Energy Efficiency Analysis of Elliptic Curve based Cryptosystems

Tanushree Banerjee and M. Anwar Hasan
Department of Electrical and Computer Engineering
University of Waterloo
{tanushree.banerjee, ahasan}@uwaterloo.ca

***Abstract*—Energy consumption is an important factor for any cryptoscheme, implemented on devices having limited energy resource. In this paper, we analyze the energy consumption during the Diffie-Hellman key exchange implemented on both supersingular and ordinary elliptic curves. The former protocol is Supersingular Isogeny based Diffie-Hellman (SIDH) and the latter is Elliptic Curve based Diffie-Hellman (ECDH) respectively. Implementations are executed on 64 bit Intel Skylake processor. The energy consumption is analysed for the classical bit security levels of 128 and 192. In this paper, a detailed comparison of power and energy consumption by elliptic curve point addition and doubling operations is presented for affine and standard projective coordinates. Projective coordinates based elliptic curve operations are found to be around 50 to 60 times more energy efficient. We then analyze SIDH and ECDH, implemented using projective coordinates. Our results show that, SIDH consumes around 37 to 47 times more energy in comparison to ECDH for the above mentioned bit security levels.**

***Index Terms*—energy consumption, elliptic curve, isogeny, Diffie-Hellman key exchange, standard projective coordinates, affine coordinates**

I. Introduction

Elliptic Curve Cryptography (ECC) was proposed in 1985 independently by Neal Koblitz [1] and Victor Miller [2]. Since then a large variety of security implementations in public key cryptohraphy have been done using elliptic curves. This particular scheme is used in many practical applications [3] like smart grids, vehicular communication, RFID, secure shell (SSH) [4], transport layer security (TLS) [5], BitCoin [6] etc. ECC provides a secure means of exchanging keys among communicating hosts using the Diffie-Hellman (DH) key exchange algorithm [7], which is referred to as Elliptic Curve DH (ECDH) and relies on the difficulty of computing discrete logarithm problem on the curve. ECDH is however not secure against attacks using future quantum computers. In 2011, David Jao and De Feo introduced a post quantum security protocol based on the DH key exchange construction, commonly referred to as Supersingular Isogeny based Diffie-Hellman (SIDH) [8]. Amongst all the other known post quantum cryptoschemes such as code-, lattice-, hash-based cryptography, and multivariate cryptography, SIDH makes use of the smallest key size. The best known classical and quantum attacks against the underlying DH problem of SIDH are both exponential in the size of the underlying finite field, and their complexities. This makes SIDH quite promising as a post quantum crypto candidate.

The fundamental operation underlying ECC is ellipitic curve point multiplication, which in turn uses point doubling and addition. An elliptic curve can be represented using various coordinate systems [9]. For each such coordinate system, the formulae for computing point addition and doubling are different, as a result the speed of computation is also different. Therefore a good choice of coordinate system is an important factor for elliptic curve exponentiations. Affine coordinates and projective coordinates are well known [10]. Comparison of execution times between these two coordinates based elliptic curve point multiplication has been reported in the past [11]. To the best of our knowledge, there has not been any work done that reports the power and energy consumption values corresponding to the use of such coordinates. In this work, we provide an insight about the differences in energy consumption between affine and projective coordinate based point addition and doubling used in ECDH and SIDH. Our work shows that projective coordinates are relatively much more energy efficient. We then use such coordinates to report the differences in energy consumption between the entire ECDH and SIDH.

Rest of this paper is organised as follows. The next section provides some preliminaries about ECDH and SIDH parameters, followed by a discussion on methodologies in Section III. Section IV focuses on coordinate systems used in the elliptic curve point doubling and point additions. Next, Section V outlines the comparative analysis of the energy efficiency between standard projective coordinates and affine coordinates based operations on elliptic curve points. The following section analyses energy and power consumption efficiency between SIDH and ECDH cryptoschemes. Experimental validation of the energy consumption results are added in Section VII followed by concluding remarks in Section VIII.

II. Preliminaries

In this section we provide details of parameters used in the crypto protocols of ECDH and SIDH. Both of these schemes involve scalar point multiplication, and are defined on different prime fields. But SIDH requires isogeny evaluation and computation, unlike the ECDH scheme.

