Journal of the Brazilian Computer Society, 2026, 32:1, doi: 10.5753/jbcs.2026.5548

High-Performance Elliptic Curve Cryptography: A SIMD Approach to Modern Curves (Thesis Distillation)*

Armando Faz-Hernandez [**University of Campinas** | ***armfazh@ic.unicamp.br***]
Julio López [**University of Campinas** | ***jlopez@ic.unicamp.br***]

Institute of Computing, University of Campinas. 1251 Albert Einstein St, Campinas, SP, 13083-852, Brazil.

Received: 14 February 2025 • **Accepted:** 21 November 2025 • **Published:** 25 March 2026

Abstract Cryptography based on elliptic curves is endowed with efficient methods for public-key cryptography. Recent research has shown the superiority of the Montgomery and Edwards curves over the Weierstrass curves as they require fewer arithmetic operations. Using these modern curves has, however, introduced several challenges to the cryptographic algorithm's design, opening up new opportunities for optimization. Our main objective is to propose algorithmic optimizations and implementation techniques for cryptographic algorithms based on elliptic curves. In order to speed up the execution of these algorithms, our approach relies on the use of extensions to the instruction set architecture. In addition to those specific for cryptography, we use extensions that follow the Single Instruction, Multiple Data (SIMD) parallel computing paradigm. In this model, the processor executes the same operation over a set of data in parallel. We investigated how to apply SIMD to the implementation of elliptic curve algorithms. As part of our contributions, we design parallel algorithms for prime field and elliptic curve arithmetic. We also design a new three-point ladder algorithm for the scalar multiplication $P + kQ$, and a faster formula for calculating $3P$ on Montgomery curves. These algorithms have found applicability in isogeny-based cryptography. Using SIMD extensions such as SSE, AVX, and AVX2, we develop optimized implementations of the following cryptographic algorithms: X25519, X448, SIDH, ECDH, ECDSA, EdDSA, and qDSA. Performance benchmarks show that these implementations are faster than existing implementations in the state of the art. Our study confirms that using extensions to the instruction set architecture is an effective tool for optimizing implementations of cryptographic algorithms based on elliptic curves. May this be an incentive not only for those seeking to speed up programs in general but also for computer manufacturers to include more advanced extensions that support the increasing demand for cryptography.

Keywords: Cryptography, Elliptic Curves, Parallel Computing, SIMD, Diffie-Hellman, Digital Signatures

1 Introduction

Extensive research efforts have focused on delivering public-key cryptography securely and efficiently. Cryptography based on elliptic curves provides more efficient methods mainly due to the use of keys shorter than the ones used in well-known cryptosystems, such at the RSA cryptosystem by [Rivest *et al*., 1978], and in those based on the Discrete Logarithm Problem (DLP) by [ElGamal, 1985]. Elliptic curve cryptography has been endorsed by international agencies that have produced standards for its usage such as [NIST, 2000; ANSI, 1998; IEEE, 2000]. Despite these standards being in circulation for several years, a recent line of research proposes a *shift to new elliptic curves* with the aim of improving rigidity on parameter selection and efficiency while preserving high-security guarantees.

With the avalanche of novel elliptic curve proposals, such as the ones highlighted by [Bernstein and Lange, 2015], new challenges have appeared. Former investigations focused on elliptic curves given in the Weierstrass model; however, there is still room for optimizing the operations of alternative elliptic curve models, including the *Montgomery* curves by [Montgomery, 1987], the *Edwards* curves introduced by [Edwards, 2007], which were further extended by [Bernstein *et al*., 2008]. The new algorithms for these curves must likely be adapted, or otherwise reformulated considering the upsides and downsides of each model. New improvements could arise by analyzing the algorithms from theoretical, computational, and practical standpoints. Therefore, designing cryptographic algorithms, implementing and putting them in practice is a task currently in progress.

From the computational perspective, a compelling approach for improving performance is using extensions to the instruction set architecture. Special attention is given to the extensions that support the *Single Instruction, Multiple Data* (SIMD) paradigm characterized in the parallel computing taxonomy by [Flynn, 1966]. Generally in this model, a *vector instruction* encodes an operation that is executed over several data units simultaneously. Figure 2 contrasts the number of

*Extended summary of a PhD thesis authored by Armando Faz Hernandez and supervised by Julio López. The full text by [Faz-Hernandez, 2022] is available at `https://hdl.handle.net/20.500.12733/6756`. Prior summaries appear at CSBC 2023 [doi: 10.5753/ctd.2023.230156], SBSeg 2024 [doi: 10.5753/sbseg_estendido.2024.241959], and CLEIej [doi: 10.19153/cleiej.27.3.3].

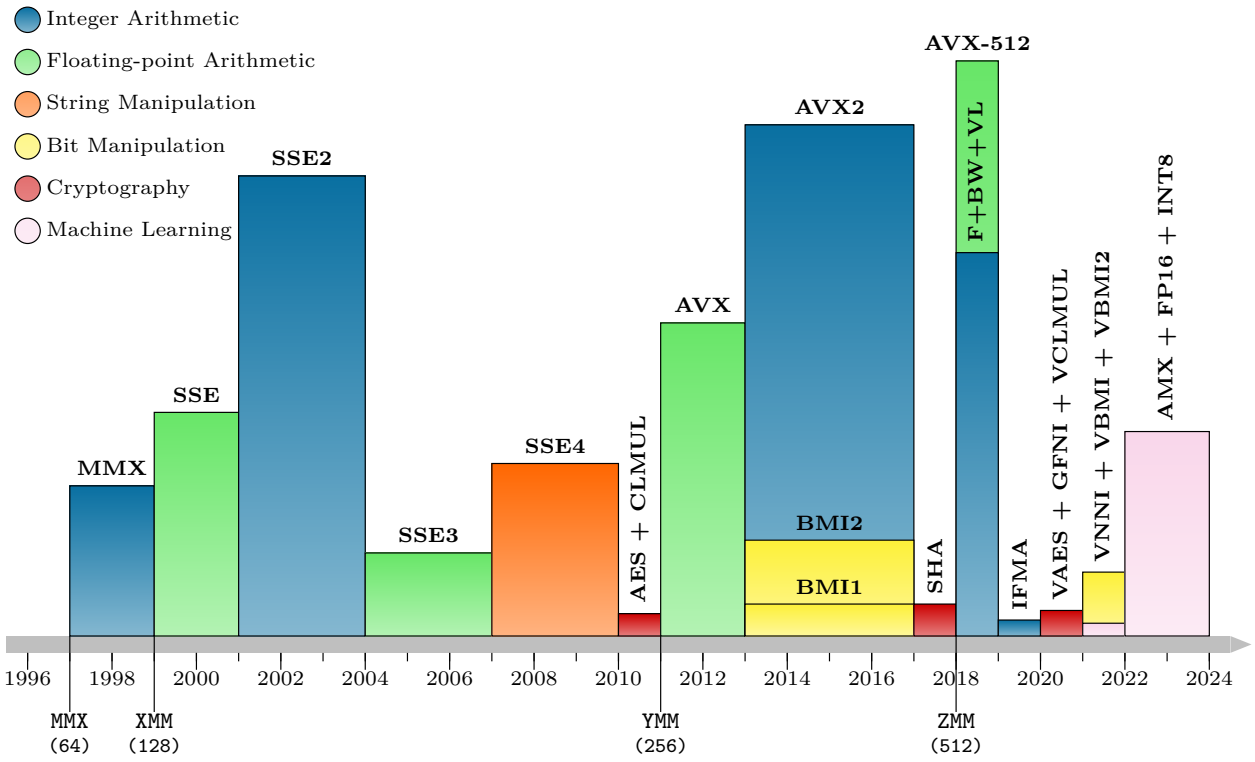
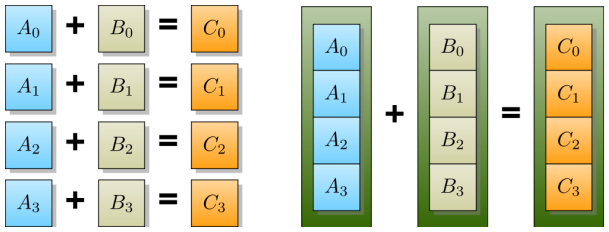


Figure 1. Evolution of SIMD Instructions. Each bar represents an instruction set showing its release date (in the horizontal dimension), the number of instructions (in the vertical dimension), and its domain of application (in the chromatic dimension). The marks show the release date of vector registers.

instructions used in scalar processing and vector processing.



