On the Performance of Provably Secure Hashing with Elliptic Curves

Anton Kargl[†], Bernd Meyer[†], and Susanne Wetzel[††]

[†]Siemens AG, Corporate Technology, Otto-Hahn-Ring 6, 81739 München, Germany

[††]Stevens Institute of Technology, Hoboken NJ 07030, USA

Summary
We propose a cryptographic hash function based on the difficulty of computing discrete logarithms in the group of points of an elliptic curve over a finite field. We prove the security of the hash function and analyze the performance. Our implementation of the finite field, the elliptic curve arithmetic, and scalar multiplication is optimized for high throughput on modern 32-bit desktop processors. The achievable data rates allow the use of provably secure hash functions in practical applications.

Key words:
Hash Function, Elliptic Curve Cryptosystem, Koblitz Curve, Optimal Extension Field, Frobenius Representation.

1. Introduction

The security of all commonly used cryptographic hash functions relies on the heuristic difficulty of mathematically modeling and analyzing a wild mix of logic and arithmetic operations in the computation of the hash value. Such combinatorial hash functions have many similarities to block ciphers and reach high data rates. Recent results in the analysis of cryptographic hash functions using techniques from differential cryptanalysis of block ciphers [2, 3, 33, 32, 34] have cast serious doubts on the long-term security of combinatorial hash functions.

On the other hand, there exist proposals for hash function designs based on algebraic methods and public-key techniques [11, 7, 9]. These functions offer provable security in the sense that finding collisions is polynomial-time reducible to the solution of a hard cryptographic problem, e.g., the difficulty of computing discrete logarithms. The main disadvantage of algebraic hash functions is their slow data rate caused by long integer arithmetic and the involved mathematical computations for the public-key part.

The aim of this paper is the analysis of the performance and security of an algebraic hash function based on the difficulty of computing discrete logarithms, where the arithmetic is based on elliptic curve cryptography over optimal extension fields and the prime field arithmetic is optimized for modern 32-bit desktop processors. Using several optimizations for the finite field and the elliptic curve arithmetic algebraic hash functions can be an alternative for practical applications with the advantage of provable security.

The remainder of the paper is organized as follows: in the next section we recapitulate related work and give some definitions used in this article. In the third section we prove the security of our hash function. The efficient implementation of the hash function is explained in the fourth section.

2. Cryptographic Hash Functions

A cryptographic hash function h is an efficiently computable function of messages with the following properties [23]:

- h compresses a message x of arbitrary length to an output $h(x)$ of fixed length,
- h is collision resistant, i.e., it is computationally infeasible to find distinct messages x and x' such that $h(x) = h(x')$.

For certain applications hash functions need special properties, for example preimage [and second preimage] resistance: for a given hash value y [or for a given message x_1] it is computationally infeasible to find a preimage x with $h(x) = y$ [or a second preimage x_2 with $h(x_1) = h(x_2)$]. It is easy to see, that preimage and second preimage resistance is implied by collision-resistance.

Often, a hash function consists of a compression function that iteratively hashes message blocks of a fixed length and the actual internal state of the hash function to the next state. The last computed internal state is the result of the hash function. Techniques for padding the message and encoding their length make sure that a collision of the hash function can be reduced to a collision of the compression function [11].

There exist proposals for hash function designs that use symmetric encryption functions as building blocks for

Manuscript received October 19, 2007
Manuscript revised October 20, 2007

the compression function of a hash function [24, 10]. For example in [24] the author proposes the compression function $H(a \mid b) = Enc_a(b) \oplus b$.[1] Since standard symmetric encryption schemes have relatively short key sizes the compression ratio of the encryption function is low and, therefore, the performance of the compression function H is reduced. The security of this scheme relies on the heuristic assumption that H is a non-invertible random function.

Nonetheless, modern hash functions, SHA-1 [23], RIPEMD-160 [13], or MD5 [23], use the same principle of construction. The compression functions of these hash functions can be interpreted as a symmetric encryption under a very large key followed by a final addition of the plaintext for chaining. Often, the round functions of the encryption schemes consist of simple binary and arithmetic operations in $Z_{2^{32}}$ that are iterated several times with different additive constants. This method of building hash functions yields very efficient algorithms with high data rates [22, 26].

