instruction set extensions for cryptography are a hybrid approach between pure-software solutions and coprocessor systems. In the current literature there is a distinction between two basic methodologies for selecting an efficient set of custom instructions. The first approach leaves instruction selection up to the designer while the second approach relies on automatic selection via benchmarking of program code which is "typical" to the targeted application domain. Different approaches for automatic selection have been described in [106, 22, 23, 76]. The Swiss Federal Institute of Technology Lausanne (EPFL), partly in cooperation with the University of California, Irvine has also released a number of publications regarding automatic generation of instruction set extensions [102, 103, 5, 13, 12].

On the one hand, automatic selection of custom instructions can yield substantial improvements when the benchmarked code represents the requirements of the application domain in a sufficient manner. But in the general case, where the designer is not familiar with the application domain, this automatic selection may well fall short of the optimal solution. A good example of this problem is the case of extensions for Elliptic Curve Cryptography, where a relatively slow pure software solution can be sped up to match the performance of the best algorithm, if both are using instruction set extensions [119]. If just the fastest algorithm is used for benchmarking, the resulting extensions may not be able to deliver the optimal performance improvement for the application domain.

Manual design of extensions takes more time than automatic selection, but the results are more likely to be nearer to the optimal solution for the application domain. Koç has examined the suitability of the Intel MMX instructions for implementing public-key cryptography [19]. Dhem proposed support for long integer modulo multiplication on ARM architectures [25]. Großschädl examined support for the same operation for the MIPS32 [48]. Further refinements of the instructions have been given in [54, 51].

Nahum et al. suggested to add support for arithmetic in $GF(2^{155})$ to RISC processors [95]. Koç et al. proposed bit-level and word-level algorithms for Montgomery multiplication over binary extension fields [20]. They found that a basic operation for word-level algorithms was the multiplication of two binary polynomials and suggested to add support for multiplication of two word-sized binary polynomials to a processor. Savaş et al. gave "word-level" algorithms for Montgomery multiplication in $GF(p)$ and $GF(2^m)$ along with a multiplier suited for both types of fields [108].

Support for multiplication of binary polynomials has been investigated by Großschädl et al. in the context of a 16-bit RISC processor [52]. An extensive treatment of extensions for arithmetic in $GF(p)$ and $GF(2^m)$ was given by Großschädl et al. in [57]. A total of five custom instructions was proposed to support word-level algorithms over both finite fields.

Fiskiran and Lee have investigated instruction set extensions for binary extension fields in relation

to varying wordsize and superscalar processing [33]. Eberle et al. have targeted an 8-bit micro-controller for integrating support for ECC over binary extensions fields $GF(2^m)$ [29]. Support for so-called *Optimal Extension Fields (OEFs)* was investigated in [55].

Tillich et al. investigated low-cost support of $GF(2^m)$ operations [119]. Großschädl et al. also investigated synergies of digital signal processing workloads with cryptographic operations [56]. The group of Ruby Lee has also done work on architectural support for permutation instructions [138, 114, 81, 88].

Burke et al. [17] have examined several AES candidates in order to identify worthwhile additional instructions. Fiskiran and Lee's work [37] has shown that extended addressing modes can lead to substantial performance gains for cryptographic algorithms. The same authors discuss support for table lookup to speed up symmetric ciphers [35] and have shown the value of instruction set extensions for ECC over binary finite fields for processors with large word sizes and superscalar execution [33].

Nadehara et al. [94], Bertoni et al. [11], and Tillich et al. [124, 121] have proposed extensions for implementation of the Advanced Encryption Standard (AES).

Important providers of microprocessors have included support for security into their products, e.g., ARM *SecurCore* [3], MIPS *SmartMIPS* [91], STMicroelectronics *SmartJ* [116], and VIA *PadLock* [131]. The IA-64 architecture by Intel and Hewlett-Packard as implemented in the Itanium processor has also been optimized to address cryptographic needs [73, 74]. Unfortunately, as all companies try to retain competitive advantages, there is not much information about the implementation of these cryptographic extensions available to the public.

2.5 General-Purpose Processor (GPP)

Efficient software implementations of cryptographic algorithms are the subject of a large number of publications. In the field of symmetric cryptography, the selection process for the Advanced Encryption Standard (AES) by the U.S. National Institute of Standards and Technology (NIST) has drawn much attention of the cryptographic research community. There have been numerous publications during the AES selection process which have focused on efficient software implementations of the candidate algorithms, e.g., [112, 2, 42, 63].

After the selection of Rijndael as the AES algorithm in 2001 [97], efficient software implementations on various platforms have received considerable attention [10, 4, 41]. The AES algorithm is assumed to be in widespread use throughout the next decades.

The reduced resource demand of resources of Elliptic Curve Cryptography (ECC) [89, 78] in comparison to traditional asymmetric cryptosystems like RSA [107] and ElGamal [30] makes it an attractive alternative for embedded and constrained devices. ECC has been the topic of a great number of publications regarding both efficient algorithms, e.g., [133, 135, 64, 83, 98], and implementations on different platforms, e.g., [136, 14, 38, 59, 65, 66, 67]. An important property of ECC is the large degree of freedom offered to the system designer. On the one hand, mathematical properties of different elliptic curves and underlying fields can be exploited to allow efficient implementations on specific platforms, e.g., [136, 59]. On the other hand, this can pose problems when a device needs to support various sets of parameters.

3 Instruction Set Extensions for Cryptography

More than a decade ago, Nahum et al. proposed methods to bring cryptographic performance up to Gigabit/s network speeds [95]. Their idea was to propagate the algorithm agility and scalability offered by modern security protocols through appropriate implementation measures. Their three main strategies encompassed:

- Design of new (more efficient) cryptographic algorithms
- Increase of processor parallelism
- Processor enhancements

For the last point—processor enhancements—the authors envisioned “to do classic RISC processor (or coprocessor) design on a large set of cryptographic software implementations”. Nahum et al. proposed to “add to a standard RISC instruction set only those instructions which significantly improve the overall performance of the test suite”. They also identified three types of operations which are important for cryptographic software but are poorly supported on typical RISC processors:

- Operations on sub-wordsize units
- Operations on super-wordsize units
- Operations of groups different from integers

While sub-wordsize operations are typical for symmetric algorithms, public-key cryptography makes heavy use of super-wordsize operands. Non-integer arithmetic is found in both types of cryptographic algorithms.

The largest part of the available literature deals with the enhancement of 32-bit RISC processors with cryptography support. Only some publications target architectures of a different wordsize.

3.1 Support for Public-Key Cryptography

Public-key primitives are slow and require considerable resources. They are used rather infrequently in most scenarios (e.g., key refreshing in a regular interval). When it comes to implementation in embedded systems, public-key cryptography tends to pose problems of feasibility and latency.

The literature on public-key extensions is divided into two broad categories: Support for (long) integer module arithmetic and support for (long) non-integer arithmetic. The first category deals with efficient implementation of public-key algorithms based on the *Integer Factorization Problem (IFP)* (like RSA [107]) or the *Discrete Logarithm Problem (DLP)* in integer groups (like ElGamal [30], DSA with integer groups [96]) or *Elliptic Curve Discrete Logarithm Problem (ECDLP)* for curves over prime fields (DSA with elliptic curves over prime fields [96]). The second category investigates support for public-key algorithms over alternative fields, most importantly *binary extension fields*. The most common systems are ECC-based with the underlying problem of ECDLP for curves over binary extension fields (like DSA with elliptic curves of binary extension fields [96]).

3.1.1 Integer Modulo Arithmetic

Koç has examined the suitability of the Intel MMX multi-media extensions for implementing public-key cryptography [19]. He concluded that these multi-media extensions bore not benefit for such cryptographic workloads. The main problem was the sole availability of signed integer operations, while the PCK algorithms require operations on unsigned integers. Dhem proposed ARM7M support for a $(32 \times 32 + 32 + 32)$ -bit multiply-accumulate operation for long integer modulo multiplication [25]. He also described a $(8 \times 32 + 8 + 32)$ -bit multiply-accumulate (MAC) unit, which could be used to execute the required operation in four clock cycles. Großschädl examined support for the same operation for the MIPS32 architecture [48]. Cycle-accurate simulation of a SystemC model of a MIPS32 core yielded a speedup factor of 2 for multiple-precision integer modulo multiplication using the Montgomery method [92]. The algorithm considered following the *coarsely integrated operand scanning (CIOS)* method [21]. A concrete implementation of a multiply-accumulate unit has been given by Großschädl et al. in [50]. Further refinements of the instructions and use of *finely integrated operand scanning (FIOS)* and *finely integrated product scanning (FIPS)* methods have been given in [54] and [51] respectively.

Großschädl et al. have investigated the energy characteristics of different algorithms for long integer modular arithmetic [49]. Using power simulation with the *JouleTrack* tool [115] they showed that the Karatsuba and Comba multiplication with Montgomery reduction (KCM method) required the least energy of all surveyed algorithms. A follow-up work demonstrated that the FIPS methods outperforms the KCM method in terms of pure performance for typical operand sizes used in state-of-the-art cryptography [58].

3.1.2 Non-Integer Arithmetic

Another line of work has dealt with support of operations in arithmetic structures other than integers. The main focus has been set on binary finite extension fields, commonly denoted as $\text{GF}(2^m)$ or \mathbb{F}_{2^m} . Nahum et al. already suggested to add support for arithmetic in $\text{GF}(2^{155})$ to RISC processors, without giving a concrete implementation [95]. Koç et al. proposed bit-level and word-level algorithms for Montgomery multiplication over binary extension fields [20]. They found that a basic operation for word-level algorithms was the multiplication of two binary polynomials. The authors suggested to add support for multiplication of two word-sized binary polynomials to processors in order to speed up the proposed algorithms. The name given to this operation was MULGF2. Previously, Drescher et al. had proposed to integrate support for $\text{GF}(2^m)$ multiplication into a conventional integer multiplier [26]. Their architecture is able to perform (17×17) -bit integer multiplication and finite field multiplication on fields of degree eight or less. It has been conceived for supporting error correction codes on Digital Signal Processors. Public-key cryptography requires finite field degrees larger by more than an order of magnitude. The architecture proposed by Drescher et al. does not scale well with increasing degree and therefore it is not directly useable for public-key algorithms.

Bucci presented ideas for a “dual-mode” modular multiplier, which could handle integers and binary polynomials, having cryptography as primary field of application [15]. Goodman et al. presented a reconfigurable cryptographic processor which integrated polynomials operations into an integer modular multiplier [43, 44]. The processor was denoted *Domain Specific Reconfigurable Cryptographic Processor (DSRCP)*. It includes a bit-serial multiplier for operands of up to 1,024 bits, which can be adapted in 32-bit steps to the actual operand size. The design of the

DSRCP is described in detail in Goodman's Ph.D. thesis [45]. Savaş et al. gave "word-level" algorithms for Montgomery multiplication in $GF(p)$ and $GF(2^m)$ along with a multiplier suited for both types of fields [108]. The authors showed that such a "unified" multiplier just required the availability of adder cells (so-called "dual-field" adders) which could suppress the carry for $GF(2^m)$ operations. The resulting hardware implementation can handle operands of arbitrary size. Großschädl [47] and Au et al. [7] presented optimized components for unified multipliers.