(a) Scalar Processing
Four scalar instructions.

(b) Vector Processing
One vector instruction.

Figure 2. SIMD vector instructions.

Historically, SIMD processing has been shown effective in the high-performance computing area applied to graphics processing, scientific computing, mathematical simulation, among other domains. In the early days, SIMD execution units were exclusive of large workstations and supercomputers; nowadays, SIMD vector units can be found at commodity computers and portable devices [Thakkar and Huff, 1999; Intel Corporation, 2011]. Figure 1 shows the increasing addition of extensions to the instruction set architecture and their applicability to different domains. Hundreds of instructions have been added in order to support SIMD processing for integer and floating-point arithmetic. As can be seen, more recently the new instructions target more specific domains, for example, the inclusion of extensions tailored to accelerate cryptography and machine learning algorithms.

1.1 Research Objectives

The widespread availability of SIMD execution units in commodity computers, Internet servers, and mobile devices motivates their application to the implementation of cryptographic algorithms. Nonetheless, a few resources explain how to use SIMD units efficiently, and even fewer are dedicated to the case of elliptic curve cryptography, and the secure software development required in this domain. Moreover, it is unclear to what extent these computational resources can help to improve efficiency.

It is interesting to know how to effectively apply SIMD processing to elliptic curve cryptography. In particular, implementing the algorithms derived from the recent proposals of elliptic curves making use of the most advanced vector instructions and other extensions found in contemporary computer architectures. For this reason, it is imperative to investigate how to design new algorithms and data structures and/or how to adapt the existing ones so that implementations take full advantage of SIMD processing.

We claim that the execution of algorithms for elliptic curve cryptography can be accelerated through a combination of algorithmic optimizations, implementation techniques, and the use of SIMD processing and other hardware extensions.

To support this assumption, we investigate algorithmic optimizations and look for implementation techniques for elliptic curve algorithms emphasizing the application of SIMD parallel processing.

An objective of our study is to close the gap between theory and practice. For instance, in addition to proposing parallel algorithms, we also cover their implementation in software. We highlight some issues on security and performance arisen during development and propose some solutions for them. The design of our proposed algorithms also considers the capabilities and limitations of the computer architecture under evaluation.

Our research aims to define an efficient SIMD application strategy. Current computer architectures support hundreds of SIMD instructions including SSE, AVX, AVX2, and AVX512. Moreover, the number of more advanced instructions is gradually increasing in the upcoming computer architectures (cf. Figure 1). Part of this research is to give guidance on the use of SIMD instructions, to identify some of their limitations, and to show how to apply them to elliptic curve cryptography.

1.2 Contributions

Our contributions are the union of several layers of improvements comprising: algorithmic optimizations for elliptic curve cryptography, practical implementation techniques using SIMD processing for field operations, and the immediate applicability of our findings in high-level applications such as isogeny-based cryptography, hashing to curves, and hybrid public-key encryption. More details can be found in the full text by [Faz-Hernandez, 2022].

2 Algorithmic Optimizations

For Montgomery curves, we introduce a new *Three-point Ladder Algorithm* that calculates the x-coordinate of $P + kQ$, where P and Q are points on an elliptic curve E, and k is a secret integer. Our method, shown in Algorithm 1, exhibits a regular execution pattern taking as input the x-coordinate of P, Q, and $Q - P$, hence the name of three-point ladder. At each iteration, a point doubling ($2R_2$) and a differential addition ($R_2 +_{(R_1)} R_0$) are performed. To deal with secret values, the algorithm uses a conditional swap function (cswap), provided either by hardware or software, that interchanges R_0 and R_1 only if the bit b is set. All operations must be implemented using constant-time operations and no branches depending on secret data help prevent against timing attacks.

Algorithm 1 Three-Point Ladder Algorithm for $P + kQ$.

Input: x_P, x_Q, x_{Q-P} are the x-coordinate of $P, Q, Q-P \in E \setminus \{\mathcal{O}, (0,1)\}$; and integers (n, k) such that $0 \leq k < 2^n$ and $n \geq 1$.
Output: x_{P+kQ} is the x-coordinate of $P + kQ$.
1: Let $(k_{n-1}, \ldots, k_0)_2$ be the n-bit representation of k and define $k_{-1} = 0$.
2: $R_0 \leftarrow x_{Q-P}$, $R_1 \leftarrow x_P$, $R_2 \leftarrow x_Q$
3: **for** $i \leftarrow 0$ **to** $n - 1$ **do**
4: $\quad b \leftarrow k_i \oplus k_{i-1}$
5: $\quad R_0, R_1 \leftarrow \text{cswap}(R_0, R_1, b)$
6: $\quad R_2, R_0 \leftarrow 2R_2, R_2 +_{(R_1)} R_0$
7: **end for**
8: $R_0, R_1 \leftarrow \text{cswap}(R_0, R_1, k_{n-1})$
9: **return** R_1

Our algorithm improves in three aspects. First, it saves one third of arithmetic operations than the previously-known three-point ladder algorithm by [Jao *et al.*, 2014]. Second, when P and Q are known in advance, the algorithm allows faster execution by employing precomputation. Third, when precomputation is used, fetching precomputed values from memory requires non-secret indexes, which naturally prevents against side-channel attacks. Figure 3 exemplifies the operation of the multiplication algorithm.

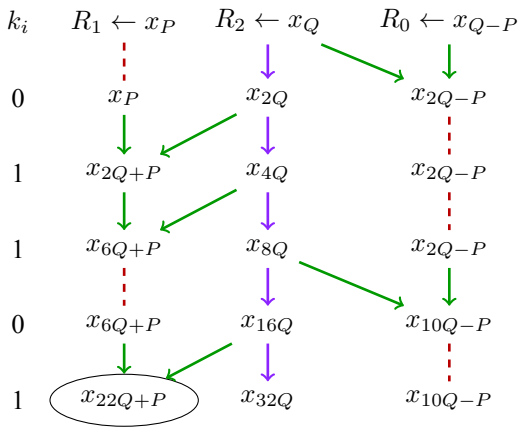


Figure 3. New Three-point Ladder Algorithm. Calculating x_{P+22Q} from x_P, x_Q, x_{Q-P}.

3 Implementation Techniques

In the following sections, we demonstrate techniques to speed up implementations of arithmetic operations over prime fields and elliptic curves whenever SIMD and other extensions to the instruction set architecture are available.