There also exist proposals for hash function designs using public-key techniques for the compression functions [11, 7, 9]. Some of these schemes offer provable security, i. e., finding a collision can be reduced to the solution of a hard problem, but most have the drawback that the computation of the public-key part needs costly long-integer arithmetic.

In [7] Chaum et al. prove the collision-freeness of m-tuple exponentiation and define a provably secure fixed-length hash function h based on the discrete logarithm problem in finite groups. In the following we use the presentation from [30]: let p be a prime such that $q = (p-1)/2$ is also prime. Let $s_1, s_2 \in Z_p$ be primitive elements. Assume that the discrete logarithm $\log_{s_1}(s_2)$ is not known and that it is computationally infeasible to compute the logarithm. Given s_1, s_2 the hash function $h : \{0,\ldots,p-1\}^2 \to Z_p$ is defined as follows:

$$h(x_1, x_2) \mapsto s_1^{x_1} s_2^{x_2} \mod p .$$

It is shown in [30]: given one collision for the function h the discrete logarithm $\log_{s_1}(s_2)$ can be computed in polynomial time contradicting our assumptions. In [7] this result is generalized to the product of m generators $s_1,\ldots,s_m$ and message blocks $x_1,\ldots,x_m$ without analyzing the exact security of the generalized scheme.

[1] $H(a \mid b) = Enc_a(b) \oplus b$ denotes encryption of plaintext b under key a and $a|b$ denotes concatenation of the strings a and b.

When iterating h as a compression function to build hash functions for arbitrary length messages (for example using techniques presented in [11]), one has the problem that the representation of the field elements resulting from an application of h is larger than the values fed back to h. This reduces the length of the message block that can be hashed in a single application of h. Even for the case of elliptic curves, where it is possible to choose the number of points on the curve to be a prime number greater than the size of the field, a compressed representation of a point consists of a field element for the x-coordinate and the sign of the y-coordinate.

Moreover, we would like to use Koblitz curves [15] which offer faster scalar multiplication over extension fields due to the efficient computation of the Frobenius endomorphism. With Koblitz curves the situation is even worse since the number of points can never be a prime: the order of the subgroup defining the Koblitz curve over the prime field always divides the order of the group over the extension field.

In the next section we present a hash function based on elliptic curves. Our construction discards the y-coordinate of the result of the compression function and uses m-tuple scalar multiplication to increase the throughput.

3. Security of the Compression and the Hash Function

In elliptic curve cryptography one considers the set of all solutions to a non-singular cubic equation $y^2 + a_1xy + a_3 = x^3 + a_2x^2 + a_4x + a_6$ over a finite field F_{p^k} . The set forms an additive group, where the neutral element O is the point at infinity. Given two points U and V on a cryptographically strong curve[2] the fastest known algorithms for the computation of the discrete logarithm $\log_V(U)$ have exponential running time in the bit-size of the representation of points [29].

Let E be a cyclic group of points of a cryptographically strong elliptic curve where ord(E) = q is a large prime number and let ℓ be the largest integer such that $2^\ell < q/2$. Let $S_1,\ldots,S_m \in E - \{O\}$ be randomly chosen pairwise distinct elements. Let $\varphi : E \to \{0,1\}^u$ be an efficiently computable mapping which is injective with respect to the x-coordinates, i.e., $\varphi(P)$ is a representation

[2] We refer for example to [6] for a list of requirements for elliptic curves to be cryptographically strong.

of the affine x-coordinate [3] of P with the additional property that if $\varphi(P) = \varphi(P')$ then we have $P = -P'$ or $P = P'$, and the value u is the size of the representation of the x-coordinate.

Given a string $x \in \{0,1\}^{m\ell}$ the compression function $H : \{0,1\}^{m\ell} \to \{0,1\}^u$ is defined by

$$H(x) = \varphi(x_1 \cdot S_1 + \ldots + x_m \cdot S_m)$$

where $x = x_1 | \ldots | x_m$ is the concatenation of the strings $x_1, \ldots, x_m \in \{0,1\}^{\ell}$.

Theorem 1: Let n be the bit-size of the representation of elements in E. If there exists an algorithm A which computes collisions for H in time t and with probability p then there exists an algorithm A_0 which computes the discrete logarithm $\log_V(U)$ for given points $U, V \in E - \{O\}$ in time $t + O(n^3 m)$ and with probability $2p/m$.