Support for the `MULGF2` instruction for $GF(2^m)$ has been investigated by Großschädl et al. in the context of a 16-bit RISC processor [52]. The resulting speedup for polynomial multiplication has been shown to range between a factor of 6 to 10. A low-power multiply-accumulator design for 32-bit processors has also been presented by the same authors [53]. This unit could perform $(32 \times 32 + 64)$ -bit multiply-accumulate operations for signed and unsigned integers as well as binary polynomials in a single clock cycle at a size of about 10,000 gate equivalents (GEs). An extensive treatment of extensions for arithmetic in $GF(p)$ and $GF(2^m)$ was given by Großschädl et al. in [57]. A total of five custom instructions was proposed to support word-level algorithms over both finite fields.

Fiskiran and Lee have investigated instruction set extensions for binary extension fields in relation to varying wordsize and superscalar processing [33]. They were able to demonstrate a potential speedup of 22.4 for ECC point multiplication. Furthermore, the authors showed that—when it comes to accelerating public-key algorithms with architectural enhancements—a doubled word-size is to be preferred over a doubling of the execution parallelism. Eberle et al. have targeted an 8-bit microcontroller for integrating support for ECC over binary extensions fields $GF(2^m)$ [29]. Their implementation of an ECC point multiplication over $GF(2^{163})$ on an enhanced ATmega128 required about 2.3 million clock cycles and was considerably faster than implementations of 160-bit ECC over $GF(p)$ and 1,024-bit RSA, which offer equivalent security.

Support for so-called *Optimal Extension Fields (OEFs)* was investigated in [55]. An OEF is a field of the form $GF(p^m)$, where p is a pseudo-Mersenne prime of the form $p = 2^n - c$, which fits into a single register of the target processor platform. Two custom instructions for the MIPS32 architecture were demonstrated to lead to a speedup of about 1.8 for ECC scalar multiplication over such a field.

Tillich et al. investigated low-cost support of $GF(2^m)$ operations, which could be integrated into the integer adder of RISC processors [119]. The authors demonstrated that ECC point multiplication over $GF(2^{191})$ accommodating an arbitrarily chosen reduction polynomial could be sped up by a factor of up to 10. For a single fixed reduction polynomial, the speedup could reach nearly 2. Other work by has dealt with synergies of digital signal processing workloads with cryptographic operations and the comparison of two common RISC architectures (MIPS32 and ARMv5TE) for their suitability to accommodate architectural enhancements [56]. In this regard the MIPS32 architecture showed slight advantages over its competitor.

Eberle et al. have described and evaluated a number of instructions for public-key cryptography and have evaluated the performance gains for RSA and ECC (over both prime fields and binary extension fields) [28]. These enhancements are contained in the UltraSPARC T2 high-performance CPU from Sun Microsystems [117].

3.2 Support for Secret-Key Cryptography

Secret-key primitives are more light-weight to implement than public-key algorithms and typically several orders of magnitude faster. In turn, they are used much more frequently (e.g., for the encryption and authentication of messages). In embedded systems, secret-key algorithms pose challenges of reaching adequate performance under the minimal use of resources.

The existing literature can be split into four categories:

- Support for general permutations
- Broad support for secret-key algorithms
- Dedicated support for AES
- Synergies of secret-key and public-key support

Unlike public-key support, the field of secret-key support bears much more synergies between extensions for general-purpose processors and concepts used in dedicated cryptographic processors (i.e. ASIPs/ADSPs for cryptography).

3.2.1 General Permutations

Shi and Lee were the amongst the first to propose instruction set extensions for secret-key cryptography [114]. They suggested to provide architectural support for arbitrary bit-level permutations; an operation perceived to be beneficial for symmetric ciphers in order to increase diffusion. The authors presented two instructions for performing fast arbitrary permutations of word-sized values: While `PPERM3R` can be used for permutations with repetitions, the `GRP` instruction is also useable for permuting operands exceeding the wordsize. In a follow-up work, Lee et al. proposed a lower-cost variant of the `PPERM3R` instruction (denoted `PPERM`) [81]. Furthermore, the `CROSS` instruction was presented, which performs permutations based on the principles of a Benes network. It has also been shown that the same functionality can be achieved with the `OMFLIP` instruction, which exploits the properties of an omega-flip network. The advantage of `OMFLIP` is a much reduced implementation cost. Another strategy for performing permutations was discussed by McGregor and Lee [88]. The basic idea was to combine subword-level permutation (`SWPERM` instruction) with bit selection and permutation within subwords (`SIEVE` instruction) in order to achieve arbitrary bit-level permutations.

3.2.2 Broad Algorithm Support

Burke et al. conducted a detailed performance analysis of eight symmetric ciphers (seven block ciphers and one stream cipher) [17]. An important conclusion was that modern block ciphers showed a significant degree of parallelism which can be exploited in suitable hardware architectures. Architectural enhancements have also been proposed with the goal of a broad support of current and (possibly) future symmetric ciphers. This included a 16-bit modular multiplication (with a fixed modulus), several rotate instructions, support for byte substitution using arbitrary tables and a bit-level permutation instruction similar in functionality to those investigated by the

group around Ruby Lee. Follow-up work by Wu et al. lead to the development of the *CryptoManiac* cryptographic coprocessor [137]. This processor is a *Very Long Instruction Word (VLIW)* machine which is able to execute up to four instructions per cycle. A distinguishing feature is the processor's ability to combine short-latency instructions (e.g., bitwise logical and arithmetic instructions) to be executed in a single cycle. This is supported by *CryptoManiac*'s ability to handle instructions with up to three source operands.

Fiskiran and Lee's work has shown that extended addressing modes can lead to substantial performance gains for cryptographic algorithms [37].

Another cryptographic processor with a VLIW architecture is the *Cryptonite* processor [16]. Similar to *CryptoManiac*, *Cryptonite*'s design is based on analysis of secret-key ciphers, but also on hash algorithms. *Cryptonite* includes two 64-bit datapaths supporting operations for cryptographic algorithms (e.g., byte permutation and rotation). Each datapath is complemented by 32 KB of dedicated local memory, which can be accessed with different address and data widths and which can be used to implement arbitrary substitution operations. The work of Ravi et al. has targeted the 32-bit extensible processor *Xtensa* from Tensilica [106]. The authors investigated the application of automatic generation of instruction set extensions for different cryptographic algorithms. The use of dedicated hardware lookup tables for the acceleration of different symmetric ciphers has also been investigated by Fiskiran and Lee [36]. Their solutions consists of a number of parallel lookup tables with 256 entries of 32-bit words each.

3.2.3 Specific Algorithm Support for the AES

Irwin and Page have investigated extensions for AES in the context of the PLX general-purpose RISC processor with a 128-bit datapath [75]. However, the presented solutions do not scale down to architectures with a smaller wordsize. Nadehara et al. and Bertoni et al. focused on the AES and proposed different extensions for general-purpose processors [94, 11]. Nadehara et al. suggested a single custom instruction to calculate the AES round transformations for a single State byte in a dedicated functional unit. Effectively, this maps the so-called round lookup (T lookup), which is commonly used for AES software implementations on 32-bit platforms, into hardware (*AESENC* instruction). In contrast, Bertoni et al. started from an AES software implementation which uses only S-box lookup tables and a transposed State representation, which has been described in [10]. They propose both byte-oriented and word-oriented extensions for a 32-bit Intel StrongARM processor. The byte-oriented extension have similar functionality as those of Nadehara et al., but they offer more flexibility (e.g., additional support for the key expansion). The word-oriented extensions allow for increased performance at a higher implementation cost.

Interesting insights have also been offered by the works of Schaumont et al. [111] and Hodjat et al. [70], who investigated the integration of an AES coprocessor into the 32-bit SPARC V8-compatible Leon2 processor. They showed that communication bottlenecks can easily become the dominant factor in overall system performance.

In [124], Tillich et al. focused on increasing memory efficiency, as memory is usually a very scarce resource in embedded systems. Furthermore, the authors demonstrated on their experimental platform that the memory subsystem can quickly become the dominating bottleneck. Therefore, the T-lookup approach with several 1 KB or 4 KB tables, which is usually the fastest software implementation option on desktop processors, can become several times slower than a memory-preserving "S-box lookup" strategy, when the available memory resources are insuffi-

cient. Moreover, several other factors can make T-lookup unattractive, e.g., the increased energy consumption of memory accesses and vulnerability against cache-based timing attacks.

The solution presented in [124] maps the S-box lookup for a single byte into a dedicated functional unit. At the minimal hardware cost of a few hundred gates, the need for memory accesses for the AES computation could be completely removed. The resulting performance increase reached up to 30 % with code size shrinking by up to 43 %.

A more performance-oriented investigation has been performed by Tillich et al. in [121]. Tackling the main performance bottleneck—the MixColumns transformation—with a dedicated instruction, a speedup of nearly 4.5 could be achieved in combination with the instructions already presented in [124]. The overall hardware cost of this solution was less than 1,000 gate equivalents.

More aggressive optimization allowed a better utilization of the 32-bit datapath. Careful analysis of the AES algorithm led to a set of high-performance instructions, which could be integrated into the processor without any significant changes of the underlying architecture. The resulting performance gain reached an factor of up to 10, i.e. improvement by an order of magnitude. Moreover, the code size could be reduced by up to 80 % with these extensions. The implementation cost ranges around 3 kGates.

A complete AES-128 encryption with a precomputed key schedule can be done in less than 200 clock cycles. Moreover, the extensions of Tillich et al. provide support for the key expansion, henceforth increasing the key agility of AES implementations. A complete encryption with on-the-fly key expansion can be performed in 255 clock cycles with a code of 65 instructions (i.e. a code size of 260 bytes).

Another advantage of these custom instructions is their eligibility for superscalar processing. For increased execution parallelism, the performance scales up linearly up to a 4-way superscalar processor, i.e. a 4-way superscalar processor (equipped with three instances of each of the proposed functional units) could reach a fourfold increase in performance (i.e. about 50 clock cycles per encryption). A further (but sub-linear) performance increase would occur with increased parallelism up to a 6-way superscalar processor. There the peak performance ranges at around 37 clock cycles per AES encryption.

Tillich et al. have proposed support for AES for the popular 8-bit Advanced Virtual RISC (AVR) architecture, which is found in many embedded devices, e.g., smart cards and sensor nodes [125].

3.2.4 Synergies of Support of Secret-Key Algorithms and Public-Key Algorithms

Synergies between instruction set support for public-key cryptography and secret-key cryptography have been investigated by Tillich et al. in two regards. On the one hand, they looked at the use of existing public-key extensions for secret-key algorithms. On the other hand, they tried to implement functional units which could support instructions for both kinds of cryptographic algorithms. In [120], Tillich et al. reused a subset of the public-key enhancements (for ECC over binary extension fields) from Großschädl et al. [57] to speed up AES implementations. This has been possible because the AES algorithm also employs arithmetic operations over binary extension fields. The multiplication of two binary polynomials realized by the `MULGF2` instruction has been used as a base to implement the relatively costly MixColumns transformation of AES. In this fashion it was possible to increase the performance of an optimized software implementation by up to 25 %. A limiting factor was posed by the greatly differing field size employed by the differ-

ent algorithms. While ECC uses operands of hundreds of bits in length, the AES transformations work in $GF(2^8)$ and small extensions thereof. Explicit support for $GF(2^8)$ has been integrated into an existing unified multiply-accumulate unit in order to increase the attainable speedup for AES [123]. At an additional cost of about 1.3 kGates, the AES execution could be accelerated by up to 43 %.