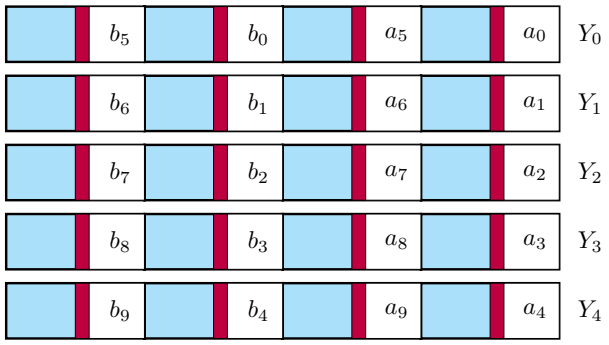
3.1 Prime Field Arithmetic using SIMD

We initially focus on the SIMD parallel processing of prime field arithmetic. To do so, we show *data structures* and *representation of numbers* suitable for parallel processing. Our study covers four families of prime moduli corresponding to the ones used in recent proposals of new elliptic curves. For each family, we show how to perform field operations using scalar and vector instructions. Our benchmark analysis shows that improvements in performance are more significant when operating over larger numbers.

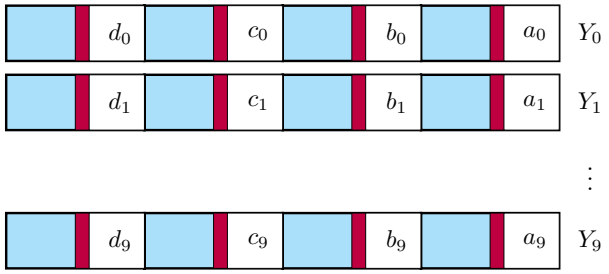
For smaller prime fields, manipulating data inside vector registers using permutation instructions has a negative effect on performance. This is explained because in the targeted computer architecture, the AVX2 permutation instructions have a higher latency than other vector instructions. Thus, the vectorized implementation of smaller prime fields has a notorious overhead limiting the amount of improvement.

A better approach that employs vector units more efficiently is taking the SIMD's essence to higher abstraction levels. We follow the notion of *n-way operations* using the n words of a vector register for calculating n *field operations* in parallel. This approach is motivated due to the overheads of using SIMD instructions to perform single field operations. Figure 4 shows how to distribute the individual words of a prime field element into a vector register. For example, note that one can store data units in such a way to perform either two-way or four-way operations. Using the AVX-512 instruction set, one extend this idea further to prepare data for performing eight-way operations.

In addition to the SIMD extensions, we studied the efficient application of other hardware extensions. We developed efficient implementations of field arithmetic using the `mulx`



(a) $\langle A, B\rangle$: Data storage for a two-way operation.



(b) $\langle A, B, C, D\rangle$: Data storage for a four-way operation.

Figure 4. Data structures for SIMD operations.

instruction and the `adcx/adox` instructions, which are part of, respectively, the BMI and ADX instruction sets. Using these instructions, our implementations render better performance than using basic instructions. Nonetheless, our vectorized implementations still offer superior improvements to prime fields of larger size.

Armed with n-way field operations, we turned our attention to investigate parallel algorithms for elliptic curve arithmetic that benefit from them.

3.2 Elliptic Curve Arithmetic using SIMD

We target the formulation of parallel algorithms for elliptic curves in three different models: Weierstrass, Montgomery, and Edwards curves.

For Weierstrass curves, we adapt the $\mathbb{F}_q$-complete formulas in such a way that point addition is performed by two parallel units. This enables the use of two-way field operations. The explicit parallel algorithms together with timings for the P-384 curve are available in a previous work by [Faz-Hernández and López, 2016].

For Montgomery curves, we showed how to calculate the Montgomery ladder step using two parallel units. The common implementation strategy for these two curve models consists on using the 256-bit AVX2 vector unit for simulating two 128-bit parallel units. Thus, each 128-bit unit can also be seen as two 64-bit parallel units that are dedicated to field arithmetic. One reason that supports this approach is because it reduces the use of expensive permutation instructions. Thus, our goal is to minimize the overheads we observed in the vectorization of smaller prime fields. The findings for Montgomery curves can be found at [Faz-Hernández and López, 2015].

For Edwards curves, we focused on parallel algorithms for point addition, point doubling, and scalar multiplication. Our implementation strategy is to perform four-way operations using the 256-bit vector unit. Then, we developed parallel algorithms for point addition (and doubling) using four-way field operations. Additionally, we constructed a four-way point addition that allowed us to perform parts of the scalar multiplication in parallel. The design of all parallel algorithms has the purpose to minimize the use of costly permutation instructions. The combined application of these strategies results on the acceleration of scalar multiplications that ultimately achieves a speedup to the higher level cryptographic algorithms.

4 High-Performance Implementation of Cryptographic Algorithms

Building on top of the improvements on the prime field and elliptic curve arithmetic, we found their direct applicability for speeding up some cryptographic algorithms. Specifically, we developed vectorized implementations of the ECDH and ECDSA with the P-384 curve; the X25519, X448, and SIDH Diffie-Hellman protocols; and the EdDSA, qDSA and hash-based digital signature schemes. In all cases, we observed performance improvements by using vector instructions.

4.1 The X25519 and X448 Protocols

The Diffie-Hellman protocol can be efficiently instantiated with Montgomery curves. This protocol allows two parties to agree on a shared secret and is widely used to establish secure communications. Its performance-critical operation is scalar multiplication: given an integer k and a point $P = (x_P, y_P)$ on an elliptic curve, a scalar multiplication algorithm calculates $Q = kP = (x_Q, y_Q)$. [Montgomery, 1987] proposed a special form of elliptic curves for speeding up computations. In this setting, operations require only the x-coordinate of points, so they are correct up to the sign. Montgomery devised a faster algorithm for calculating x_Q given k and x_P. Fortunately, the x-coordinate is more than enough to accomplish the Diffie-Hellman protocol.

We propose the use of the three-point ladder (Algorithm 1) to calculate kP in [Oliveira *et al.*, 2018]. Our approach relies on an auxiliary low-order point S, so we first compute $Q' = S + \frac{k}{h}P$ using the three-point ladder, where h is the curve's order cofactor. For security, it is required to remove any low order points. One can get rid of S by calculating $Q = hQ'$ because the order of S is at most h obtaining the desired result $Q = kP$.

We apply this strategy to the X25519 and X448 Diffie-Hellman protocols. We also specialize the algorithm to the case when P is known in advance. Part of these results are published at [Faz-Hernández *et al.*, 2019]. Table 1 shows a comparison of several X25519 implementations. It is evident that the use of advanced SIMD instruction sets reduces the latency of X25519. Table 2 shows timings of the X25519 and X448 protocols measured on different platforms such as the Haswell, Skylake and Tiger Lake micro-architectures.

Table 1. Timings of the X25519 Diffie-Hellman protocol. Entries are 10^3 clock cycles.

Implementation	Inst. Sets	Haswell	Skylake
[Moon, 2012]	SSE2	237.6	196.2
[Moon, 2012]	x64	166.3	140.0
[Faz-Hernández and López, 2015]	AVX2	156.5	137.8
[Chou, 2016]	AVX	155.9	137.2
[Oliveira *et al*., 2018]	x64, BMI2, ADX	144.3	111.2
[Faz-Hernández *et al*., 2019]	AVX2	121.0	99.4
[Faz-Hernández *et al*., 2019]	AVX2, AVX-512	—	87.3
[Faz-Hernández *et al*., 2019]	AVX-512	—	81.6

Table 2. Timings of the X25519 and X448 Diffie-Hellman protocols, and the Ed25519 and Ed448 signature schemes. Entries are 10^3 clock cycles.