Proof: The algorithm A_0 of the attacker works as follows: we choose randomly numbers $1 < j < m$, $1 < w_i < q$ for $1 < i < m$ and define the compression function $H'(x) = \varphi(x_1 \cdot S'_1 + \ldots + x_m \cdot S'_m)$, where $S'_i = w_i V$ for $1 < i < m$, $i \neq j$ and $S'_j = w_j U$. If the S'_i, $1 < i < m$, are not pairwise distinct then we either found the discrete logarithm or we choose new numbers w_i. Next, we use A to find a collision for H'. Let $x, x' \in \{0,1\}^{m\ell}$ be the output of A such that $H'(x) = H'(x')$ and $x_i \neq x_j$. The properties of φ imply that the points on the curve E which correspond to the preimages of φ have the same x-coordinate. Therefore, either the equation

$$(x_1 - x'_1) \cdot S'_1 + \ldots + (x_m - x'_m) \cdot S'_m = O$$

or

$$(x_1 + x'_1) \cdot S'_1 + \ldots + (x_m + x'_m) \cdot S'_m = O$$

holds. Furthermore, it follows from the restriction $0 \leq x_i, x'_i < p/2$ for $1 < i < m$ that there exist at least two indices $1 \leq c, d \leq m$ that do not vanish, i.e., $x_c \neq x'_c \bmod q$ and $x_d \neq x'_d \bmod q$. If $j \in \{c, d\}$ we can solve the equation given by the collision for the point U and get the discrete logarithm $\log_V(U)$:

$$U = -\left(\frac{1}{w_j (x_j \pm x'_j)} \sum_{1<i<m} w_i (x_i \pm x'_i) \right) V .$$

It is easy to check which sign yields the discrete logarithm. If $j \notin \{c, d\}$ then it is possible that $x_j = x'_j \bmod q$ and the attack fails.
Since all elements of $E - \{O\}$ are generators of the cyclic group the distributions of the tuples $(S'_1, \ldots, S'_m)$ are identical for all choices of j and the behaviour of A on input H' is independent from j. The success probability $2p/m$ of the attack follows. The black-box reduction of A_0 involves m scalar multiplications for the randomization step and several computations in the field Z_q for the final computation of $\log_V(U)$. The complexity of these computations can be upper bounded by $O(n^3 m)$. □

The hash function G for messages of arbitrary length is built by iterating the compression function H. Every message $w \in \{0,1\}^*$ must be padded to get a multiple of the block length $m\ell - u$ and the length of w must be encoded. Let $w_1 | \ldots | w_r$ be the padded and encoded message in blocks of length $m\ell - u$. The hash value $G(w_1 | \ldots | w_r)$ is defined recursively:

$$G(w_1) = H(IV | w_1)$$
$$G(w_1 | \ldots | w_i) = H(G(w_1 | \ldots | w_{i-1}) | w_i) \quad \text{for } i > 1$$

where $IV \in \{0,1\}^u$ is a fixed initial value. If the initial value is chosen such that IV is not a valid representation of a x-coordinate, i.e., $IV \notin \varphi(E)$, then the proof technique in [11] shows that it is sufficient to encode only the length of the last block (before padding) instead of the length of the complete message to get a postfix-free encoding of arbitrary length messages.[4]

Since the padding and encoding scheme of the messages assures that a collision for G implies the existence of a collision for H the security of the scheme follows with the same bounds on the probability of success as in Theorem 1.

When implementing a concrete hash function H it follows from the construction that the discrete logarithms of the elements $S_1, \ldots, S_m$ must be kept secret. For example, a trusted third party could choose and publish the elements.

[3] The point O is mapped to a fixed string that is different from all images of points in $E - \{O\}$.

[4] If fixed points of the compression function can be computed in polynomial time encoding only the length of the last block is easily susceptible to the attacks in [18] resulting in a decreased preimage resistance of the hash function.

4. Implementation

The proof of security does not impose constraints on the parameters of the proposed hash function. The characteristic of the underlying finite field is arbitrary and can be adapted to the hardware in the targeted systems. It is clear, that the performance of the hash function is not comparable to standard hash algorithms since scalar multiplications on elliptic curves have to be computed. But a sophisticated choice of the parameters increases the performance drastically, such that the data rate becomes acceptable for applications with higher security requirements.

