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

Armando Faz-Hernández, Julio López

Institute of Computing, University of Campinas.
1251 Albert Einstein, Cidade Universitária. Campinas, São Paulo. Brazil.

{armfazh,jlopez}@ic.unicamp.br

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.

1. Motivation

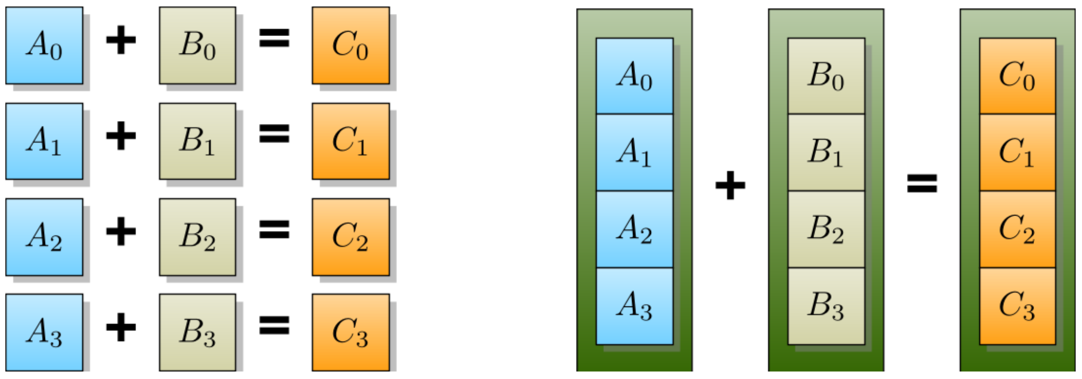
Extensive research efforts have focused on delivering public-key cryptography securely and efficiently. Cryptography based on elliptic curves provides efficient methods due to the use of keys that are shorter than the ones used in the Rivest-Shamir-Adleman (RSA)

[*] Summary of a PhD thesis by [Faz-Hernández 2022], the full text can be found at `https://hdl.handle.net/20.500.12733/6756`

cryptosystem [Rivest et al. 1978] and in algorithms based on the Discrete Logarithm Problem [ElGamal 1985]. Despite elliptic curve cryptography have been endorsed by international standardization agencies [NIST 2000, ANSI 1998, IEEE 2000] for several years, a recent line of research proposes a shift to new elliptic curves with the aim of improving 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, such as the *Montgomery* curves [Montgomery 1987] and the *Edwards* curves [Bernstein et al. 2008]. 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, the pathway for designing cryptographic algorithms, their implementation, and their put in practice is currently in progress.

From the computational perspective, a compelling approach for improving performance is using extensions to the instruction set architecture. There exist extensions that support the *Single Instruction, Multiple Data* (SIMD) paradigm characterized in Flynn's taxonomy [Flynn 1966] of parallel computing. In this model, a *vector instruction* encodes an operation that is executed over several data units simultaneously, as shown in Figure 1.



(a) Scalar (non-vector) Processing: Four scalar instructions perform the workload.

(b) Vector Processing: A single vector instruction performs the same workload.

Figure 1. 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; but nowadays, computer manufacturers have incorporated SIMD vector units [Thakkar and Huff 1999, ARM , Intel Corporation 2011]. Figure 2 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 instructions target more specific domains, for example, the inclusion of extensions tailored to accelerate cryptographic algorithms [Gueron 2009].