Algorithm	Operation	Haswell	Skylake	Tiger Lake
X25519	Key Gen.	43.7	34.5	18.2
	Shared Secret	121.0	99.4	50.7
X448	Key Gen.	129.0	107.7	53.7
	Shared Secret	428.1	364.2	168.1
Ed25519	Key Gen.	42.8	34.8	18.4
	Sign	48.6	39.5	20.1
	Verify	156.0	123.3	77.1
Ed448	Key Gen.	126.7	104.9	54.8
	Sign	132.7	110.1	57.4
	Verify	465.8	409.5	193.6

4.2 The SIDH Protocol

Post-Quantum Cryptography is a branch of cryptography looking for secure algorithms resistant against adversaries with quantum computing power. In this field, isogeny-based cryptosystems have been formulated by looking at the elliptic curves as the vertex set of a regular graph, and *isogenies*, algebraic maps that link two elliptic curves, as the edges. Given a starting elliptic curve, it is easy to reach to other curves by computing an isogeny; however, finding the path of isogenies that connects two arbitrary elliptic curves in the graph is known to be hard using either a classical or a quantum computer, as shown by [Jao and De Feo, 2011].

[Velú, 1971] showed explicit formulas for isogenies that require knowledge of the isogeny's kernel. The kernel is calculated as $P + kQ$, where P and Q are points on an elliptic curve, and k is a secret integer chosen uniformly at random.

We found direct application of our three-point ladder algorithm for calculating the kernel of isogenies. The previous three-point ladder algorithm by [Jao and De Feo, 2011] performs two differential additions and one differential doubling per bit of the scalar. In contrast, our three-point ladder algorithm reduces the operations to only one differential addition and one differential doubling per bit. Theoretically, our algorithm saves around a third of the arithmetic operations when computing the kernel of the isogeny. We validate these approach experimentally by developing an optimized implementation of the Supersingular Isogeny Diffie-Hellman (SIDH) protocol by [Jao and De Feo, 2011]. The description of our three-point ladder algorithm and its implementation can be found at [Faz-Hernández *et al*., 2018]. Table 3 (as appears in the original article) shows the timings measured in a Skylake micro-architecture. The timings confirm the theoretical estimates, and in total, our optimized SIDH implementation provides $1.33\times$ faster operations.

Table 3. Timings of the SIDH-751 v2 implementation.

SIDH Operation		Skylake (10^6 cycles)		
		CLN[1]	FLOR[2]	Speedup
Key Generation	Alice	35.7	26.9	1.33×
	Bob	39.9	30.5	1.31×
Shared Secret	Alice	33.6	24.9	1.35×
	Bob	38.4	28.6	1.34×

[1] [Costello *et al*., 2016]
[2] [Faz-Hernández *et al*., 2018]

In addition, we showed an *optimized formula for tripling points* of Montgomery curves. That is, given a point P, it calculates $3P$. This operation is relevant for multi-base scalar multiplication methods as well as in the SIDH protocol. SIDH requires to evaluate $3^n P$ for an integer $n > 0$. By applying our formula, we reduce the total number of field operations by an observable margin. We acknowledge some trade-offs against formulas independently proposed by [Costello and Hisil, 2017].

4.3 The EdDSA Signature Scheme

For digital signatures, the dominant operation of EdDSA is the elliptic curve scalar multiplication. Figure 5 shows a breakdown of the internal operations of the EdDSA signature scheme contrasting the timings measured by our vectorized implementation. Our implementation of Ed25519, published at [Faz-Hernández *et al*., 2019], reduces by 19 % the latency of signing and by 24 % the latency of signature verification. Table 2 shows timings measured on Haswell, Skylake and Tiger Lake micro-architectures of the Ed25519 and Ed448 signature schemes.

In February 2023, [NIST, 2023] approved the standardization of EdDSA. In practical terms, it means that EdDSA is endorsed to be used on Internet communications massively. This is relevant to secure communication protocols such as SSL/TLS, SSH, VPN, and others. Our contributions on accelerating the performance of EdDSA are pertinent.

4.4 The qDSA Signature Scheme

The qDSA signature scheme by [Renes and Smith, 2017] also uses the fast arithmetic of Montgomery curves. In this case, signing requires to perform a scalar multiplication kP using

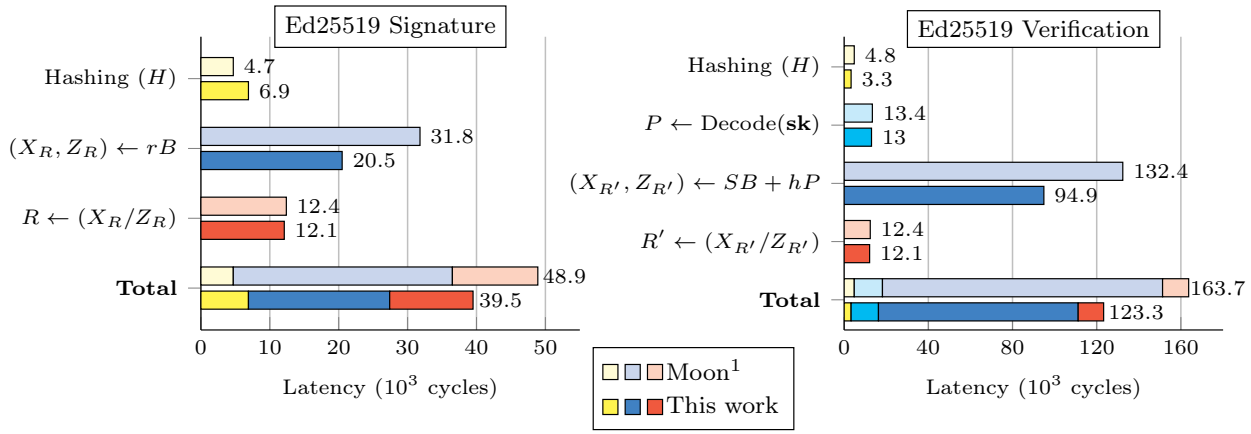


Figure 5. Breakdown of Ed25519's internal operations.

only the x-coordinates of the points. In a previous work [Faz-Hernández *et al*., 2017], we apply a similar approach as the one for the Diffie-Hellman protocol resulting on 35 % faster signatures by employing the three-point ladder algorithm. In addition, we propose an alternative signature verification procedure that checks for a stronger signature verification.

4.5 Hash-based Signatures with SHA-NI

The availability of the SHA New Instructions (SHA-NI) allowed us to reevaluate the performance of SHA-256. Hash functions are relevant for their widespread application on digital signature schemes and hash-based cryptographic algorithms. Introduced by Intel in [Gulley *et al*., 2013], SHA-NI augments the instruction set architecture with seven instructions that calculate parts of the SHA-256 cryptographic hash function. Looking at opportunities to employ these instructions efficiently, we develop a *four-way pipelined implementation* of SHA-256, i.e., it hashes four independent same-length messages simultaneously. This function is an instance of a more generic functionality called multiple buffer hashing. Figure 6 shows the performance of our implementation as the number and length of messages increases.

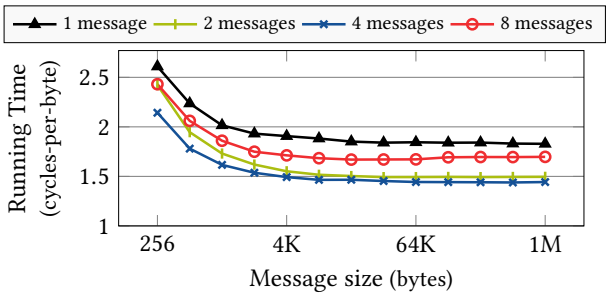


Figure 6. Performance comparison SHA-256 hash (multiple buffer).