In our implementation we compared the performance of hash functions defined over prime and extension fields of characteristic $p > 3$. We did not consider fields of characteristic 2 since according to the performance comparisons in [15] the arithmetic might not be competitive without coprocessor. The arithmetic for the hash function based on elliptic curves over prime fields is built on the open source multiple precision library GMP [14]. Since the running time of the modular reduction in GMP does not depend on the Hamming weight of the modulus, curve and modulus were chosen randomly.

For the implementation of the hash function based on elliptic curves over extension fields we used optimal extension fields (OEF) for the arithmetic and implemented the basic OEF arithmetic completely in assembly language. In the next section we outline the chosen parameters and some of the optimizations.

OEFs F_{p^k} have the following properties for speeding up the modular reduction step [1]:

- p is a pseudo-Mersenne prime $p = 2^n \pm c$, where, $\log_2(c) < n/2$.
- field elements are represented using a polynomial basis given by a sparse irreducible polynomial $x^k \pm \omega$.

Furthermore, it is recommended to select a prime extension degree greater than 7 to prevent the GHS attack generalized by Diem in [12].

4.1. Choice and Implementation of the Optimal Extension Field

For the implementation of the finite field arithmetic over F_{p^k} we use $p = 2^{29} - 511$, degree of extension $k = 11$, and a polynomial basis given by the irreducible polynomial x^{11}-2. The modulus p is a pseudo-Mersenne prime with a low Hamming weight representation $p = 2^{29} - 2^9 + 1$.[5] Due to this representation reduction modulo p can be done by four shifts, two subtractions, and two additions. Moreover, the prime p is chosen in such a way that during a modular multiplication in F_{p^k} all intermediate sums of (possibly reduced modulo x^{11}-2, i.e., doubled) partial products of coefficients of the operands can be stored in 64-bit registers without overflow. Reduction modulo p must only be done once at the end of the computation of the coefficients of the product.

Using Intel's SSE2 assembly instruction set on Pentium 4 [16, 17] it is possible to parallelize part of the finite field arithmetic over F_{p^k}. The single instruction multiple data concept (SIMD) of SSE2 instructions and the 128-bit registers allow the computation of two partial products at the same time. For addition and subtraction in F_{p^k} it is even possible to compute and reduce four coefficients simultaneously using SIMD instructions operating on four doublewords.

The performance was measured on a common PC equipped with a Pentium 4 HT processor (Northwood core, 3.2 GHz, 512 kByte 2-nd level cache) and 2 GByte RAM running SuSE Linux 9.2. The program was compiled with GNU gcc 3.3.4.

Table 1: Performance in the finite field F_{p^k}

operation	***Time in ns***
Addition	16.7
Subtraction	25.6
Frobenius endomorphism	53.1
Squaring	102.8
Multiplication	144.8
Inversion	1924.3

4.2. Selecting Elliptic Curve Parameters

In our implementation of the hash function we use the Koblitz curve $y^2 = x^3 + ax + b$ with the parameters a = 468383287, b = 63579074, and $p = 2^{29} - 511$. The curve has 3·178961333 points over the prime field F_p. The order of the group of points over F_{p^k} has the prime factorization

[5] Prime numbers with low Hamming weight have been used before in [27, 5] to speed up modular reduction in large prime fields.

3·178961333·198922362779901429842787\
170820607991513538200253821938 2554\
833826430409188656090806536760 1.

The largest prime factor of the order has length 290 bits and the hash result has length u = 319 bit.

4.3. The Scalar Multiplication on Koblitz Curves

Several implementations of elliptic curve cryptosystems based on OEFs were reported in [1, 28, 21, 31]. In [19] Kobayashi pointed out the suitability of Koblitz curves for a fast scalar multiplication using the Frobenius endomorphism ϕ over F_{p^k} . It can be lifted to a homomorphism ϕ_E on E:

$$\phi_E : E \to E$$
$$(x, y) \mapsto (\phi(x), \phi(y)) = (x^p, y^p)$$

The linearity of ϕ_E can be easily verified using the affine addition law [15]. Since the endomorphism ring (E) is isomorphic to $Z[\phi]$ every element $s \in (E)$ can be represented by a finite power series in ϕ_E :

$$s = \sum_{i=0}^{\ell} s_i \phi_E^i, \quad where\,|\, s_i \,| \le p/2 \; for \; 0 < i \le \ell.$$