4 Proposed Extensions for EP2P Systems

The previous section has given an overview of the work performed on instruction set extensions for cryptography. The target devices for the SMEPP middleware will predominantly be constrained embedded devices, which will require reasonable cryptographic performance at a low cost. Therefore we envision future embedded devices to include a strong support for a “core set” of public-key and secret-key algorithms, which can be used to implement cryptographic protocols and to realize all necessary security assurances demanded by the application. Devices sharing support for this core set then have a cheap and effective means for secure communication. If necessary, communication with systems using other algorithms can still be achieved through software implementation.

The core set of cryptographic algorithms which we propose to support consists of Elliptic Curve Cryptography (ECC) on the public-key side and the Advanced Encryption Standard (AES) on the secret-key side. AES can be applied in different modes of operations, and can be used to provide confidentiality, integrity, and authenticity of data in an efficient manner (a discussion of applicable modes of operations has been provided in SMEPP deliverable D4.2 “Security services and primitives for EP2P systems”). On the other hand, ECC can be used to exchange keys and provide digital signatures.

4.1 Cryptography Instruction Set Support for 32-bit Microprocessors

Section 3 has shown the extensive body of work available on instruction set support for cryptography on 32-bit processors. In the following, we will describe the extensions which we deem the most beneficial for supporting our core set of cryptographic algorithms.

4.1.1 Instruction Set Extensions for ECC

The propositions for ECC extensions mainly center around various multiply and multiply-accumulate instructions for integers [25, 48, 50, 54, 51] and binary polynomials [20, 26, 52, 33]. Investigation of unified architectures for both kinds of operations are of particular interest [108, 57, 28].

We take the fairly extensive treatment from [57] as the basis for ECC support both over prime fields $GF(p)$ as well as over binary extension fields $GF(2^m)$. In this work, the authors propose a set of seven instructions with a small architectural enhancement in the form of a “wide” accumulator register. In their paper, Großschädl et al. target the MIPS32 architecture [90], which already features an accumulator in the form of the HI/LO register pair. They propose to create a wide accumulator by extending the size of the HI register by a eight guard bits, which allow to accumulate up to 256 multiplication results without overflow. In the generic description below we will denote this wide accumulator by ACCU.

MULTU *src1*, *src2* Performs an unsigned multiplication of the two source operands, writing the 64-bit result into the accumulator ACCU.

MADDU *src1*, *src2* Same as **MULTU**, but adding the multiplication result to the previous value in the accumulator ACCU.

M2ADDU *src1*, *src2* Unsigned multiplication of the source operands, followed by a doubling of the result and addition to the previous value in the accumulator *ACCU*.

ADDAU *src1*, *src2* Adds both source operands to the value in the accumulator *ACCU*.

SHA Shift the value in the accumulator *ACCU* right by 32 (shifting in zero from the left).

MULGF2 *src1*, *src2* Interprets the two source operands as binary polynomials and multiplies them, writing the product into the accumulator *ACCU*.

MADDF2 *src1*, *src2* Same as **MULGF2**, but adding (i.e. bitwise exclusive-or) the product to the previous value in the accumulator *ACCU*.

All these instructions can be implemented with the help of a so-called *unified* multiply-accumulate unit, e.g. [53]. With the help of these instructions, the field operations underlying the ECC algorithms can be sped up considerably. A complete point multiplication—which is the core operation of all ECC systems—can then be performed in less than 1.2 million clock cycles for GF(p) ($|p| = 192$) and less than 0.7 million clock cycles for GF(2^m) ($m = 191$) [57].

Though our primary concern is to speed up ECC over GF(p) and GF(2^m), these extensions can also be used to optimize other public-key algorithms like RSA or DSA. Furthermore these extensions can be used in digital signal processing [56].

4.1.2 Instruction Set Extensions for AES

The most important works on support for AES have been provided by Nadehara et al. [94], Bertoni et al. [11], and Tillich et al. [121]. Table 4 in [121] provides a comparison of the performance figures for AES of embedded general-purpose processors either equipped with extensions or a cryptographic coprocessor, cryptographic processors, and desktop processors. For brevity, Table 1 replicates Table 4 from [121].

The columns DPW and IW/DMRP in Table 1 give a general classification of the structure and resources used in the respective processor architecture. DPW denotes the width of the datapath, IW the issue width (i.e. the number of instructions which can be executed in parallel), and DMRP indicates the number of read ports from the data memory. The column LUTs/Size shows whether the processor includes dedicated hardware lookup tables, indicating their number and the size of a single table in bytes. The last two columns state the cycle count for a complete AES-128 encryption and decryption respectively.

Limiting the scope to singlescalar 32-bit processors, we can see from Table 1 that the best performance is delivered by the implementation of [121] (Leon2 + ISE (1/1)). Furthermore, the singlescalar performance of [121] beats the performance of the coprocessor solutions reported in [70] and [111] and it can even compete with desktop performance [86, 85]. The performance of the extensions of [121] can also be scaled up by increasing the parallelism of the processor, so that a 4-way 32-bit processor (Leon2 + ISE (4/1)), can outperform cryptographic processors with broader datapaths and/or dedicated hardware lookup tables.

Further advantages of the extensions of [121] include a simple integration into existing RISC architectures, small implementation cost. Implementation flexibility is preserved, as the AES can be implemented both with a precomputed key schedule (for higher performance) or on-the-fly

Table 1: AES-128 performance on different platforms

Platform	Reference	DPW	IW/DMRP	LUTs/Size	Encr.	Decr.
RISC-like	Fiskiran [36]	128	1/1	16/1,024	32	32
PLX-128	Irwin [75]	128	1/1	0/0	609	n/a
Alpha (8W+)	Burke [17]	64	8/4	4/1,024	99	n/a
Alpha (4W+)	Burke [17]	64	4/2	4/1,024	164	n/a
Cryptonite	Oliva [99]	64	2/1	16/256	71	83
RISC-like	Fiskiran [36]	64	1/1	8/1,024	126	126
CryptoManiac	Wu [137]	32	4/1	4/1,024	90	n/a
Leon2 + ISE	Tillich [121]	32	4/1	0/0	51	51
RISC-like	Nadehara [94]	32	2/1	0/0	200	200
RISC-like	Fiskiran [36]	32	1/1	4/1,024	315	315
Xtensa + ISE	Ravi [106]	32	1/1	0/0	1,400	1,400
StrongARM	Bertoni [11]	32	1/1	0/0	311	n/a
Leon2 + ISE	Tillich [121]	32	1/1	0/0	196	196
Leon2 + COP (CPI)	Hodjat [70]	32	1/1	0/0	704	n/a
Leon2 + COP (MM)	Hodjat [70]	32	1/1	0/0	1,228	n/a
Leon2 + COP (MM)	Schaumont [111] ^a	32	1/1	0/0	1,494	n/a
Athlon 64	Matsui [85]	64	3/2	0/0	170	n/a
Pentium 4	Matsui [86]	32	3/1	0/0	251	n/a

^aPerformance calculated from time for encryption at 50 MHz.

key expansion (for increased key agility) and as the accelerated AES can be used in all modes of operation. Moreover, all key sizes of AES (128, 192, and 256 bit) can be supported, thus satisfying the requirement of scalability.

In the following we give a brief description of the extensions proposed by Tillich et al. In their paper, the authors described several sets of instructions; we refer here to the set which is denoted as “Advanced Word-Oriented AES Extensions with Implicit ShiftRows” given in Section 4.3 of [121]. This set consists of five instructions: Two for encryption, two for decryption and one for the key expansion. All instructions have a RISC-like instruction format with at most two input operands and one output operand.

SBOX4S src1, src2, dst Takes two bytes from each of the two source registers (1st and 3rd byte from src1, 2nd and 4th byte from src2), substitutes each byte according to the (forward) AES S-box, and writes the substituted bytes to the destination register dst.

ISBOX4S src1, src2, dst Same as SBOX4S, but uses the inverse S-box.

SBOX4R src, dst Substitutes all four bytes of the single source register src with the AES S-box, rotates the result by eight to the left and writes the result to the destination register dst.

MIXCOL4S src1, src2, dst Selects two bytes from each of the two source registers (1st and 2nd byte from src1, 3rd and 4th byte from src2), interprets these four bytes as an AES State column, applies the AES MixColumns transformation to this column and writes the result to the destination register dst.

IMIXCOL4S *src1*, *src2*, *dst* Same as MIXCOL4S, but uses the AES InvMixColumns transformation.

These instructions can be implemented in two dedicated functional units as shown in Figures 2 and 3 for the SPARC V8 architecture.

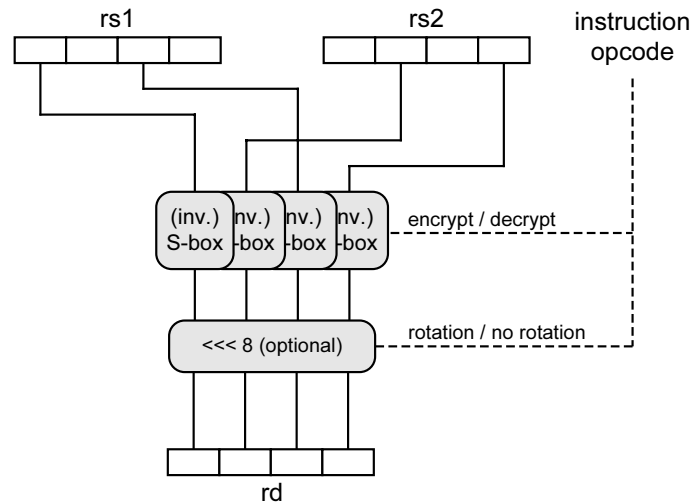


Figure 2: Implementation of the SBOX4S, ISBOX4S and SBOX4R instructions

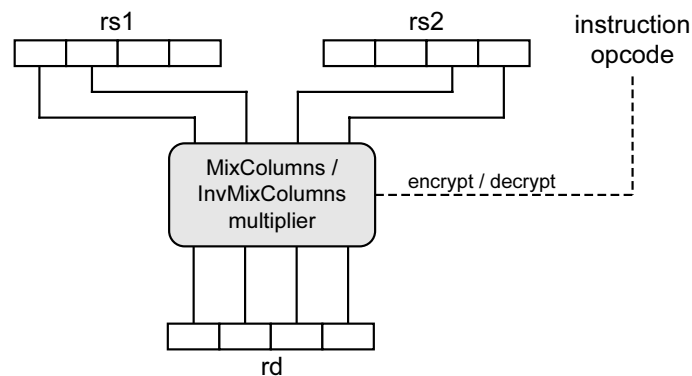


Figure 3: Implementation of the MIXCOL4S and IMIXCOL4S instructions

For the hardware implementation of the AES S-box, several options exist. An analysis of the existing options has been provided in [118]. Tillich et al. have implemented these functional units using the S-box of Canright [18] and the MixColumns multiplier of Wolkerstorfer [134], yielding in a total hardware overhead of about 3,000 gate equivalents.