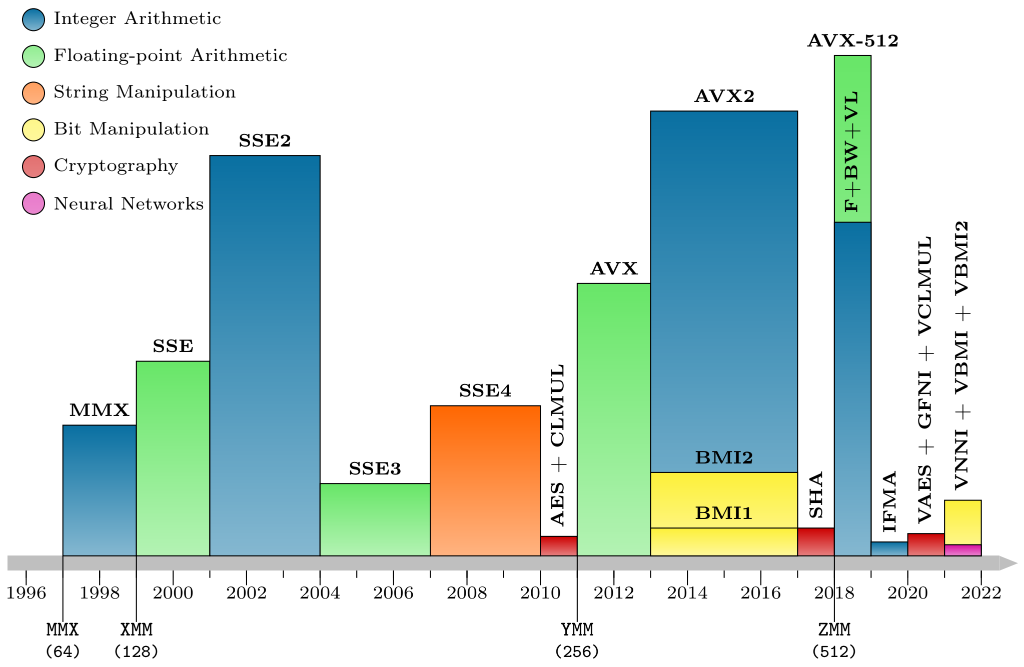


Figure 2. Evolution of SIMD Instructions. Each bar represents an instruction set showing its release date (in the horizontal dimension), the number of new instructions (in the vertical dimension), and the main application (in the color dimension). The milestones show the release date of vector registers.

1.1. Problem Statement

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 the implementation of elliptic curve cryptography. Especially to the algorithms derived from the recent proposals of elliptic curves and making use of the most advanced vector instructions [Intel Corporation 2011] and other extensions found in contemporary computer architectures. For this reason, it is imperative to investigate how to design new algorithms and data structures (or adapt the existing ones) so that implementations take full advantage of SIMD vector processing.

Thesis Statement 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.

1.2. Aims

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 arisen during development and propose some solutions for them. The design of our proposed algorithms considers the capabilities and limitations of the computer architecture studied.

Our research aims to enlighten a pathway for applying SIMD efficiently. Current computer architectures support hundreds of SIMD instructions including SSE, AVX, AVX2, AVX512, and others. Moreover, the number of instructions is gradually increasing in the upcoming computer architectures (cf. Fig. 2). 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.

2. Contributions

Our contributions to the Computer Science field are the union of several layers of improvements comprising algorithmic optimizations for elliptic curve cryptography, practical implementation techniques for SIMD vector processing of mathematical field operations, and the immediate applicability of our findings into current information security standards. Now, we briefly describe these contributions. More details can be found in the full text available at [Faz-Hernández 2022].

2.1. Algorithmic Optimizations

For Montgomery curves, we introduced a new *Three-point Ladder Algorithm* that calculates the x-coordinate of $P + kQ$, where P, Q are points on the curve and k is an integer. Our algorithm improves in three aspects. First, it requires fewer operations than previously-known algorithms [Costello et al. 2016, 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 prevents against side-channel attacks. Figure 3 exemplifies the operation of the multiplication algorithm.

We showed the immediate application of the algorithm to the Diffie-Hellman (DH) protocol [Diffie and Hellman 1976]. In fact, we apply it to concrete cryptographic algorithms, such as X25519, X448, qDSA with Montgomery curves, and SIDH/SIKE. The latter algorithm is part of Isogeny-based Cryptography, a branch of cryptography looking for secure algorithms resistant against adversaries with quantum computing power. By using our multiplication algorithm, all of these algorithms show better performance than previous approaches. The improvement is independent of the computer's architecture.

For Montgomery curves, we showed an *optimized formula for tripling points*, that is, given a point P, to calculate $3P$. This operation is relevant for multi-base scalar multiplication methods as well as for Isogeny-based Cryptography. For instance, the SIDH protocol requires to evaluate 3^nP 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 in [Costello and Hisil 2017].