Müller [25] and Kobayashi [19] prove that the length ℓ of the power series is upper bounded by $\lfloor 2\log_p(s) \rfloor + 3$. The coefficients s_i can be computed with the algorithm in [19] or [1]. Since ϕ_E^k is the identity on E the length of the series can be shortened further to k-1:

$$s = \sum_{i=0}^{k-1} (s_i + s_{i+k} + s_{i+2k})\phi_E^i.$$

Transforming the scalar s to this Frobenius representation leads to a fast algorithm consisting of k-1 computations of the iterated Frobenius homomorphism and k scalar multiplications in E with smaller scalars $\tilde{s}_i = s_i + s_{i+k} + s_{i+2k}$ for $0 \le i < k$.

For our choice of the prime $p = 2^{29} - 511$, the extension degree k = 11, and the irreducible polynomial x^{11}-2 it follows that $p \equiv 1$ mod k and hence $\phi_E^i(x^j) = 2^{\lfloor jp^i/11 \rfloor} x^j$, i.e., the iterated Frobenius homomorphism $\phi_E^i(P)$ can be calculated with the complexity of a scalar multiplication in F_{p^k} if the appropriate powers of 2 are precomputed. Let $a = \sum_{j=0}^{k-1} \alpha_j x^j \in F_{p^k}$ be a coordinate of a point, then we have:

$$\phi_E^i(a) = \sum_{j=0}^{k-1} 2^{\lfloor jp^i/11 \rfloor} \alpha_j x^j.$$

The short scalar multiplications $\tilde{s}_i \cdot \phi_E^i(P)$ can be implemented using known scalar multiplication algorithms. Standard bit-by-bit algorithms as well as window methods are relevant [4, 20].

In the proposed hash function H we have to compute the sum of m scalar multiplications of generators of the cryptographically strong subgroup of E. Increasing the number of generators affects the storage requirement of the precomputed tables, in particular if window methods are applied. The linearity of the iterated Frobenius homomorphism can be utilized to decrease the number of point additions on the curve by the technique of Lim and Hwang [20]. With this method the sum of several scalar multiplications is computed during one multiplication. Since the costs of Frobenius computations are much smaller than the complexity of an addition on the curve, it is recommended to neglect the computation of the Frobenius homomorphism in the precomputation phase in order to reduce the table size. Table 2 shows this relation for practically useful table sizes. The precomputed points are represented in affine coordinates, intermediate results of scalar multiplications are given in Jacobian coordinates [8].

Table 2: Memory requirements for precomputed tables in kByte

Window width	Number of generators						
	2	3	4	5	6	7	8
1	0.4	1.2	3.5	10.4	31.3	94.0	282.0
2	2.1	14.7	103.0	722.0	-	-	-
4	41.3	1280.0	-	-	-	-	-

We have implemented several combinations with respect to the number of generators and window sizes in the m-tuple scalar multiplication and measured the performance of the resulting hash functions. The results are listed in Table 3.

Table 3: Data rate of the hash function in kByte/s

Window width	*Number of generators*	*Data rate*
1	6	197.9
2	3	125.0
3	2	130.0

5. Conclusion

We proposed a provably secure hash function based on elliptic curves following the lines of [7]. The hash function is not as fast as the standard hash algorithms, but the speed is sufficient for implementation in systems with higher security requirements.

In our implementation example we used Koblitz curves over OEFs. The security of the hash function is still preserved if the underlying field is a prime field or a finite field of characteristic 2. Actually the choice of the field depends on the hardware components in the particular system.

Furthermore, we studied several techniques for the implementation of the scalar multiplication on Koblitz curves. It turns out that the Frobenius representation of the scalar helps to lower the running time of scalar multiplications and that standard bit-by-bit algorithms will be the best choice in terms of storage requirements if the number of generators is increased.

Acknowledgments

The authors would like to thank Erwin Heß and Torsten Schütze for their help and encouragement during the implementation of the OEF arithmetic and the writing of the article.