4.2 Cryptography Instruction Set Support for 8-bit Microcontrollers

4.2.1 Instruction Set Extensions for ECC

As of now, there is little research on instructions for low-cost 8-bit microcontrollers. An exception are the works of Gura et al. on ECC over $GF(p)$ [60] and Eberle et al. on ECC over $GF(2^m)$ [29].

The proposed instructions are basically scaled down versions of the ones proposed for architectures with a larger wordsize. They make use of an accumulator of size 16 (*ACCU*), i.e. without guard bits. In [60, 29], the accumulator is formed by the destination register *dst* (lower part) and a dedicated register *XC* (upper part).

MULACC *src, dst* Performs unsigned multiplication of the source operand *src* with the contents of a fixed architectural register and adds to the product the previous contents of the destination register *dst* and the 8 high bits of the accumulator *ACCU*. The result is written to the accumulator.

MULX *src1, src2* Interprets the two source operands as binary polynomials, multiplies them and writes the result into a fixed register pair.

MULACCX *src1, dst* Same as *MULACC*, but with polynomial multiplication and addition.

4.2.2 Instruction Set Extensions for AES

We are not aware of any instruction set extensions for AES (or any other secret-key algorithm) for 8-bit platforms being proposed prior to the SMEPP project. In order to fill this gap, Tillich et al. have investigated possibilities for such support [125]. Their work proposes a set of six instructions (three for encryption and three for decryption) to increase the performance and to lower memory consumption of AES implementations.

The proposed instruction set extensions have been designed in regard to the modern Advanced Virtual RISC (AVR) architecture. AVR-compatible microcontrollers can be found in many embedded devices, e.g., smart cards and sensor nodes. A very nice feature of the AVR architecture is the provision of an abundance of 32 general-purpose registers, which present many opportunities for software optimization.

The custom instructions in [125] try to encompass transformations of a single AES round in a single invocation. The narrow 8-bit datapath is somewhat compensated for by mimicking the behavior of the AVR's native multiplication instructions. Although a single multiplication instruction can indicate only two arbitrary operand registers, two additional fixed registers (*R0* and *R1*) can be used implicitly. The instructions for AES use the same feature to maximize the number of performed transformations.

The two most important instructions for encryption encompass similar functionality. As the performed operations differ only slightly, they are denoted as *AESENC(1)* and *AESENC(2)*. The instructions perform the *SubBytes*, *MixColumns*, and *AddRoundKey* transformations explicitly, while *ShiftRows* is achieved by supplying the appropriate State bytes as operands. The only difference between the two instructions are the constants used for *MixColumns* multiplication. An invocation of *AESENC(1)* followed by *AESENC(2)* (using four State bytes at the beginning of the round as operands and two bytes from the round key preloaded into the registers *R0* and *R1*) produces two State bytes at the end of the round. An additional instruction *AESSBOX* excludes the *MixColumns* step and can be used for the final round as well as for supporting the key expansion. The instructions *AESDEC(1)* and *AESDEC(2)* and *AESINVSBOX* proposed for decryption have similar functionality involving the inverse round transformations.

AESENC(*x*) *src1, src2* Takes the two State bytes at the beginning of an AES round contained in the source registers and calculates their contribution to two State bytes at the end

of the MixColumns transformation. This result is added to the fixed registers R0 and R1. Depending on the value of x , different constants are used for MixColumns multiplication.

AESSBOX src1, src2 Substitutes the first register with the AES S-box and exclusive-ors the result to the second register.

AESDEC(x) src1, src2 Similar to $\text{AESENC}(x)$, only using the inverse S-box, different constants for InvMixColumns multiplication, and a slight variation of the location where the feedback from R0 and R1 is added.

AESINVSBOX src1, src2 Just like AESSBOX but using the inverse S-box.

The granularity of the instructions has been chosen so that performance is maximized, implementation flexibility is retained, and integration into existing microcontrollers is facilitated. Three different aspects of the extensions lead to an improvement of performance for AES implementations. Firstly, the instructions perform transformations which are not very well supported by the microcontroller's native instruction set, especially MixColumns and InvMixColumns. Secondly, the instructions for the AES rounds (i.e. $\text{AESENC}(x)$ and $\text{AESDEC}(x)$) perform calculations for two resulting State bytes simultaneously, thus "widening" the 8-bit datapath. Finally, the instructions encompass all transformations of a complete AES round and therefore minimize the number of invocations.

A single invocation of an $\text{AESENC}(x)$ instruction can be seen as processing a quadrant of the 4×4 finite field constant matrix of MixColumns. As there are only two distinct quadrants in this matrix, two versions of this instruction (namely $\text{AESENC}(1)$ and $\text{AESENC}(2)$) are sufficient. ShiftRows is realized by selection of the instructions' source registers. The other transformations of AES (SubBytes and AddRoundKey) work uniformly for all State bytes. Consequently, four invocations of $\text{AESENC}(x)$ are sufficient to calculate a new State column at the end of a round. Note however that it is necessary to execute additional instructions in order to move the State bytes to the required registers. These "overhead" instructions then require a considerable portion of the total execution time, as the AES transformations themselves are made very fast by the instruction set extensions. Any further optimization of the AES extensions would therefore deliver only small improvements in overall performance.

Figure 4 depicts the principal architecture of a functional unit to implement all of the six custom instructions for AES. The basic building blocks are hardware implementations of the AES S-box (both forward and inverse direction) and finite field multipliers for MixColumns and InvMixColumns. The conditional XOR operations are required for AddRoundKey and for adding together contributions to the State bytes. The control signals running along the dashed lines select the direction of the S-box lookup, the constants used for MixColumns multiplication, and conditional XOR operations after the S-box lookup and after MixColumns. They are derived from the type of executed instruction. At the right side of the figure, a fixed XOR and an additional output are shown. These components are used to realize the AESSBOX and AESINVSBOX instructions.

The functional unit includes a pipeline stage after the S-boxes. This allow use of the functional unit with a standard register file as included in most AVR microcontroller implementations, which can read two arbitrary 8-bit registers and can write a 16-bit result per clock cycle, see [6]. The instructions $\text{AESENC}(x)$ and $\text{AESDEC}(x)$ can then be executed in two clock cycles: In the first clock cycle the registers R_d and R_r are read and transformed with the help of the S-box. The substituted bytes are stored in the pipeline stage at the end of the first clock cycle. In the second

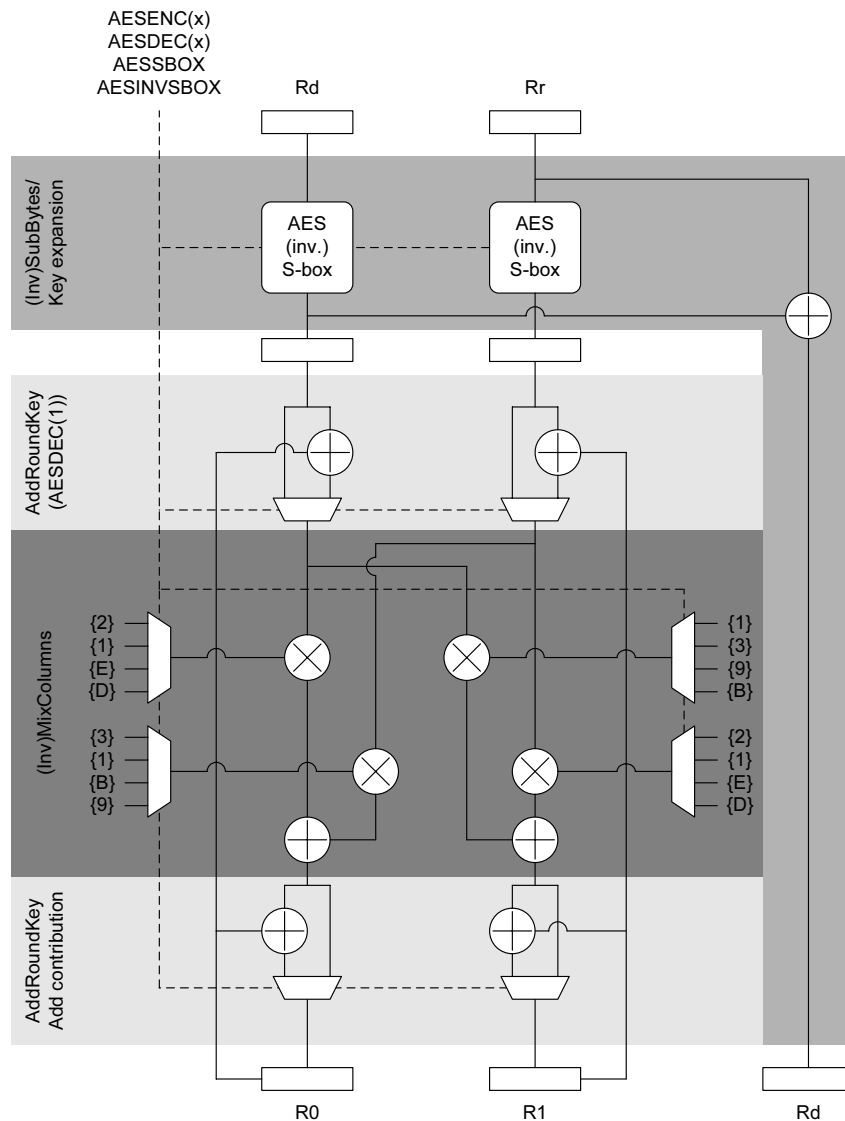


Figure 4: Possible implementation of the AES extensions for AVR

clock cycle the registers R0 and R1 are loaded as required by the feedback paths illustrated in Figure 4. They are combined with the intermediate results from the pipeline registers and written back to the register pair R0 and R1 at the end of the second clock cycle.

In order to illustrate the application of the AES extensions, two small code examples are provided. Figure 5 shows the assembly code for performing all four round transformations on a single State column. Figure 6 contains the update of the first round key word.