Based on these experimental measurements, we determine the optimal number of messages to be processed by our pipelined implementation. Observe that hashing eight messages produces diminishing returns, which is explained by the increased memory used and a higher pressure on register allocation. Thus, we conclude that hashing four messages exhibits the best performance.

We use this multiple buffer hashing function on the implementation of the XMSS and XMSS^{MT} hash-based signatures. These signature schemes are in the portfolio of quantum-resistant algorithms and are specified at the [RFC 8391] document as well as recommended by NIST [Cooper *et al*., 2020].

Using SHA-NI, we observed that signature operations run up to four times faster than using a non-hardware aided implementation. Moreover, the performance is slightly better than implementing a four-way version with SIMD vector instructions. Figure 7 shows the latency of an XMSS signature using the SHA-NI and the AVX2 SIMD instructions measured, respectively, in the Zen and Kaby Lake micro-architectures. The performance analysis presented in this section is available at [Faz-Hernandez *et al*., 2018].

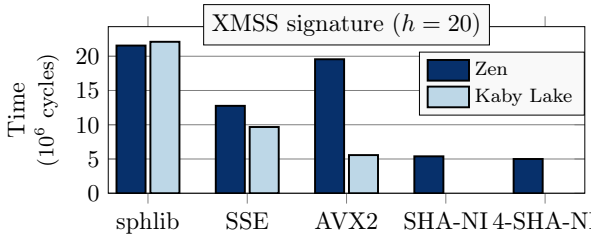


Figure 7. Timings of XMSS signing measured on Zen and Kaby Lake.

5 Further Applications

Several works have leveraged the foundational contributions of the thesis. This section highlights relevant and novel cryptographic algorithms benefiting from its proposals.

5.1 SQISign using the Three-Point Ladder

SQISign is a quantum-resistant signature scheme introduced in 2020 by [De Feo *et al*., 2020] that requires the calculation of isogenies between elliptic curves. Recently, it qualified to the second round of the [NIST, 2024] project called "Post-Quantum Cryptography: Additional Digital Signature Schemes". Among the contenders, SQISign's keys and signatures are in the shorter size range; and conversely, it is significantly slower to be used in practice. SQISign needs more optimizations for reducing its execution time.

In [Faz-Hernandez and López, 2024], we look for similarities with SIDH that help to speed up SQISign. We applied our three-point ladder algorithm to SQISign and investigated

its impact on the latency of signature operations. In this signature scheme, verifying a signature consists on validating whether the challenge isogenies were computed honestly, and to do so, several isogenies are performed by first calculating their kernel. Like in SIDH, our three-point ladder algorithm is directly applied to kernel calculation.

Using the SQISign's accompanying reference implementation[1], we measure the time taken to verify a signature for the security parameters at Level 1. Timings are shown in Table 4. From the experimental measurements, we observe that signature verification gets 9 % faster just by replacing the three-point ladder algorithm.

Table 4. Timings of SQISign verification at Level 1.

Three-point Ladder Algorithm	[Jao and De Feo, 2011]	[Faz-Hernández *et al.*, 2018]
SQISign Verification	20×10^6 cycles	18×10^6 cycles

A recent work by [Aardal *et al.*, 2024] presents state-of-the-art techniques for SQISign implementation. Besides the kernel calculation for isogenies, they employed our methods in [Faz-Hernández *et al.*, 2018] for implementing fast field arithmetic achieving significant savings. Another implementation of SQISign[2] performs $P + kQ$ invoking a function for $k_0P + k_1Q$ with $k_0 = 1$. In this case, our three-point ladder algorithm can also be applied. SQISign is a promising signature algorithm, however more investigation is needed for reducing its execution time before it gets used in practice.

5.2 Hash to Curve using SIMD

A hash function H that maps bit strings to points on an elliptic curve E over a finite field F is known as *hashing to curve*. This function plays an important role in the random oracle model when group-based cryptographic algorithms are instantiated with elliptic curves. A general strategy for constructing H is to compose a deterministic encoding with a cryptographic hash function. [Brier *et al.*, 2010] proved that

$$H(m) = f(h_0(m)) + f(h_1(m))$$

is indifferentiable whenever f is the Icart's encoding, and h_0 and h_1 are independent hash functions mapping to F. Like Icart's, some other constructions have been proposed. Despite these constructions having different parameters depending on the elliptic curve and the underlying field, their execution pattern is similar. The [RFC 9380] document presents state-of-the-art hash functions for the most common elliptic curves used in practice.

In [Faz-Hernández and López, 2020], we speed up the execution of hashing to curve functions through SIMD parallelism. Note that the internal operations of H can be split in two sub-tasks: clearly, $P_0 = f(h_0(m))$ and $P_1 = f(h_1(m))$ are independent. Also, since the internal algorithms exhibit a regular execution pattern, both calculations can run in parallel evenly. After the parallel stage is done, the final result is aggregated. This processing is shown in Figure 8.

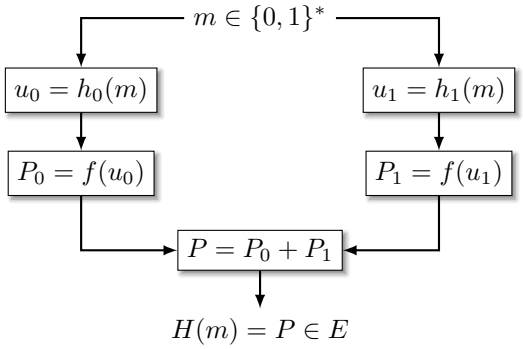


Figure 8. Parallel hashing of $m \in \{0,1\}^*$ to a point $P = H(m) \in E$.

To validate this approach, we implement a hash to curve function H compatible with [RFC 9380] for the Edwards curve used in Ed25519. An encoding f is usually represented as algebraic rational functions over F, so f only requires arithmetic operations in the field. For its implementation, we make use of the two-way field operations implemented with SIMD vector instructions to calculate two instances of f simultaneously. Figure 9 shows how the speedup factor varies as the message length increases. For short inputs, the encoding dominates the computation limiting the speedup factor to around 1.4×. For messages longer than 1 Kb, the cost of message hashing starts to dominate the total cost of H obtaining a speedup factor closer to 2× (the ideal factor).

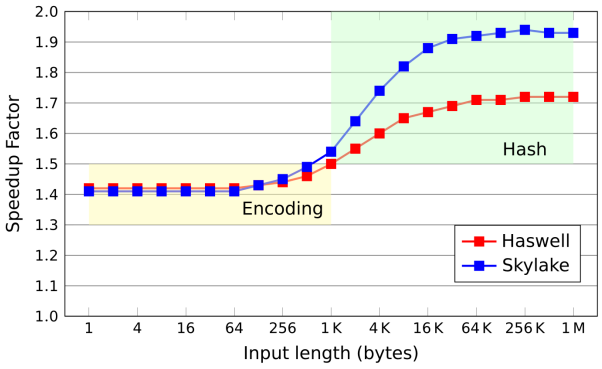


Figure 9. Speedup factor of vectorized implementation of hash to curve function for the Ed25519's curve.

5.3 Faster HPKE Encryption with AVX512

We present novel results about the implementation of the standard specification [RFC 9180] for hybrid public-key encryption using the AVX512 vector instructions.