2.2. Implementation Techniques

On the availability of SIMD and other extensions to the instruction set architecture, we speed up implementations of arithmetic operations over prime fields and elliptic curves.

$$\begin{tikzcd}
k_i & Q_1 \leftarrow S \arrow[d, dashed, no head] & Q_0 \leftarrow P \arrow[d] \arrow[dr] & Q_2 \leftarrow P - S \arrow[d] \\
0 & S \arrow[d] & 2P \arrow[d] \arrow[dl] & 2P - S \arrow[d, dashed, no head] \\
1 & 2P + S \arrow[d] & 4P \arrow[d] \arrow[dl] & 2P - S \arrow[d, dashed, no head] \\
1 & 6P + S \arrow[d, dashed, no head] & 8P \arrow[d] \arrow[dr] & 2P - S \arrow[d] \\
0 & 6P + S \arrow[d] & 16P \arrow[d] \arrow[dl] & 10P - S \arrow[d, dashed, no head] \\
1 & 22P + S & 32P & 10P - S
\end{tikzcd}$$

Figure 3. New Three-point Ladder Algorithm. Execution flow for calculating $22P + S$ given P, S, $P - S$ and $k = 22$.

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

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

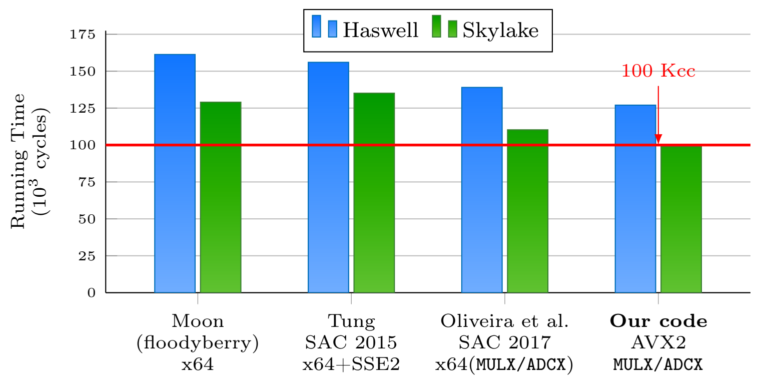
A better approach that employs vector units more efficiently is taking the SIMD's essence to higher abstraction levels. We follow the notion of *n-way operations* using the n words of a vector register for calculating n *field operations* in parallel. This approach was motivated due to the overheads of using SIMD instructions to perform single field operations. Armed with n-way field operations, we turned our attention to investigate parallel algorithms for elliptic curve arithmetic that benefit from them.

SIMD Implementation of Elliptic Curve Arithmetic. For Weierstrass curves, we adapt the $\mathbb{F}_q$-complete formulas for point addition, in such a way that point addition is performed by two parallel units, enabling the application of two-way field operations.

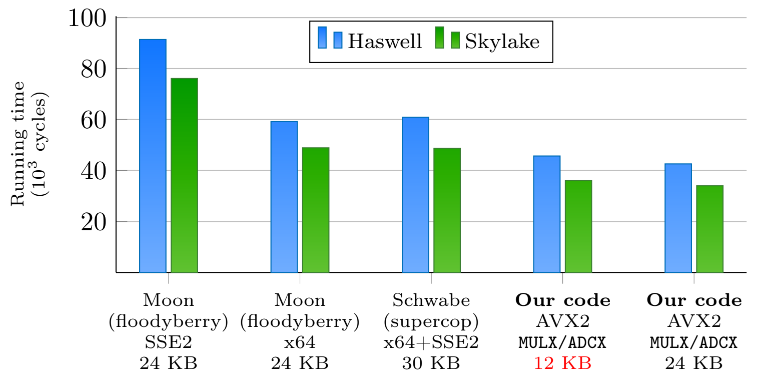
For Montgomery curves, we showed how to calculate the Montgomery ladder step using two parallel units. The common implementation strategy for these two curve models consists on using the 256-bit AVX2 vector unit for simulating two 128-bit parallel units. Thus, each 128-bit unit can also be seen as two 64-bit parallel units that are dedicated to field arithmetic. We follow this approach because it reduces the use of expensive permutation instructions; thus, minimizing the overheads observed in the vectorization of smaller prime fields.