The hardware cost of the functional unit depicted in Figure 4 is about 1.1 kGates. Its speed is well above the maximal speed of state-of-the-art AVR microcontrollers—which is currently at about 20 MHz—even when implemented in relatively old technology (0.35 μm CMOS standard cell library, cf. [125]). Encryption or decryption of a single block with AES-128 takes about 1,259 clock cycles (with an on-the-fly key expansion). When compared to an optimized AES software implementation in assembly, this constitutes a speedup of 3.2 for encryption and 4.8 for decryption. Code size can be traded of for speed, where a fast implementation of AES encryption requires

```

; State column in R6, R11, R16, R5
; Round key word in R22-R25
; New State column is written over old column

; Calculate upper half of new column
MOVW R0, R22          ; Move two round key bytes into R0-R1
AESENC(1) R6, R11     ; ShiftRows, SubBytes, MixColumns & AddRoundKey
AESENC(2) R16, R5     ; ShiftRows, SubBytes, MixColumns
MOVW R30, R0          ; Save half column in temporary registers R30-R31

; Calculate lower half of new column
MOVW R0, R24          ; Move the other two round key bytes into R0-R1
AESENC(1) R16, R5     ; ShiftRows, SubBytes, MixColumns & AddRoundKey
AESENC(2) R6, R11     ; ShiftRows, SubBytes, MixColumns

; Store new column over old column
MOV R6, R30
MOV R11, R31
MOV R16, R0
MOV R5, R1

```

Figure 5: Round transformations for a single State column

```

; First word of old round key in R18-R21
; Last word of old round key in R26-R29
; Rcon located in R30
; New first round key word written over old word

EOR R18, R30          ; Add Rcon
AESSBOX R18, R27      ; RotWord, SubWord, Add to old byte
AESSBOX R19, R28      ; RotWord, SubWord, Add to old byte
AESSBOX R20, R29      ; RotWord, SubWord, Add to old byte
AESSBOX R21, R26      ; RotWord, SubWord, Add to old byte

```

Figure 6: Update of the first round key word

1,090 bytes when using the extensions. At the same time, such an implementation requires only 4 bytes of RAM.

5 Protection against Side-Channel Attacks

Since the first publications on side-channel attacks (SCAs) by Kocher et al. about a decade ago [79], the topic of implementation security has attracted an ever increasing interest of the cryptographic research community. The principal problem is that a concrete implementation of a cryptographic algorithm normally leaks information about intermediate results via various physical values. Prominent examples of such physical values are execution time [79], power consumption [80], and electromagnetic emanations [105, 40]. Capture and subsequent analysis of these values allows to draw conclusions about the actual intermediate results which occurred during the cryptographic computations. One interesting variant of timing analysis in the context of software implementations is the cache-based timing attack [130, 100, 9].

A great bulk of research work is available, which discusses various forms of SCA attacks and/or appropriate countermeasures for both software and hardware implementations. In the context of software implementation on general-purpose processors, the most important approaches are algorithmic masking (e.g. [46]) and randomized execution (e.g. [87]). Moreover, the efficient application of secure logic styles (e.g. [127, 128, 101]) is still an open research problem. A more complete overview of SCAs has been given in SMEPP deliverable D4.1 “Threat Models for EP2P Networks”.

The advent of cryptography instruction set extensions has added a new facet for research into SCA countermeasures. Only few works have investigated the challenges and opportunities for SCA countermeasures for processors with cryptography extensions. It has been shown that ISEs can help to protect software implementations against implementation attacks [126]. The use of dedicated instructions for the S-box lookup can both prevent cache-based timing attacks on the AES rounds and power analysis of the AES key expansion [84]. However, protection against advanced power analysis methods demands a dramatic overhead in terms of execution time. In order to increase the protection against power analysis by a factor of about 10^4 , the performance degrades by a factor of about 100.

This has shown that software countermeasures alone will not be sufficient to protect devices against SCA. Another work has considered to equip the implementation of the enhanced processor with hardware protection measures [122]. Three approaches for increasing implementation security have been presented. These approaches only have a relatively moderate impact on AES performance, with an increase in cycle count from about 17 % to 100 %. They mainly rely on the application of so-called secure logic styles, which provide protection at the hardware cell level. Depending on the attainable security and implementation details, the hardware overhead has been estimated to be about 28 kGates. Although this is a considerable increase in silicon area, this solution is relatively easy to implement as it leaves most of the original processor’s hardware untouched.

Glossary

Acronym	Definition
ADSP	Application Domain-Specific Processor
AES	Advanced Encryption Standard
ARM	Advanced RISC Machine
ASIC	Application-Specific Integrated Circuit
ASIP	Application-Specific Instruction Set Processor
AVR	Advanced Virtual RISC
CIOS	Coarsely Integrated Operand Scanning
COP	Coprocessor
CPI	Coprocessor Interface
DLP	Discrete Logarithm Problem
DMRP	Data Memory Read Port
DPW	Datapath Width
DSA	Digital Signature Algorithm
DSRCP	Domain Specific Reconfigurable Cryptographic Processor
ECC	Elliptic Curve Cryptography
ECDLP	Elliptic Curve Discrete Logarithm Problem
FIOS	Finely Integrated Operand Scanning
FIPS	Finely Integrated Product Scanning
GE	Gate Equivalent
GF	Galois Field
GPP	General-Purpose Processor
IFP	Integer Factorization Problem
ISA	Instruction Set Architecture
ISE	Instruction Set Extension
IW	Issue Width
KCM	Karatsuba and Comba multiplication with Montgomery reduction
LUT	Lookup Table
MAC	Multiply Accumulate
MIPS	Microprocessor without Interlocked Pipeline Stages
MM	Memory Mapped
OEF	Optimal Extension Field
PKC	Public-Key Cryptography
RISC	Reduced Instruction Set Computer
RSA	Rivest Shamir Adleman (cryptosystem)
SCA	Side-Channel Attack
SMEPP	Secure Middleware for Embedded Peer-to-Peer Systems
SPARC	Scalable Processor Architecture
SKC	Secret-Key Cryptography
VLIW	Very Long Instruction Word
XOR	Exclusive Or

References

- [1] H. Aigner, H. Bock, M. Hüutter, and J. Wolkerstorfer. A Low-Cost ECC Coprocessor for Smartcards. In M. Joye and J.-J. Quisquater, editors, *Proceedings of the Workshop on Cryptographic Hardware and Embedded Systems – CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 107–118. Springer, 2004.
- [2] K. Aoki and H. Lipmaa. Fast Implementations of AES Candidates. In *Proceedings of the Third AES Candidate Conference, NIST*, pages 106–120, 2000.
- [3] ARM Ltd. SecurCore Family Website. <http://www.arm.com/products/CPUs/families/SecurCoreFamily.html>.
- [4] K. Atasu, L. Breveglieri, and M. Macchetti. Efficient AES Implementations for ARM Based Platforms. In *Proceedings of the 2004 ACM Symposium on Applied Computing (SAC 2004)*, pages 841–845. ACM Press, 2004.
- [5] K. Atasu, L. Pozzi, and P. Ienne. Automatic Application-Specific Instruction-Set Extensions under Microarchitectural Constraints. In *40th Design Automation Conference, DAC 2003, Anaheim, CA, USA, June 2-6, 2003, Proceedings, 2003*.
- [6] Atmel Corporation. 8-bit AVR Microcontroller with 128K Bytes In-System Programmable Flash. Available online at http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf, August 2007.
- [7] L. S. Au and N. Burgess. A (4:2) Adder for Unified GF(p) and GF(2^m) Galois Field Multipliers. In *Conference Record of the Thirty-Sixth Asilomar Conference on Signals, Systems and Computers, 2002.*, volume 2, pages 1619–1623. IEEE, November 2002.
- [8] G. Bai, Z. Huang, H. Yuan, H. Chen, M. Liu, G. Chen, T. Zhou, and Z. Chen. A High Performance VLSI Chip of the Elliptic Curve Cryptosystems. In *Proceedings of the 7th International Conference on Solid-State and Integrated Circuits Technology, 2004.*, volume 3, pages 2059–2062. IEEE, October 2004.
- [9] D. J. Bernstein. Cache-timing attacks on AES. Available online at <http://cr.yp.to/antiforgery/cachetiming-20050414.pdf>, April 2005.
- [10] G. Bertoni, L. Breveglieri, P. Fragneto, M. Macchetti, and S. Marchesin. Efficient Software Implementation of AES on 32-Bit Platforms. In B. S. K. Jr., Çetin Kaya Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 159–171. Springer, 2003.
- [11] G. Bertoni, L. Breveglieri, F. Roberto, and F. Regazzoni. Speeding Up AES By Extending a 32-Bit Processor Instruction Set. In *Proceedings of the 17th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP 2006)*, pages 275–282. IEEE Computer Society, September 2006.
- [12] P. Biswas, S. Banerjee, N. Dutt, L. Pozzi, and P. Ienne. ISEGEN: Generation of High-Quality Instruction Set Extensions by Iterative Improvement. In *2005 Design, Automation and Test in Europe Conference and Exposition (DATE 2005), 7-11 March 2005, Munich, Germany*, pages 1246–1251. IEEE Computer Society, 2005.

- [13] P. Biswas, V. Choudhary, K. Atasu, L. Pozzi, P. Ienne, and N. Dutt. Introduction of Local Memory Elements in Instruction Set Extensions. In *Proceedings of the 41st annual conference on Design automation (DAC 2004)*, pages 729–734. ACM Press, 2004.
- [14] M. K. Brown, D. R. Hankerson, J. C. L. Hernández, and A. J. Menezes. Software Implementation of the NIST Elliptic Curves Over Prime Fields. In D. Naccache, editor, *Topics in Cryptology - CT-RSA 2001, The Cryptographers' Track at the RSA Conference 2001, San Francisco, CA, USA, April 8-12, 2001, Proceedings*, volume 2020 of *Lecture Notes in Computer Science*, pages 250–265. Springer, 2001.
- [15] M. Bucci. Dual Mode (Integer, Polynomial) Fast Modular Multipliers, May 1997. Rump session talk given at EUROCRYPT'97, May 13, 1997, Konstanz, Germany.
- [16] R. Buchty. *Cryptonite — A Programmable Crypto Processor Architecture for High-Bandwidth Applications*. Ph.d. thesis, Technische Universität München, LRR, September 2002. Available online at <http://tumb1.biblio.tu-muenchen.de/publ/diss/in/2002/buchty.pdf>.
- [17] J. Burke, J. McDonald, and T. Austin. Architectural Support for Fast Symmetric-Key Cryptography. In *ASPLOS-IX Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems, Cambridge, MA, USA, November 12-15, 2000*, pages 178–189, New York, NY, USA, 2000. ACM Press.
- [18] D. Canright. A Very Compact S-Box for AES. In J. R. Rao and B. Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 441–455. Springer, 2005.
- [19] Çetin Kaya Koç. Analysis of MMX Technology for Implementing Number-Theoretic Cryptosystems. Technical report, Intel Corporation, February 1997.
- [20] Çetin Kaya Koç and T. Acar. Montgomery Multiplication in $GF(2^k)$. *Designs, Codes and Cryptography*, 14(1):57–69, April 1998.
- [21] Çetin Kaya Koç, T. Acar, and B. S. K. Jr. Analyzing and Comparing Montgomery Multiplication Algorithms. *IEEE Micro*, 16(3):26–33, June 1996.
- [22] N. Clark, H. Zhong, and S. Mahlke. Processor Acceleration Through Automated Instruction Set Customization. In *Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture*, pages 129–140. IEEE Computer Society, 2003.
- [23] N. Clark, H. Zhong, W. Tang, and S. Mahlke. Automatic Design of Application Specific Instruction Set Extensions through Dataflow Graph Exploration. *International Journal of Parallel Programming*, 31(6):429–449, December 2003.
- [24] L. Dadda, M. Macchetti, and J. Owen. The Design of a High Speed ASIC Unit for the Hash Function SHA-256 (384, 512). In *2004 Design, Automation and Test in Europe Conference and Exposition (DATE 2004), 16-20 February 2004, Paris, France*, volume 3, pages 70–75. IEEE Computer Society, February 2004.
- [25] J.-F. Dhem. *Design of an efficient public-key cryptographic library for RISC-based smart cards*. PhD thesis, Université Catholique de Louvain, Louvain-la-Neuve, Belgium, May 1998.