Secure communication is a central problem in cryptography. Suppose Alice wants to send a message to Bob such that only Bob can read it. Data encryption emerges as the most natural solution for this problem. If only symmetric-key cryptography is used, Alice and Bob must know a shared key for encrypting (and decrypting) messages; however, Alice is able to decrypt messages too, which unmet the initial requirement. On the other hand, using only public-key cryptography,

[1] Source code at: `https://github.com/SQISign/sqisign/`
[2] Source code at: `https://github.com/SQISign/sqisign-ec23/`

Alice encrypts the message using Bob's public key ensuring that only Bob decrypts it. Unfortunately, public-key encryption algorithms are expensive and only short messages can be encrypted. Therefore, a cryptosystem that addresses these shortcomings is needed.

5.3.1 The HPKE Algorithm

Hybrid encryption is an approach that takes the best of both worlds: it has a public-key component called key encapsulation mechanism (KEM) (defining the `KeyGen`, `Encap`, `Decap` algorithms), and a symmetric-key component called data encapsulation mechanism (DEM) (defining the `Seal` and `Open` algorithms). Let λ be the security parameter, Bob generates his pair of keys as $(pk_B, sk_B) \leftarrow_{\$} \texttt{KeyGen}(\lambda)$. When Alice wants to send a message pt to Bob, they proceed as follows.

Alice (pk_B)		Bob (pk_B, sk_B)
$(k, c) \leftarrow \texttt{Encap}(pk_B)$		
$ct \leftarrow \texttt{Seal}(k, pt)$		
	$\xrightarrow{ct, c}$	
		$k \leftarrow \texttt{Decap}(sk_B, c)$
		$pt \leftarrow \texttt{Open}(k, ct)$

A well-known hybrid encryption algorithm is the Diffie-Hellman Integrated Encryption Scheme (DHIES, formerly known as DHAES) introduced by [Bellare and Rogaway, 1997], and [Abdalla *et al*., 1999]. DHIES works on groups and relies its security on the Gap Diffie-Hellman problem. This same algorithm when instantiated with elliptic curves is known as ECIES. In practice, several variants of ECIES exist, however most of them are incompatible.

For interoperability reasons, [Barnes *et al*., 2022] proposed a hybrid encryption algorithm called HPKE. It is based on elliptic curves and is secure against chosen ciphertext attacks. HPKE's specification is the [RFC 9180] document published by the Internet Research Task Force.

HPKE also allows Bob to authenticate messages sent by Alice. To do so, Alice generates $(pk_A, sk_A) \leftarrow_{\$} \texttt{KeyGen}(\lambda)$, which are used by the `AuthEncap` and `AuthDecap` functions. Like `Encap`, `AuthEncap` takes as input Bob's public key, as well as Alice's private key. `AuthDecap` guarantees that the KEM shared secret was generated by Alice. Analogously, `AuthDecap` is similar to `Decap` taking also Alice's public key as input, thus Bob is assured that the KEM shared secret was generated by Alice. The authenticated version of HPKE proceeds as follows.

Alice (pk_A, sk_A, pk_B)		Bob (pk_B, sk_B, pk_A)
$(k, c) \leftarrow \texttt{AuthEncap}(pk_B, sk_A)$		
$ct \leftarrow \texttt{Seal}(k, pt)$		
	$\xrightarrow{ct, c}$	
		$k \leftarrow \texttt{AuthDecap}(sk_B, c, pk_A)$
		$pt \leftarrow \texttt{Open}(k, ct)$

HPKE constructs the KEM using modern Diffie-Hellman functions such as X25519. The `KeyGen` algorithm is the X25519's key generation function. HPKE also uses an auxiliary key derivation function (`KDF`) that takes as input the key material and a context string for generating a symmetric key k for DEM. [RFC 9180] provides the exact definition of all HPKE operations, briefly shown in Algorithms 2, 3, 4, and 5.

Algorithm 2 $\texttt{Encap}(pk_B)$

1: $sk_E \leftarrow_{\$} \{0,1\}^{256}$
2: $c \leftarrow \texttt{X25519.KeyGen}(sk_E)$
3: $h \leftarrow \texttt{X25519.Shared}(sk_E, pk_B)$
4: $k \leftarrow \texttt{KDF}(h, c \parallel pk_B)$
5: **return** k, c

Algorithm 3 $\texttt{Decap}(sk_B, c)$

1: $h \leftarrow \texttt{X25519.Shared}(sk_B, c)$
2: $k \leftarrow \texttt{KDF}(h, c \parallel pk_B)$
3: **return** k

Algorithm 4 $\texttt{AuthEncap}(pk_B, sk_A)$

1: $sk_E \leftarrow_{\$} \{0,1\}^{256}$
2: $c \leftarrow \texttt{X25519.KeyGen}(sk_E)$
3: $h_0 \leftarrow \texttt{X25519.Shared}(sk_E, pk_B)$
4: $h_1 \leftarrow \texttt{X25519.Shared}(sk_A, pk_B)$
5: $k \leftarrow \texttt{KDF}(h_0 \parallel h_1, c \parallel pk_B \parallel pk_A)$
6: **return** k, c

Algorithm 5 $\texttt{AuthDecap}(sk_B, c, pk_A)$

1: $h_0 \leftarrow \texttt{X25519.Shared}(sk_B, c)$
2: $h_1 \leftarrow \texttt{X25519.Shared}(sk_B, pk_A)$
3: $k \leftarrow \texttt{KDF}(h_0 \parallel h_1, c \parallel pk_B \parallel pk_A)$
4: **return** k

5.3.2 Optimized AVX512 Implementation of HPKE

The performance-critical operation of HPKE is the execution of KEM. Each HPKE algorithm performs at least one X25519 operation, and one additional X25519 operation if sender's authentication is required. In Section 4.1, we show optimizations for X25519 that rely on AVX2 instructions and are readily available for implementing the KEM functions.

Looking for more optimizations for HPKE, we note that the encapsulation functions have independent X25519 computations enabling their execution in parallel. Moreover, both `AuthEncap` and `AuthDecap` calculate two X25519 shared secrets over independent data. This is exactly the execution pattern suitable for SIMD processing. Therefore, we implement *two-way parallel X25519* functions using AVX512 instructions by [Intel Corporation, 2018]. We employ the 512-bit registers as two AVX2 parallel units of 256-bit registers. Since the calculations in each unit are independent, there is no need to move data between units avoiding slow operations at all. We only move data at the beginning and at the end of the function for loading and storing the keys. This parallelization strategy also applies to the `Encap` function if we perform $\texttt{X25519.KeyGen}(x)$ equivalently as $\texttt{X25519.Shared}(x, 9)$.

This parallel strategy can be generalized to other KEMs, such as the X448 function and the Diffie-Hellman KEM based on prime order curves.

We conduct a performance benchmark of our implementation and compare against OpenSSL (v3.2.2) and BoringSSL (v0.20250114.0). We measure the latency of the KEM operations using X25519 and HKDF(SHA-256). Table 5 shows the timings measured on a Tiger Lake micro-architecture.

Table 5. Timings of HPKE with X25519 and HKDF(SHA-256). Entries are 10^3 clock cycles measured on Tiger Lake.