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.

Optimized Implementation of Cryptographic Algorithms. Building on top of the improvements on the prime field arithmetic and the elliptic curve arithmetic, we found their direct applicability for speeding up some cryptographic algorithms. We developed vectorized implementations of the ECDH and ECDSA with the P-384 curve; the X25519, X448, and Supersingular Isogeny Diffie-Hellman protocols; and the EdDSA and qDSA digital signature schemes. In all cases, we observed improvements on performance by using vector instructions. Figure 4 shows the time latency savings on the X25519 and Ed25519 algorithms.



(a) Performance comparison of X25519.



(b) Performance comparison of Ed25519 (EdDSA).

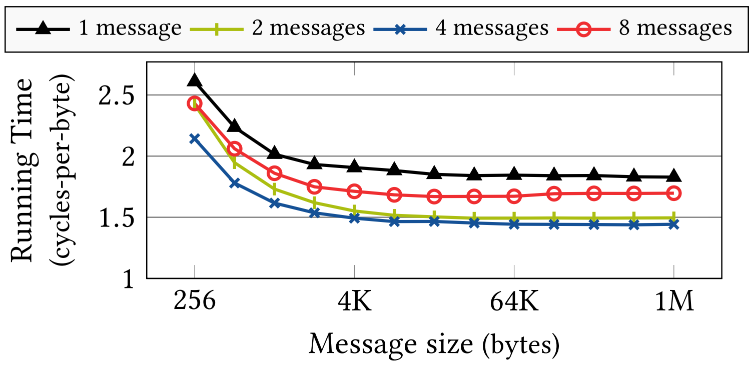
Figure 4. Performance benchmark on Haswell and Skylake micro-architectures.

In February 2023, the National Institute of Standards and Technology [NIST 2023] has approved the standardization of EdDSA, which in practical terms 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 this algorithm are pertinent.

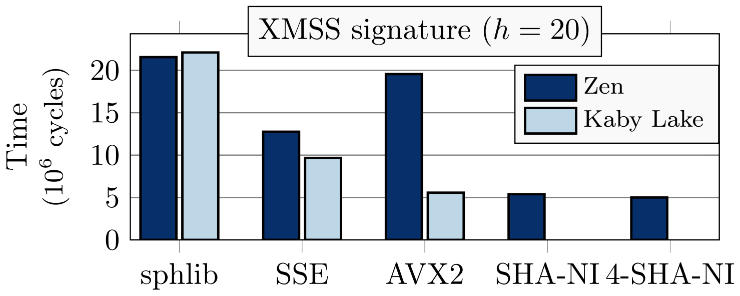
Implementations using other Extensions. In addition to the SIMD extensions, we studied the efficient application of other hardware extensions such as BMI2, ADX, and SHA-NI instructions.

We developed efficient implementations of field arithmetic using the MULX 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 offer superior improvements to prime fields of larger size.

The availability of SHA-NI allowed us to evaluate the performance of the SHA-256 cryptographic hash function. We developed a pipelined implementation that performs a four-way version of the SHA-256 function. We applied this function to the XMSS and XMSS^{MT} hash-based signatures, which are in the portfolio of quantum-resistant algorithms. 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.



(a) Performance comparison SHA-256 hash (multiple buffer).



(b) Performance comparison of XMSS.

Figure 5. Performance benchmark on Zen and Kaby Lake micro-architectures.

2.3. Software Libraries

We developed a set of software libraries that show implementation techniques and optimizations of the cryptographic algorithms mentioned above. Source code was optimized for SIMD processing, and performance benchmarks provide evidence of their superiority. To enable reproducibility, our libraries are available at public repositories and released under permissive software licenses. The code is also available at an institutional repository:

`https://gitlab.ic.unicamp.br/ra142685/phd_libs/`

Third-party Usage. Our x64 implementation of X25519 was included in the implementation of the Wireguard protocol [Donenfeld 2018], which offers a VPN-like secure communication tunel between remote machines. Wireguard was recently included in the kernel of Linux.