- [26] W. Drescher, K. Bachmann, and G. Fettweis. VLSI Architecture for Datapath Integration of Arithmetic Over $GF(2^m)$ on Digital Signal Processors. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, 1997 (ICASSP-97)*, volume 1, pages 631–634. IEEE Computer Society, April 1997.
- [27] H. Eberle, N. Gura, S. C. Shantz, V. Gupta, and L. Rarick. A Public-key Cryptographic Processor for RSA and ECC. In *Proceedings of the 15th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP 2004)*, pages 98–110. IEEE Computer Society, September 2004.
- [28] H. Eberle, S. Shantz, V. Gupta, N. Gura, L. Rarick, and L. Spracklen. Accelerating Next-Generation Public-Key Cryptosystems on General-Purpose CPUs. *IEEE Micro*, 25(2):52–59, March-April 2005.
- [29] H. Eberle, A. Wander, N. Gura, S. Chang-Shantz, and V. Gupta. Architectural Extensions for Elliptic Curve Cryptography over $GF(2^m)$ on 8-bit Microprocessors. In *Proceedings of the 16th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP 2005)*, pages 343–349. IEEE Computer Society, July 2005.
- [30] T. ElGamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In *Advances in Cryptology: Proceedings of CRYPTO 84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer, 1984.
- [31] G. Elias, L. S. Cheng, A. Miri, and H. Yeap. An Improved FPGA Implementation of a Hyperelliptic Cryptosystem Coprocessor. In *Canadian Conference on Electrical and Computer Engineering, 2004.*, volume 2, pages 773–776. IEEE, May 2004.
- [32] M. Feldhofer, S. Dominikus, and J. Wolkerstorfer. Strong Authentication for RFID Systems using the AES Algorithm. In M. Joye and J.-J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems – CHES 2004, 6th International Workshop, Cambridge, MA, USA, August 11-13, 2004, Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 357–370. Springer, August 2004.
- [33] A. M. Fiskiran and R. B. Lee. Evaluating instruction set extensions for fast arithmetic on binary finite fields. In J. Cavallaro, T. Noll, S. Rajopadhye, and L. Thiele, editors, *Proceedings of the 15th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP 2004)*, pages 125–136. IEEE Computer Society, 2004.
- [34] A. M. Fiskiran and R. B. Lee. *PAX: A Datapath-Scalable Minimalist Cryptographic Processor For Mobile Environments*, chapter 2. Nova Science, NY, October 2004.
- [35] A. M. Fiskiran and R. B. Lee. Performance Scaling of Cryptography Operations in Servers and Mobile Clients. In *Proceedings of the Workshop on Building Block Engine Architectures for Computer Networks (BEACON 2004)*, October 2004. Available online at http://palms.ee.princeton.edu/PALMSopen/fiskiran04performance_with_citation.pdf.
- [36] A. M. Fiskiran and R. B. Lee. On-Chip Lookup Tables for Fast Symmetric-Key Encryption. In *Proceedings of the 16th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP 2005)*, pages 356–363. IEEE, IEEE Press, July 2005.

- [37] M. A. Fiskiran and R. B. Lee. Performance Impact of Addressing Modes on Encryption Algorithms. In *Proceedings of the International Conference on Computer Design (ICCD 2001)*, pages 542–545. IEEE, September 2001.
- [38] C. K. Fong, D. R. Hankerson, J. C. L. Hernández, A. J. Menezes, and M. B. Tucker. Performance Comparisons of Elliptic Curve Systems in Software. Available online at <http://www.cacr.math.uwaterloo.ca/conferences/2001/ecc/hankerson.pdf>, 2001. Presentation at the 5th Workshop on Elliptic Curve Cryptography (ECC 2001).
- [39] P. A. L. for Multimedia and S. (PALMS). PAX Project Website. <http://palms.ee.princeton.edu/PAX/>.
- [40] K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic Analysis: Concrete Results. In Çetin Kaya Koç, D. Naccache, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer, 2001.
- [41] B. Gladman. Implementations of AES (Rijndael) in C/C++ and Assembler. Available online at http://fp.gladman.plus.com/cryptography_technology/rijndael/index.htm.
- [42] B. Gladman. AES Algorithm Efficiency. Available online at http://fp.gladman.plus.com/cryptography_technology/aesr1/, 1999.
- [43] J. Goodman and A. Chandrakasan. An Energy Efficient Reconfigurable Public-Key Cryptography Processor Architecture. In Çetin Kaya Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings*, volume 1965 of *Lecture Notes in Computer Science*, pages 175–190. Springer, 2000.
- [44] J. Goodman and A. Chandrakasan. An Energy Efficient Reconfigurable Public-Key Cryptography Processor Architecture. *IEEE Journal of Solid-State Circuits*, 36(11):1808–1820, November 2001.
- [45] J. R. Goodman. *Energy Scalable Reconfigurable Cryptographic Hardware for Portable Applications*. Ph.d. thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, August 2000.
- [46] L. Goubin and J. Patarin. DES and Differential Power Analysis – The Duplication Method. In Çetin Kaya Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES’99, First International Workshop, Worcester, MA, USA, August 12-13, 1999, Proceedings*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer, 1999.
- [47] J. Großschädl. A unified radix-4 partial product generator for integers and binary polynomials. In *Proceedings of the 35th IEEE International Symposium on Circuits and Systems (ISCAS 2002)*, volume III, pages 567–570. IEEE, 2002.
- [48] J. Großschädl. Instruction Set Extension for Long Integer Modulo Arithmetic on RISC-Based Smart Cards. In *Proceedings of the 14th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD 2002)*, pages 13–19. IEEE Computer Society, 2002.

- [49] J. Großschädl, R. M. Avanzi, E. Savaş, and S. Tillich. Energy-Efficient Software Implementation of Long Integer Modular Arithmetic. In J. R. Rao and B. Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 75–90. Springer, 2005.
- [50] J. Großschädl and G.-A. Kamendje. A Single-Cycle $(32 \times 32 + 32 + 64)$ -bit Multiply/Accumulate Unit for Digital Signal Processing and Public-Key Cryptography. In *Proceedings of the 2003 10th IEEE International Conference on Electronics, Circuits and Systems (ICECS 2003)*, volume 2, pages 739–741. IEEE, December 2003.
- [51] J. Großschädl and G.-A. Kamendje. Architectural Enhancements for Montgomery Multiplication on Embedded RISC Processors. In J. Zhou, M. Yung, and Y. Han, editors, *Applied Cryptography and Network Security, First International Conference, ACNS 2003, Kunming, China, October 16-19, 2003, Proceedings*, volume 2846 of *Lecture Notes in Computer Science*, pages 418–434. Springer, October 2003.
- [52] J. Großschädl and G.-A. Kamendje. Instruction Set Extension for Fast Elliptic Curve Cryptography over Binary Finite Fields $GF(2^m)$. In E. Deprettere, S. Bhattacharyya, J. Cavallaro, A. Darte, and L. Thiele, editors, *Proceedings of the 14th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP 2003)*, pages 455–468. IEEE Computer Society, June 2003.
- [53] J. Großschädl and G.-A. Kamendje. Low-Power Design of a Functional Unit for Arithmetic in Finite Fields $GF(p)$ and $GF(2^m)$. In K. Chae and M. Yung, editors, *Information Security Applications (WISA 2003)*, volume 2908 of *Lecture Notes in Computer Science*, pages 227–243. Springer, 2003.
- [54] J. Großschädl and G.-A. Kamendje. Optimized RISC Architecture for Multiple-Precision Modular Arithmetic. In D. Hutte, G. Müller, W. Stephan, and M. Ullmann, editors, *Security in Pervasive Computing - SPC 2003*, volume 2802 of *Lecture Notes in Computer Science*, pages 253–270. Springer, 2003.
- [55] J. Großschädl, S. S. Kumar, and C. Paar. Architectural Support for Arithmetic in Optimal Extension Fields. In J. Cavallaro, T. Noll, S. Rajopadhye, and L. Thiele, editors, *Proceedings of the 15th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP 2004)*, pages 111–124. IEEE Computer Society, 2004.
- [56] J. Großschädl, K. C. Posch, and S. Tillich. Architectural Enhancements to Support Digital Signal Processing and Public-Key Cryptography. In B. Rinner and W. Elmenreich, editors, *Proceedings of the 2nd Workshop on Intelligent Solutions in Embedded Systems (WISES 2004)*, pages 129–143, June 2004.
- [57] J. Großschädl and E. Savaş. Instruction Set Extensions for Fast Arithmetic in Finite Fields $GF(p)$ and $GF(2^m)$. In M. Joye and J.-J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems – CHES 2004, 6th International Workshop, Cambridge, MA, USA, August 11-13, 2004, Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 133–147. Springer, August 2004.
- [58] J. Großschädl, A. Szekely, and S. Tillich. Algorithm Exploration for Long Integer Modular Arithmetic on a SPARC V8 Processor with Cryptography Extensions. In *Proceedings of*