References

[1] D.V. Bailey and C. Paar. Optimal Extension Fields for Fast Arithmetic in Public-Key Algorithms. In H. Krawczyk, editor, Advances in Cryptology – CRYPTO 98, volume 1462 of Lecture Notes in Computer Science, pages 472–485. Springer-Verlag, 2000.

[2] E. Biham and R. Chen. Near-Collisions of SHA-0. In M. Franklin, editor, Advances in Cryptology – CRYPTO 2004, volume 3152 of Lecture Notes in Computer Science, pages 290–305. Springer-Verlag, 2004.

[3] E. Biham, R. Chen, A. Joux, P. Carribault, C. Lemuet, and W. Jalby. Collisions of SHA-0 and Reduced SHA-1. In R. Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 36–57. Springer-Verlag, 2005.

[4] I. Blake, G. Seroussi, and N. Smart. *Elliptic Curves in Cryptography*. Cambridge University Press, 1999.

[5] M. Brown, D. Hankerson, J. Lopez, and A. Menezes. Software Implementation of the NIST Elliptic Curves over Prime Fields. Technical Report CORR 2000-58, Centre for Applied Cryptography, University of Waterloo, 2000.

[6] BSI (Bundesamt für Sicherheit in der Informationstechnik). Geeignete Kryptoalgorithmen zur Erfüllung der Anforderungen nach § 17 Abs. 1 bis 3 SigG vom 22. Mai 2001 in Verbindung mit Anlage 1 Abschnitt I Nr. 2 SigV vom 22. November 2001. Available at http://www.bsi.bund.de/esig/basics/techbas/krypto/index.htm, November 2004.

[7] D. Chaum, E. van Heist, and B. Pfitzmann. Cryptographically Strong Undeniable Signatures Unconditionally Secure for the Signers. In J. Feigenbaum, editor, *Advances in Cryptology – CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 470–484. Springer-Verlag, 1991.

[8] H. Cohen, A. Miyaji, and T. Ono. Efficient Elliptic Curve Exponentiation Using Mixed Coordinates. In K. Ohta and D. Pei, editors, *Advances in Cryptology – ASIACRYPT'98*, volume 1514 of *Lecture Notes in Computer Science*, pages 51–65. Springer-Verlag, 1998.

[9] S. Contini, A.K. Lenstra, and R. Steinfeld. VSH, an Efficient and Provable Collision Resistant Hash Function. In S. Vaudenay, editor, *Advances in Cryptology – EUROCRYPT'06,* volume 4004 of *Lecture Notes in Computer Science,* pages 165—182. Springer-Verlag, 2006. Available at http://eprint.iacr.org/ Cryptology ePrint Archive, Report 2005/193, 2005.

[10] J. Daemen and V. Rijmen. AES Proposal: Rijndael, 1998. Available at citeseer.ist.psu.edu/daemen98aes.html.

[11] I.B. Damgård. A Design Principle for Hash Functions. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO 1989*, volume 435 of *Lecture Notes in Computer Science*, pages 416–427. Springer-Verlag, 1990.

[12] C. Diem. The GHS Attack in Odd Characteristic. *J. Ramanujan. Math. Soc.*, 18(1):1–32, 2003.

[13] H. Dobbertin, A. Bosselaers, and B. Preneel. RIPEMD-160: A Strengthened Version of RIPEMD. In D. Gollmann, editor, *Fast Software Encryption – FSE 1996*, volume 1039 of *Lecture Notes in Computer Science*, pages 71–82. Springer-Verlag, 1996.

[14] T. Granlund. GNU Multiple Precision Arithmetic Library. Available at http://www.swox.com/gmp/, 2004.

[15] D. Hankerson, A. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer-Verlag, 2004.

[16] Intel. *IA-32 Intel-Architecture Software Developer's Manual, Volume 2A*, 2004.

[17] Intel. *IA-32 Intel-Architecture Software Developer's Manual, Volume 2B*, 2004.

[18] J. Kelsey and B. Schneier. Second Preimages on n-Bit Hash Functions for Much Less than 2^n Work. In R. Cramer, editor, Advances in Cryptology – EUROCRYPT 2005, volume 3494 of Lecture Notes in Computer Science, pages 474–490. Springer-Verlag, 2005.