Derived from our work, Protzenko et al. [Protzenko et al. 2020] formally verified a x64 implementation of X25519, which closely follows our own implementation, proving the correctness of the code as part of the EverCrypt project.

Our three-point ladder algorithm and other implementation techniques were adopted by SIKE [Jao et al. 2017], a project part of the NIST's Post-Quantum Cryptography Standardization project [NIST 2016].

3. Conclusions

Based on the experimentation performed in our investigation, we conclude that the application of SIMD vector instructions does reduce the execution time of both prime field operations and elliptic curve arithmetic resulting in observable improvements in high-level cryptographic algorithms. However, we remark that in order to get better performance several changes in the algorithms are needed. Some of them are naturally motivated by the SIMD parallel computing paradigm, but others arose from the instruction set used.

Our investigation provided explicit optimizations and implementation techniques that resulted in a faster execution of cryptographic algorithms than those existent in the state of the art. Our software implementations render better performance when using AVX2 vector instructions for the X25519 and X448 Diffie-Hellman protocols, and the Ed25519 and Ed448 digital signature schemes. We also identified trade-offs and limitations of these developments, which can provide insights for future improvements. We hope our work and the ideas presented motivate to students and researchers on future projects.

References

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.

ARM. ARM NEON Intrinsics. `https://developer.arm.com/architectures/instruction-sets/intrinsics`.

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*, pages 389–405. Springer Berlin Heidelberg.

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

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*, pages 303–329, Cham. Springer International Publishing.

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*, pages 572–601, Berlin, Heidelberg. Springer Berlin Heidelberg.

Diffie, W. and Hellman, M. (1976). New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654.

Donenfeld, J. A. (2018). Wireguard Linux Compat. Patch to Wireguard: https://git.zx2c4.com/wireguard-linux-compat/commit/src/crypto/curve25519-x86_64.h?id=186be2742c948351c27bc068102252e10a28959b.

ElGamal, T. (1985). A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. pages 10–18.

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

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

Gueron, S. (2009). Intel's New AES Instructions for Enhanced Performance and Security. In Dunkelman, O., editor, *Fast Software Encryption: 16th International Workshop, FSE 2009 Leuven, Belgium, February 22-25, 2009 Revised Selected Papers*, pages 51–66, Berlin, Heidelberg. Springer.

IEEE (2000). IEEE Standard Specifications for Public-Key Cryptography. https://doi.org/10.1109/IEEESTD.2000.92292.

Intel Corporation (2011). Intel® Advanced Vector Extensions Programming Reference. https://software.intel.com/sites/default/files/m/f/7/c/36945.

Jao, D., Azarderakhsh, R., Campagna, M., Costello, C., Feo, L. D., Hess, B., Jalali, A., Koziel, B., LaMacchia, B., Longa, P., Naehrig, M., Renes, J., Soukharev, V., Urbanik, D., and Pereira, G. (2017). Supersingular Isogeny Key Encapsulation.

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.

Montgomery, P. L. (1987). Speeding the Pollard and Elliptic Curve Methods of Factorization. *Mathematics of Computation*, 48(177):243–264.

NIST (2000). Digital Signature Standard (DSS). Technical report, National Institute of Standards and Technology. http://csrc.nist.gov/publications/fips/archive/fips186-2/fips186-2.pdf.

NIST (2016). Post-Quantum Cryptography Standardization. National Institute of Standards and Technology. https://www.nist.gov/pqcrypto.

NIST (2023). Digital Signature Standard (DSS). Technical Report FIPS PUB 186-5, National Institute of Standards and Technology. https://doi.org/10.6028/NIST.FIPS.186-5.

Protzenko, J., Parno, B., Fromherz, A., Hawblitzel, C., Polubelova, M., Bhargavan, K., Beurdouche, B., Choi, J., Delignat-Lavaud, A., Fournet, C., Kulatova, N., Ramananandro, T., Rastogi, A., Swamy, N., Wintersteiger, C. M., and Zanella-Beguelin, S. (2020).

EverCrypt: A Fast, Verified, Cross-Platform Cryptographic Provider. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 983–1002.

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.

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