- the 2nd International Conference on Embedded Software and Systems (ICESS 2005)*, pages 187–194. IEEE Computer Society, 2005.
- [59] J. Guajardo, R. Blümel, U. Krieger, and C. Paar. Efficient Implementation of Elliptic Curve Cryptosystems on the TI MSP430x33x Family of Microcontrollers. In K. Kim, editor, *Public Key Cryptography: 4th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2001, Cheju Island, Korea, February 13-15, 2001. Proceedings*, volume 1992 of *Lecture Notes in Computer Science*, pages 365–382. Springer, 2001.
- [60] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz. Comparing Elliptic Curve Cryptography and RSA on 8-Bit CPUs. In M. Joye and J.-J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems – CHES 2004, 6th International Workshop, Cambridge, MA, USA, August 11-13, 2004, Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 119–132. Springer, 2004.
- [61] F. K. Gürkaynak, A. Burg, N. Felber, W. Fichtner, D. Gasser, F. Hug, and H. Kaeslin. A 2GB/s balanced AES crypto-chip implementation. In D. Garrett, J. Lach, and C. A. Zukowski, editors, *14th ACM Great Lakes Symposium on VLSI 2004, Boston, MA, USA, April 26-28, 2004, Proceedings*, pages 39–44. ACM Press, 2004.
- [62] C.-S. Ha, J. H. Lee, D. S. Leem, M.-S. Park, and B.-Y. Choi. ASIC Design of IPsec Hardware Accelerator for Network Security. In *Proceedings of 2004 IEEE Asia-Pacific Conference on Advanced System Integrated Circuits 2004.*, pages 168–171. IEEE, August 2004.
- [63] G. Hachez, F. Koeune, and J.-J. Quisquater. cAESar results: Implementation of Four AES Candidates on Two Smart Cards. In *Second Advanced Encryption Standard Candidate Conference*, pages 95–108, Hotel Quirinale, Rome, Italy, March 1999.
- [64] Y. Han, P.-C. Leong, P.-C. Tan, and J. Zhang. Fast Algorithms for Elliptic Curve Cryptosystems over Binary Finite Field. In K. Y. Lam, E. Okamoto, and C. Xing, editors, *Advances in Cryptology (ASIACRYPT 1999)*, volume 1716 of *Lecture Notes in Computer Science*, pages 75–85. Springer, 1999.
- [65] D. R. Hankerson, J. C. López Hernandez, and A. J. Menezes. Software Implementation of Elliptic Curve Cryptography Over Binary Fields. In Çetin Kaya Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings*, volume 1965 of *Lecture Notes in Computer Science*, pages 1–24. Springer, 2000.
- [66] T. Hasegawa, J. Nakajima, and M. Matsui. A Practical Implementation of Elliptic Curve Cryptosystems over GF(p) on a 16-Bit Microcomputer. In H. Imai and Y. Zheng, editors, *Public Key Cryptography (PCK 1998)*, volume 1431 of *Lecture Notes in Computer Science*, pages 182–194. Springer, 1998.
- [67] Y. Hitchcock, E. Dawson, A. Clark, and P. Montague. Implementing an efficient elliptic curve cryptosystem over GF(p) on a smart card. In K. Burrage and R. B. Sidje, editors, *Proceedings of the 10th International Conference on Computational Techniques and Applications (CTAC 2001)*, pages 354–381. Australian Mathematical Society, 2001. Available for download at <http://anziamj.austms.org.au/V44/CTAC2001/Hitc/Hitc.pdf>.

- [68] A. Hodjat, D. Hwang, B.-C. Lai, K. Tiri, and I. Verbauwhede. A 3.84 Gbits/s AES crypto coprocessor with modes of operation in a 0.18-um CMOS Technology. In J. Lach, G. Qu, and Y. I. Ismail, editors, *15th ACM Great Lakes Symposium on VLSI 2005, Chicago, Illinois, USA, April 17-19, 2005, Proceedings*, pages 60–63. ACM, ACM Press, April 2005. Available online at http://portal.acm.org/ft_gateway.cfm?id=1057677&type=pdf&coll=GUIDE&dl=GUIDE&CFID=50585284&CFTOKEN=38947284.
- [69] A. Hodjat and I. Verbauwhede. High-Throughput Programmable Cryptocoprocessor. *IEEE Micro*, 24(3):34–45, May/June 2004.
- [70] A. Hodjat and I. Verbauwhede. Interfacing a High Speed Crypto Accelerator to an Embedded CPU. In *Conference Record of the Thirty-Eighth Asilomar Conference on Signals, Systems, and Computers, 2004*, volume 1, pages 488–492. IEEE, November 2004.
- [71] IEEE. IEEE 802.3 Higher Speed Study Group (HSSG) Website. <http://grouper.ieee.org/groups/802/3/hssg/index.html>.
- [72] IEEE. IEEE Standard 802.3-2005: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications. Available online at <http://standards.ieee.org/getieee802/>, December 2005.
- [73] Intel Corporation. Itanium 2 Processor Website. <http://www.intel.com/products/processor/itanium2/index.htm>.
- [74] Intel Corporation. Intel Itanium Processor: High Performance On Security Algorithms (RSA Decryption Kernel). Test Report SEG_103, Intel Corporation, 2000. Available online at <http://www.intel.com/cd/ids/developer/asmo-na/eng/dc/itanium/optimization/80661.htm>.
- [75] J. Irwin and D. Page. Using Media Processors for Low-Memory AES Implementation. In E. Deprettere, S. Bhattacharyya, J. Cavallaro, A. Darte, and L. Thiele, editors, *Proceedings of the 14th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP 2003)*, pages 144–154. IEEE Computer Society, 2003.
- [76] U. Kastens, D. K. Le, A. Slowik, and M. Thies. Feedback Driven Instruction-Set Extension. In *Proceedings of the 2004 ACM SIGPLAN/SIGBED conference on Language, Compiler and Tool Support for Embedded Systems (LCTES 2004)*, pages 126–135. ACM Press, 2004.
- [77] H. W. Kim and S. Lee. Design and Implementation of a Private and Public Key Crypto Processor and Its Application to a Security System. *IEEE Transactions on Consumer Electronics*, 50(1):214–224, February 2004.
- [78] N. Koblitz. Elliptic Curve Cryptosystems. *Mathematics of Computation*, 48:203–209, 1987.
- [79] P. C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In N. Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, number 1109 in Lecture Notes in Computer Science, pages 104–113. Springer, 1996.

- [80] P. C. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In M. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.
- [81] R. B. Lee, Z. Shi, and X. Yang. Efficient Permutation Instructions for Fast Software Cryptography. *IEEE Micro*, 21(6):56–69, December 2001.
- [82] H. Lipmaa. AES/Rijndael: speed. Available online at <http://www.adastral.ucl.ac.uk/~helger/research/aes/rijndael.html>.
- [83] J. López and R. Dahab. Fast Multiplication on Elliptic Curves over $GF(2^m)$ without Pre-computation. In Çetin Kaya Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES'99, First International Workshop, Worcester, MA, USA, August 12-13, 1999, Proceedings*, volume 1717 of *Lecture Notes in Computer Science*, pages 316–327. Springer, 1999.
- [84] S. Mangard. A Simple Power-Analysis (SPA) Attack on Implementations of the AES Key Expansion. In P. J. Lee and C. H. Lim, editors, *Information Security and Cryptology - ICISC 2002, 5th International Conference Seoul, Korea, November 28-29, 2002, Revised Papers*, volume 2587 of *Lecture Notes in Computer Science*, pages 343–358. Springer, 2003.
- [85] M. Matsui. How Far Can We Go on the x64 Processors? In M. J. B. Robshaw, editor, *Fast Software Encryption, 13th International Workshop, FSE 2006, Graz, Austria, March 15-17, 2006, Revised Selected Papers*, volume 4047 of *Lecture Notes in Computer Science*, pages 341–358. Springer, March 2006.
- [86] M. Matsui and S. Fukuda. How to Maximize Software Performance of Symmetric Primitives on Pentium III and 4 Processors. In B. K. Roy and W. Meier, editors, *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers*, volume 3017 of *Lecture Notes in Computer Science*, pages 398–412. Springer, February 2004.
- [87] D. May, H. L. Muller, and N. P. Smart. Non-deterministic Processors. In V. Varadharajan and Y. Mu, editors, *Information Security and Privacy, 6th Australasian Conference, ACISP 2001, Sydney, Australia, July 11-13, 2001, Proceedings*, volume 2119 of *Lecture Notes in Computer Science*, pages 115–129. Springer, 2001.
- [88] J. P. McGregor and R. B. Lee. Architectural Enhancements for Fast Subword Permutations with Repetitions in Cryptographic Applications. In *Proceedings of the International Conference on Computer Design (ICCD 2001)*, pages 453–461. IEEE, September 2001.
- [89] V. S. Miller. Use of Elliptic Curves in Cryptography. In H. C. Williams, editor, *Advances in Cryptology - CRYPTO '85, Santa Barbara, California, USA, August 18-22, 1985, Proceedings*, volume 218 of *Lecture Notes in Computer Science*, pages 417–426. Springer, 1986.
- [90] MIPS Technologies, Inc. MIPS32 Architecture for Programmers. Available online at <http://www.mips.com/content/Documentation/MIPSDocumentation/ProcessorArchitecture/doclibrary>.

- [91] MIPS Technologies, Inc. SmartMIPS AES Product Website. http://www.mips.com/products/architectures/SmartMIPS_ASE.php.
- [92] P. L. Montgomery. Modular Multiplication without Trial Division. *Mathematics of Computation*, 44:519–521, 1985.
- [93] D. Mukhopadhyay and D. RoyChowdhury. An Efficient End To End Design of Rijndael Cryptosystem in 0.18 μ CMOS. In *18th International Conference on VLSI Design, 2005.*, pages 405–410. IEEE, January 2005.
- [94] K. Nadehara, M. Ikekawa, and I. Kuroda. Extended Instructions for the AES Cryptography and their Efficient Implementation. In *IEEE Workshop on Signal Processing Systems (SIPS'04)*, pages 152–157, Austin, Texas, USA, October 2004. IEEE Press.
- [95] E. Nahum, S. O'Malley, H. Orman, and R. Schroepel. Towards High Performance Cryptographic Software. In *Third IEEE Workshop on the Architecture and Implementation of High Performance Communication Subsystems, 1995 (HPCS '95)*, pages 69–72. IEEE, August 1995.
- [96] National Institute of Standards and Technology (NIST). FIPS-186-2: Digital Signature Standard (DSS), January 2000. Available online at <http://www.itl.nist.gov/fipspubs/>.
- [97] National Institute of Standards and Technology (NIST). FIPS-197: Advanced Encryption Standard, November 2001. Available online at <http://www.itl.nist.gov/fipspubs/>.
- [98] K. Okeya and K. Sakurai. Efficient Elliptic Curve Cryptosystems from a Scalar Multiplication Algorithm with Recovery of the y-Coordinate on a Montgomery-Form Elliptic Curve. In Çetin Kaya Koç, D. Naccache, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, volume 2162 of *Lecture Notes in Computer Science*, pages 126–141. Springer, 2001.
- [99] D. Oliva, R. Buchty, and N. Heintze. AES and the Cryptonite Crypto Processor. In *CASES '03: Proceedings of the 2003 international conference on Compilers, architecture and synthesis for embedded systems*, pages 198–209, New York, NY, USA, 2003. ACM Press.
- [100] D. Page. Theoretical Use of Cache Memory as a Cryptanalytic Side-Channel. Technical Report CSTR-02-003, University of Bristol, Department of Computer Science, June 2002. Available online at <http://www.cs.bris.ac.uk/Publications/Papers/1000625.pdf>.
- [101] T. Popp and S. Mangard. Masked Dual-Rail Pre-Charge Logic: DPA-Resistance without Routing Constraints. In J. R. Rao and B. Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 172–186. Springer, 2005.
- [102] L. Pozzi, M. Vuletić, and P. Ienne. Automatic Topology-Based Identification of Instruction-Set Extensions for Embedded Processors. Technical Report CS 01/377, Swiss Federal Institute of Technology Lausanne, Processor Architecture Laboratory, IN-F Ecublens, 1015 Lausanne, Switzerland, December 2001. Available online at http://lap.epfl.ch/publications/PozziDec01_AutomaticTopologyBasedIdentification_TR.pdf.