Code	ISA	`Encap`	`Decap`	`Auth Encap`	`Auth Decap`
OpenSSL	x64	302	168	431	312
BoringSSL	x64	264	155	405	296
This work	x64	225	160	358	284
	AVX2	209	142	325	251
	AVX512	211	142	248	203
Speedup AVX512 vs BoringSSL		1.25×	1.09×	1.63×	1.45×

The measurements consistently show that the vectorized AVX2 and AVX512 implementations outperform the non-vectorized x64 implementations. In comparison to AVX2, the AVX512 implementation exhibits a better performance for the authenticated operations validating our hypothesis. `Encap` is faster in AVX2 because `X25519.KeyGen` is calculated with a faster fixed-point scalar multiplication on the Edwards curve, unlike the two-way `X25519.Shared` that uses the Montgomery curve. Table 5 reports speedup factors against BoringSSL as is faster than OpenSSL. The AVX512 implementation is $1.09\times$-$1.25\times$ faster than BoringSSL's and is $1.45\times$-$1.63\times$ faster for the authenticated mode.

In HPKE moving from the basic mode to the authenticated mode incurs in overheads. For encapsulation, the sender takes $1.42\times$ more time using OpenSSL and $1.53\times$ using BoringSSL. The overheads are higher for decapsulation, the receiver spends almost twice the time, $1.85\times$ more time using OpenSSL and $1.90\times$ using BoringSSL. In our AVX512 implementation, the overheads are lower only $1.17\times$ for encapsulation and $1.42\times$ for decapsulation.

HPKE significantly benefits from the use of vectorized implementations. Using either AVX2 and AVX512, we reduce the execution time of the KEM operations for the basic and authenticated modes. Our implementations are available at `https://github.com/armfazh/hpke-simdium/`.

6 Conclusion

Supported by the experimentation performed, we conclude that the application of SIMD vector instructions reduces the execution time of several cryptographic algorithms. We remark that in order to get better performance algorithms must be adapted accordingly. Some of these changes appear naturally due to the SIMD parallel computing paradigm, however, others arose from the limitations of the instruction set used.

Our investigation provided optimizations and implementation techniques for accelerating cryptographic algorithms, resulting in performance within the vanguard of speed capabilities. The developed software implementations are readily accessible for reproduction and future extension. Furthermore, the identified trade-offs and limitations in these developments may offer valuable insights for subsequent research. We hope this work and the presented ideas motivate students and researchers in their projects.

Declarations

Acknowledgements

The authors would like to thank the anonymous referees for their valuable comments. Sincere appreciation to the organizing committee of the CTDSeg competition and the SBSeg 2024 conference.

Funding

The authors acknowledge support during the development of this research from Intel and Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) as part of the project "Secure Execution of Cryptographic Algorithms" (Grant No. 14/50704-7), and thematic project "Security and Reliability of Information: Theory and Applications" (Grant No. 13/25977-7).

Authors' Contributions

AFH: Writing – review & editing, Writing – original draft, Software, Methodology, Investigation, Conceptualization. JL: Conceptualization, Investigation, Supervision, Writing – review & editing. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Availability of data and materials

The software generated and/or analyzed during the current study is available at an institutional repository:
`https://gitlab.ic.unicamp.br/ra142685/phd_libs/`

References

Aardal, M. A., Adj, G., Alblooshi, A., Aranha, D. F., Canales-Martínez, I. A., Chávez-Saab, J., Gazzoni Filho, D. L., Reijnders, K., and Rodríguez-Henríquez, F. (2024). Optimized One-Dimensional SQIsign Verification on Intel and Cortex-M4. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2025(1):497–522. DOI: 10.46586/tches.v2025.i1.497-522.

Abdalla, M., Bellare, M., and Rogaway, P. (1999). DHAES: An Encryption Scheme Based on the Diffie-Hellman Problem. Available at: `https://eprint.iacr.org/1999/007`.

ANSI (1998). Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA). Technical Report ANSI X9.62-1998, American National Standards Institute. Available at: `https://webstore.ansi.org/Standards/ASCX9/ANSIX9621998`.

Barnes, R., Bhargavan, K., Lipp, B., and Wood, C. A. (2022). Hybrid Public Key Encryption. *The RFC Series*. RFC 9180. DOI: 10.17487/RFC9180.

Bellare, M. and Rogaway, P. (1997). Minimizing the Use of Random Oracles in Authenticated Encryption Schemes. In *Proceedings of the First International Conference on Information and Communication Security*, ICICS '97, page 1–16, London, UK, UK. Springer-Verlag. DOI: 10.1007/BFb0028457.

Bernstein, D. J., Birkner, P., Joye, M., Lange, T., and Peters, C. (2008). Twisted Edwards Curves. In Vaudenay, S., editor, *Progress in Cryptology – AFRICACRYPT 2008*, volume 5023 of *Lecture Notes in Computer Science*, page 389–405. Springer Berlin Heidelberg. DOI: 10.1007/978-3-540-68164-9_26.

Bernstein, D. J. and Lange, T. (2015). SafeCurves: choosing safe curves for elliptic-curve cryptography. Available at: `http://safecurves.cr.yp.to` Accessed 20 March 2015.

Brier, E., Coron, J.-S., Icart, T., Madore, D., Randriam, H., and Tibouchi, M. (2010). Efficient Indifferentiable Hashing into Ordinary Elliptic Curves. In Rabin, T., editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223, page 237–254. Springer Berlin Heidelberg. DOI: 10.1007/978-3-642-14623-7_13.

Chou, T. (2016). Sandy2x: New Curve25519 Speed Records. In Dunkelman, O. and Keliher, L., editors, *Selected Areas in Cryptography – SAC 2015*, page 145–160, Cham. Springer International Publishing. DOI: 10.1007/978-3-319-31301-6_8.

Cooper, D., Apon, D., Dang, Q., Davidson, M., Dworkin, M., and Miller, C. (2020). Recommendation for Stateful Hash-Based Signature Schemes. (NIST SP 800-208). DOI: 10.6028/NIST.SP.800-208.

Costello, C. and Hisil, H. (2017). A Simple and Compact Algorithm for SIDH with Arbitrary Degree Isogenies. In Takagi, T. and Peyrin, T., editors, *Advances in Cryptology – ASIACRYPT 2017*, page 303–329, Cham. Springer International Publishing. DOI: 10.1007/978-3-319-70697-9_11.

Costello, C., Longa, P., and Naehrig, M. (2016). Efficient Algorithms for Supersingular Isogeny Diffie-Hellman. In Robshaw, M. and Katz, J., editors, *Advances in Cryptology – CRYPTO 2016*, page 572–601, Berlin, Heidelberg. Springer Berlin Heidelberg. DOI: 10.1007/978-3-662-53018-4_21.

De Feo, L., Kohel, D., Leroux, A., Petit, C., and Wesolowski, B. (2020). SQISign: Compact Post-quantum Signatures from Quaternions and Isogenies. In Moriai, S. and Wang, H., editors, *Advances in Cryptology – ASIACRYPT 2020*, page 64–93, Cham. Springer International Publishing. DOI: 10.1007/978-3-030-64837-4_3.

Edwards, H. M. (2007). A normal form for elliptic curves. *Bulletin of the American Mathematical Society*, 44(3):393–422. DOI: 10.1090/S0273-0979-07-01153-6.

ElGamal, T. (1985). A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In Blakley, G. R. and Chaum, D., editors, *Advances in Cryptology. CRYPTO 1984*, page 10–18, Berlin, Heidelberg. Springer Berlin Heidelberg. DOI: 10.1007/3-540-39568-7_2.

Faz-Hernandez, A. (2022). *High-Performance Elliptic Curve Cryptography: A SIMD Approach to Modern Curves*. PhD thesis, University of Campinas, Campinas, Brazil. Available at: `https://hdl.handle.net/20.500.12733/6756`.