[19] T. Kobayashi, H. Morita, K. Kobayashi, and F. Hoshimo. Fast Elliptic Curve Algorithm Combining Frobenius Map and Table Reference to Adapt Higher Characteristic. In

J. Stern, editor, Advances in Cryptology – EUROCRYPT '99, volume 1592 of Lecture Notes in Computer Science, pages 176–189. Springer-Verlag, 1999.
[20] C.H. Lim and H. Hwang. Speeding up Elliptic Scalar Multiplication with Precomputation. In J. Song, editor, Information Security and Cryptology 99, volume 1787 of Lecture Notes in Computer Science, pages 102–119. Springer-Verlag, 1999.
[21] C.H. Lim and H.S. Hwang. Fast Implementation of Elliptic Curve Arithmetic in $GF(p^k)$. In H. Imai and Y. Zheng, editors, Public Key Cryptography – PKC 2000, volume 1751 of Lecture Notes in Computer Science, pages 405–421. Springer-Verlag, 2000.
[22] M. Matsui and S. Fukuda. How to Maximize Software Performance of Symmetric Primitives on Pentium III and 4 Processors. In H. Handschuh and H. Gilbert, editors, *Fast Software Encryption – FSE 2005*, volume 3557 of *Lecture Notes in Computer Science*, pages 411–424. Springer-Verlag, 2005.
[23] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.
[24] R.C. Merkle. One Way Hash Functions and DES. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO 1989*, volume 435 of *Lecture Notes in Computer Science*, pages 428–446. Springer-Verlag, 1990.
[25] V. Müller. Fast Multiplication on Elliptic Curves over Small Fields of Characteristic Two. *J. Crypt.*, 11(4):219–234, 1998.
[26] J. Nakajima and M. Matsui. Performance Analysis and Parallel Implementation of Dedicated Hash Functions. In L. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 165–180. Springer-Verlag, 2002.
[27] National Institute of Standard and Technology. Digital Signature Standard. FIPS Publication 186-2, 2000.
[28] J.H. Park, J.H. Cheon, and S.G. Hahn. New Type of Optimal Extension Fields and its Applications. preprint available at http://crypt.kaist.ac.kr/publications.html, 2005.
[29] G.C. Pohlig and M.E. Hellman. An Improved Algorithm for Computing Logarithms over *GF*(*p*) and its Cryptographic Significance. *IEEE Trans. Info. Theory*, 24:106–110, 1978.
[30] D.R. Stinson. *Cryptography - Theory and Practice*. CRC Press, first edition, 1995.
[31] Y. Tsuruoka and K. Koyama. Fast Computation over Elliptic Curves $E(F_{q^n})$ Based on Optimal Addition Sequences. *IEICE Trans. Fund.*, E84-A(1):114–119, January 2001.
[32] X. Wang, X. Lai, D. Feng, H. Chen, and H. Yu. Cryptanalysis of the Hash Functions MD4 and RIPEMD. In R. Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 1–18. Springer-Verlag, 2005.
[33] X. Wang and H. Yu. How to Break MD5 and Other Hash Functions. In R. Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 19–35. Springer-Verlag, 2005.
[34] C. De Canniere, F. Mendel and C. Rechberger. Collisions for 70-step SHA-1: On the Full Cost of Collision Search. To appear in Proceedings of Selected Areas in Cryptography – SAC2007, Ottawa, Canada

Anton Kargl received the MS degree in Mathematics from Ludwig-Maximilians-University, Munich, in 2002. Since 2004 he works in the department "Security" of Corporate Technology at Siemens AG.

Bernd Meyer received the MS degree in Computer Science in 1992 and the PhD degree in 1995 from Universität des Saarlandes in Saarbrücken (Germany). Since 1997 he works in the department "Security" of Corporate Technology at Siemens AG.

Susanne Wetzel received her MS degree in Computer Science in 1993 from the Universität Karlsruhe (Germany) and her PhD in 1998 from the Universität des Saarlandes in Saarbrücken (Germany). Subsequently, she worked at DaimlerChrysler Research (Stuttgart, Germany), Lucent Technologies Bell Laboratories (Murray Hill, USA), and RSA Laboratories (Stockholm, Sweden). Since 2002 she is an Assistant Professor in the Computer Science Deparment of Stevens Institute of Technology (Hoboken, USA).