- [103] L. Pozzi, M. Vuletić, and P. Ienne. Automatic Topology-Based Identification of Instruction-Set Extensions for Embedded Processors. In *Proceedings of the conference on Design, automation and test in Europe (DATE 2002)*, page 1138. IEEE Computer Society, 2002.
- [104] N. Pramstaller and J. Wolkerstorfer. A Universal and Efficient AES Co-Processor for Field Programmable Logic Arrays. In J. Becker, M. Platzner, and S. Vernalde, editors, *Field Programmable Logic and Application, 14th International Conference, FPL 2004, Leuven, Belgium, August 30-September 1, 2004, Proceedings*, volume 3203 of *Lecture Notes in Computer Science*, pages 565–574. Springer, August 2004. ISBN 978-3-540-22989-6.
- [105] J.-J. Quisquater and D. Samyde. ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards. In I. Attali and T. P. Jensen, editors, *Smart Card Programming and Security, International Conference on Research in Smart Cards, E-smart 2001, Cannes, France, September 19-21, 2001, Proceedings*, volume 2140 of *Lecture Notes in Computer Science*, pages 200–210. Springer, 2001.
- [106] S. Ravi, A. Raghunathan, N. Potlapally, and M. Sankaradass. System design methodologies for a wireless security processing platform. In *DAC '02: Proceedings of the 39th Conference on Design Automation*, pages 777–782, New York, NY, USA, 2002. ACM Press.
- [107] R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978. ISSN 0001-0782.
- [108] E. Savaş, A. F. Tenca, and Çetin Kaya Koç. A Scalable and Unified Multiplier Architecture for Finite Fields $GF(p)$ and $GF(2^m)$. In Çetin Kaya Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings*, volume 1965 of *Lecture Notes in Computer Science*, pages 277–292. Springer, August 2000.
- [109] H. Scharwaechter, D. Kammler, A. Wieferink, M. Hohenauer, K. Karuri, J. Ceng, R. Leupers, G. Ascheid, and H. Meyr. ASIP Architecture Exploration for Efficient Isec Encryption: A Case Study. In H. Schepers, editor, *Software and Compilers for Embedded Systems, 8th International Workshop, SCOPES 2004, Amsterdam, The Netherlands, September 2-3, 2004, Proceedings*, volume 3199 of *Lecture Notes in Computer Science*, pages 33–46. Springer, September 2004.
- [110] H. Scharwaechter, D. Kammler, A. Wieferink, M. Hohenauer, K. Karuri, J. Ceng, R. Leupers, G. Ascheid, and H. Meyr. ASIP Architecture Exploration for Efficient IPsec Encryption: A Case Study. *ACM Transactions on Embedded Computing Systems (TECS)*, 6(2):Article 12, May 2007.
- [111] P. Schaumont, K. Sakiyama, A. Hodjat, and I. Verbauwhede. Embedded Software Integration for Coarse-Grain Reconfigurable Systems. In *18th International Parallel and Distributed Processing Symposium (IPDPS 2004), CD-ROM / Abstracts Proceedings, 26-30 April 2004, Santa Fe, New Mexico, USA*, pages 137–142. IEEE Computer Society, April 2004.
- [112] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson. Performance Comparison of the AES Submissions. In *Proceedings of the Second AES Candidate Conference, NIST*, pages 15–34, March 1999.

- [113] R. Sever, A. N. İsmailoğlu, Y. C. Tedmen, and M. Aşkar. A High Speed ASIC Implementation of the Rijndael Algorithm. In *International Symposium on Circuits and Systems (ISCAS 2004), Vancouver, British Columbia, Canada, May 23-26, 2004, Proceedings*, volume 2, pages 541–544. IEEE, May 2004.
- [114] Z. Shi and R. B. Lee. Bit Permutation Instructions for Accelerating Software Cryptography. In *Proceedings of the 11th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP 2000)*, pages 138–148. IEEE, 2000.
- [115] A. Sinha and A. Chandrakasan. JouleTrack – A Web Based Tool for Software Energy Profiling. In *38th Design Automation Conference, DAC 2001, Las Vegas, NV, USA, June 18-22, 2001, Proceedings*, pages 220–225. ACM Press, June 2001.
- [116] STMicroelectronics. Smartcard ICs, 32-bit Platform ICs Website. http://www.st.com/stonline/products/families/smartcard/sc_sol_ics_st22.htm.
- [117] Sun Microsystems, Inc. UltraSPARC T2 Processor Website. <http://www.sun.com/processors/UltraSPARC-T2/>.
- [118] S. Tillich, M. Feldhofer, and J. Großschädl. Area, Delay, and Power Characteristics of Standard-Cell Implementations of the AES S-Box. In S. Vassiliadis, S. Wong, and T. Härmäläinen, editors, *6th International Workshop on Embedded Computer Systems: Architectures, Modeling, and Simulation, SAMOS 2006, Samos, Greece, July 17-20, 2006, Proceedings*, volume 4017 of *Lecture Notes in Computer Science*, pages 457–466. Springer, July 2006.
- [119] S. Tillich and J. Großschädl. A Simple Architectural Enhancement for Fast and Flexible Elliptic Curve Cryptography over Binary Finite Fields $GF(2^m)$. In P.-C. Yew and J. Xue, editors, *Advances in Computer Systems Architecture, 9th Asia-Pacific Conference, ACSAC 2004, Beijing, China, September 2004, Proceedings*, volume 3189 of *Lecture Notes in Computer Science*, pages 282–295. Springer, September 2004.
- [120] S. Tillich and J. Großschädl. Accelerating AES Using Instruction Set Extensions for Elliptic Curve Cryptography. In M. Gavrilova, Y. Mun, D. Taniar, O. Gervasi, K. Tan, and V. Kumar, editors, *Computational Science and Its Applications - ICCSA 2005*, volume 3481 of *Lecture Notes in Computer Science*, pages 665–675. Springer, May 2005.
- [121] S. Tillich and J. Großschädl. Instruction Set Extensions for Efficient AES Implementation on 32-bit Processors. In L. Goubin and M. Matsui, editors, *Cryptographic Hardware and Embedded Systems – CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, volume 4249 of *Lecture Notes in Computer Science*, pages 270–284. Springer, 2006.
- [122] S. Tillich and J. Großschädl. Power-Analysis Resistant AES Implementation with Instruction Set Extensions. In P. Paillier and I. Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems – CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 303–319. Springer, September 2007.
- [123] S. Tillich and J. Großschädl. VLSI Implementation of a Functional Unit to Accelerate ECC and AES on 32-bit Processors. In C. Carlet and B. Sunar, editors, *Arithmetic of Finite Fields, First International Workshop, WAIFI 2007, Madrid, Spain, June 2007, Proceedings*, volume 4547 of *Lecture Notes in Computer Science*, pages 40–54. Springer, June 2007.

- [124] S. Tillich, J. Großschädl, and A. Szekeley. An Instruction Set Extension for Fast and Memory-Efficient AES Implementation. In J. Dittmann, S. Katzenbeisser, and A. Uhl, editors, *Communications and Multimedia Security — 9th IFIP TC-6 TC-11 International Conference, CMS 2005, Salzburg, Austria, September 2005, Proceedings*, volume 3677 of *Lecture Notes in Computer Science*, pages 11–21. Springer, September 2005.
- [125] S. Tillich and C. Herbst. Boosting AES Performance on a Tiny Processor Core. In *Topics in Cryptology - CT-RSA 2008, The Cryptographers' Track at the RSA Conference 2008, San Francisco, CA, USA, April 8-11, 2008, Proceedings*, 2008. To be published.
- [126] S. Tillich, C. Herbst, and S. Mangard. Protecting AES Software Implementations on 32-bit Processors against Power Analysis. In J. Katz and M. Yung, editors, *Proceedings of the 5th International Conference on Applied Cryptography and Network Security (ACNS 2007)*, volume 4521 of *Lecture Notes in Computer Science*, pages 141–157. Springer, June 2007.
- [127] K. Tiri, M. Akmal, and I. Verbauwhede. A Dynamic and Differential CMOS Logic with Signal Independent Power Consumption to Withstand Differential Power Analysis on Smart Cards. In *28th European Solid-State Circuits Conference - ESSCIRC 2002, Florence, Italy, September 24-26, 2002, Proceedings*, pages 403–406. IEEE, September 2002.
- [128] K. Tiri and I. Verbauwhede. A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation. In *2004 Design, Automation and Test in Europe Conference and Exposition (DATE 2004), 16-20 February 2004, Paris, France*, volume 1, pages 246–251. IEEE Computer Society, 2004.
- [129] E. Trichina, M. Bucci, D. D. Seta, and R. Luzzi. Supplemental Cryptographic Hardware for Smart Cards. *IEEE Micro*, 21(6):26–35, November/December 2001.
- [130] Y. Tsunoo, E. Tsujihara, K. Minematsu, and H. Miyauchi. Cryptanalysis of Block Ciphers Implemented on Computers with Cache. In *International Symposium on Information Theory and Its Applications (ISITA 2002)*, October 2002.
- [131] VIA Technologies, Inc. VIA PadLock Security Initiative Website. <http://www.via.com.tw/en/initiatives/padlock/index.jsp>.
- [132] C. Weaver, R. Krishna, L. Wu, and T. Austin. Application Specific Architectures: A Recipe for Fast, Flexible and Power Efficient Designs. In *Proceedings of the 2001 International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES 2001)*, pages 181–185. ACM Press, 2001.
- [133] A. Weimerskirch, D. Stebila, and S. C. Shantz. Generic $GF(2^m)$ Arithmetic in Software and its Application to ECC. In G. Goos, J. Hartmanis, and J. van Leeuwen, editors, *Information Security and Privacy (ACISP 2003)*, volume 2727 of *Lecture Notes in Computer Science*, pages 79–92. Springer, 2003.
- [134] J. Wolkerstorfer. An ASIC Implementation of the AES-MixColumn operation. In P. Rössler and A. Döderlein, editors, *Austrochip 2001*, pages 129–132, 2001. ISBN 3-9501517-0-2.
- [135] A. D. Woodbury. Efficient Algorithms for Elliptic Curve Cryptosystems on Embedded Systems. M.Sc. Thesis, Worcester Polytechnic Institute, September 2001.

- [136] A. D. Woodbury, D. V. Bailey, and C. Paar. Elliptic Curve Cryptography on Smart Cards without Coprocessors. In J. Domingo-Ferrer, D. Chan, and A. Watson, editors, *Proceedings of the Fourth Working Conference on Smart Card Research and Advanced Applications (CARDIS 2000)*, volume 180, pages 71–92. Kluwer Academic Publishers, 2000.
- [137] L. Wu, C. Weaver, and T. Austin. CryptoManiac: A Fast Flexible Architecture for Secure Communication. In *ISCA '01: Proceedings of the 28th annual international symposium on Computer architecture*, pages 110–119, New York, NY, USA, 2001. ACM Press.
- [138] X. Yang and R. B. Lee. Fast Subword Permutation Instructions Using Omega and Flip Network Stages. In *2000 International Conference on Computer Design, 2000. Proceedings*, pages 15–22. IEEE, September 2000.
- [139] X. Zeng, C. Chen, and Q. Zhang. A Reconfigurable Public-Key Cryptography Coprocessor. In *Proceedings of 2004 IEEE Asia-Pacific Conference on Advanced System Integrated Circuits 2004.*, pages 172–175. IEEE, August 2004.