Faz-Hernandez, A. and López, J. (2024). High-Performance Elliptic Curve Cryptography: A SIMD Approach to Modern Curves (Extended Thesis Summary). *CLEI Electronic Journal*, 27(3):1–8. DOI: 10.19153/cleiej.27.3.3.

Faz-Hernandez, A., López, J., and de Oliveira, A. K. D. S. (2018). SoK: A Performance Evaluation of Cryptographic Instruction Sets on Modern Architectures. In *Proceedings of the 5th ACM on ASIA Public-Key Cryptography Workshop*, APKC '18, page 9–18, New York, NY, USA. ACM. DOI: 10.1145/3197507.3197511.

Faz-Hernandez, A., Scott, S., Sullivan, N., Wahby, R. S., and Wood, C. A. (2023). Hashing to Elliptic Curves. *The RFC Series*. RFC 9380. DOI: 10.17487/RFC9380.

Faz-Hernández, A., Fujii, H., Aranha, D. F., and López, J. (2017). A Secure and Efficient Implementation of the Quotient Digital Signature Algorithm (qDSA). In Ali, S. S., Danger, J.-L., and Eisenbarth, T., editors, *Security, Privacy, and Applied Cryptography Engineering*, page 170–189, Cham. Springer International Publishing. DOI: 10.1007/978-3-319-71501-8_10.

Faz-Hernández, A. and López, J. (2015). Fast Implementation of Curve25519 Using AVX2. In Lauter, K. and Rodríguez-Henríquez, F., editors, *Progress in Cryptology – LATINCRYPT 2015*, volume 9230 of *Lecture Notes in Computer Science*, page 329–345. Springer International Publishing. DOI: 10.1007/978-3-319-22174-8_18.

Faz-Hernández, A. and López, J. (2016). Speeding up Elliptic Curve Cryptography on the P-384 Curve. In *XVI Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, volume 16, page 170–183. Sociedade Brasileira de Computação – SBC. DOI: 10.5753/sbseg.2016.19306.

Faz-Hernández, A. and López, J. (2020). Generation of Elliptic Curve Points in Tandem. In Moraes, I. M. and Kowada, L., editors, *XX Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, volume 20, page 1–9, Petrópolis, RJ, Brasil. Sociedade Brasileira de Computação. DOI: 10.5753/sbseg.2020.19230.

Faz-Hernández, A., López, J., and Dahab, R. (2019). High-performance Implementation of Elliptic Curve Cryptography Using Vector Instructions. *ACM Transactions on Mathematical Software (TOMS)*, 45(3):1–35. DOI: 10.1145/3309759.

Faz-Hernández, A., López, J., Ochoa-Jiménez, E., and Rodríguez-Henríquez, F. (2018). A Faster Software Implementation of the Supersingular Isogeny Diffie-Hellman Key Exchange Protocol. *IEEE Transactions on Computers*,

67(11):1622–1636. DOI: 10.1109/TC.2017.2771535.

Flynn, M. (1966). Very high-speed computing systems. *Proceedings of the IEEE*, 54(12):1901–1909. DOI: 10.1109/PROC.1966.5273.

Gulley, S., Gopal, V., Yap, K., Feghali, W., Gullford, J., and Wolrich, G. (2013). Intel ® SHA Extensions New Instructions Supporting the Secure Hash Algorithm on Intel ® Architecture Processors. Technical report, Intel Corporation. Available at: `https://software.intel.com/sites/default/files/article/402097/intel-sha-extensions-white-paper.pdf`.

Huelsing, A., Butin, D., Gazdag, S.-L., Rijneveld, J., and Mohaisen, A. (2018). XMSS: eXtended Merkle Signature Scheme. *The RFC Series*. RFC 8391. DOI: 10.17487/RFC8391.

IEEE (2000). IEEE Standard Specifications for Public-Key Cryptography. *IEEE Std 1363*, page 1–228. DOI: 10.1109/IEEESTD.2000.92292.

Intel Corporation (2011). Intrinsics for Intel Advanced Vector Extensions. Available at: `https://software.intel.com/en-us/isa-extensions`.

Intel Corporation (2018). The Intel® Advanced Vector Extensions 512 (Intel® AVX-512) Vector Length Extensions Feature on Intel® Xeon® Scalable Processors. Available at: `https://www.intel.com/content/www/us/en/developer/articles/technical/the-intel-advanced-vector-extensions-512-feature-on-intel-xeon-scalable.html?wapkw=avx512`.

Jao, D. and De Feo, L. (2011). Towards Quantum-Resistant Cryptosystems from Supersingular Elliptic Curve Isogenies. In Yang, B.-Y., editor, *Post-Quantum Cryptography: 4th International Workshop, PQCrypto 2011, Taipei, Taiwan, November 29 – December 2, 2011. Proceedings*, page 19–34, Berlin, Heidelberg. Springer. DOI: 10.1007/978-3-642-25405-5_2.

Jao, D., De Feo, L., and Plût, J. (2014). Towards Quantum-Resistant Cryptosystems from Supersingular Elliptic Curve Isogenies. *Journal of Mathematical Cryptology*, 8(3):209–247. DOI: 10.1007/978-3-642-25405-5_2.

Montgomery, P. L. (1987). Speeding the Pollard and Elliptic Curve Methods of Factorization. *Mathematics of Computation*, 48(177):243–264. DOI: 10.1090/S0025-5718-1987-0866113-7.

Moon, A. (2012). Implementations of a fast Elliptic-curve Diffie-Hellman primitive. Available at: `https://github.com/floodyberry/curve25519-donna/`.

NIST (2000). Digital Signature Standard (DSS). Technical Report FIPS PUB 186-2, National Institute of Standards and Technology. Available at: `http://csrc.nist.gov/publications/fips/archive/fips186-2/fips186-2.pdf`.

NIST (2023). Digital Signature Standard (DSS). (FIPS PUB 186-5):86. DOI: 10.6028/NIST.FIPS.186-5.

NIST (2024). Post-Quantum Cryptography: Additional Digital Signature Schemes. National Institute of Standards and Technology. Available at: `https://csrc.nist.gov/projects/pqc-dig-sig/round-2-additional-signatures`.

Oliveira, T., López, J., Hışıl, H., Faz-Hernández, A., and Rodríguez-Henríquez, F. (2018). How to (Pre-)Compute a Ladder. In Adams, C. and Camenisch, J., editors, *Selected Areas in Cryptography – SAC 2017*, page 172–191, Cham. Springer International Publishing. DOI: 10.1007/978-3-319-72565-9_9.

Renes, J. and Smith, B. (2017). qDSA: Small and Secure Digital Signatures with Curve-Based Diffie–Hellman Key Pairs. In Takagi, T. and Peyrin, T., editors, *Advances in Cryptology – ASIACRYPT 2017*, page 273–302, Cham. Springer International Publishing. DOI: 10.1007/978-3-319-70697-9_10.

Rivest, R. L., Shamir, A., and Adleman, L. (1978). A Method for Obtaining Digital Signatures and Public-key Cryptosystems. *Commun. ACM*, 21(2):120–126. DOI: 10.1145/359340.359342.

Thakkar, S. and Huff, T. (1999). Internet Streaming SIMD Extensions. *Computer*, 32(12):26–34. DOI: 10.1109/2.809248.

Velú, J. (1971). Isogénies entre courbes elliptiques. *Comptes rendus de l'Académie des Sciences de Paris*, 273:238–241. Available at: `https://gallica.bnf.fr/ark:/12148/bpt6k56191248`.