A. ECDH parameters

Elliptic curves defined over prime fields are considered in this paper, where the security level of the implementation is decided by the size of the chosen prime. Elliptic curve based cryptoschemes have an exponential time complexity with the smallest key size, amongst all other public key cryptoschemes. Based on the factors such as required security level and the underlying prime field, an appropriate curve on that field is to be chosen. In this paper, implementation of ECDH is chosen such that it provides the classical bit security of 128

TABLE I: Sizes of keys and the primes used in ECDH and SIDH [12] protocols

Classical bit security level	ECDH(Length of keys and primes in bits)				SIDH(Length of keys and primes in bits)			
	Prime	Private Key	Public Key	Shared Secret Key	Prime	Private Key	Public Key	Shared Secret Key
128	256	256	512	256	503	256	3024	1008
192	384	384	768	384	751	384	4512	1504

and 192. We have used (National Institute of Standards and Technology) NIST recommended curves P-256 and P-384 in this investigation [13]. The pseudo-random curves are in the short Weierstrass form [10] as given below :

$$y^2 = x^3 + a \cdot x + b \tag{1}$$

This curve is defined over the field F_{p_n}, where p_n is a prime of length 256 bits and 384 bits for providing 128 bits and 192 bits of security level, as recommended by NIST. The value of the coefficient b is also provided by NIST. The coefficient a is chosen to be -3, so that it would require fewer field computations for elliptic curve point operations. For the curve P-256, the prime used is [9] $2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$. For P-384, the prime used is $2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$. These primes have the special property that is they can be written as the sum or difference of small numbers to the powers of 2, which makes reduction algorithms with them easier.

B. SIDH parameters

The security of SIDH depends on its "hard problem", which states : *Given two supersingular elliptic curves say E and E' belonging to the same isogeny class defined over finite field of characteristic p_s, finding the isogeny mapping between the two curves $\phi : E \rightarrow E'$.*

Isogeny property is a group homomorphism, which is a rational map from the curve E to E' satisfying $\phi(0) = 0$ and $\phi(E) \neq 0$ [10]. The two elliptic curves E and E' are isogenous to each other over $F_{p_s^2}$ if and only if $\#E(F_{p_s^2}) = \#E'(F_{p_s^2})$. Given a finite subgroup $H \subseteq E(F_{p_s^2})$, if there exists such unique isogeny $\phi : E \rightarrow E'$ then the kernel(ϕ) = H, and the degree of ϕ is $|H|$. Velu′s formula [14] can be used to find the isogeny ϕ and the isogenous curve E' which is also denoted by E/H. For arbitrary subgroups, Velu′s formula is computationally impractical. Therefore this particular SIDH implementation uses isogenies over subgroups that are powers of 2 and 3. The public starting curve used for SIDH is

$$y^2 = x^3 + x \tag{2}$$

This is a special instance of the Montgomery curve of order $(2^{e_A} \cdot 3^{e_B})^2$ and e_A and e_B are the two primes that define the finite field $F_{p_s^2}$ where $p_s = (2^{e_A} \cdot 3^{e_B} - 1)$ [15]. In this particular implementation, values of e_A and e_B are 250 and 159 for 128 bits security and 372 and 239 for 192 bits security.

C. Remarks

In ECDH, all the computations are done on a fixed elliptic curve, unlike SIDH where computations are performed on different isogenous curves. Table I provides the public, private and shared secret key sizes used for various security levels with ECDH and SIDH. The public key size is almost 6 times larger in case of SIDH compared to ECDH for both the levels of security. This demands extra storage space in the device that implements the scheme. Private keys are of same size but the shared secret key of SIDH is again around 4 times larger than the ones used in ECDH [15], [12].

III. Methodology

In this investigation, power consumption is measured using Intel Power Gadget 3.5. ECDH and SIDH protocols are implemented in C and built on Visual Studio 2015, using Intel i7-6700 Skylake, 64 bit processor. Elliptic curve point operations are one of the most important steps in both the cryptoschemes. The computation of SIDH is adapted from Microsoft's implementation [12]. On the otherhand, ECDH protocol is implemented using the regular "Double and Add" algorithm [16] for public key and shared secret key generation. Power consumption corresponding to these schemes is then recorded while the codes are executed on the above mentioned processor. Energy efficiency is analysed by considering the cumulative power consumption throughout the execution time of these algorithms.

With respect to energy efficiency of ECDH, OpenSSL's implementation has also been considered. Since such implementations have undergone a lot of improvements and optimisations in terms of speed over a long period of time, it doesn't lead to a fair comparison with current SIDH implementations, which is still in the process of getting optimized. Moreover unlike Microsoft's SIDH implementation, ECDH implementation by OpenSSL uses Jacobian Projective coordinates and w-ary Non Adjacent Form(w-NAF) based point multiplication. Therefore in this paper, in order to make a sensible comparison, ECDH is implemented with regular scalar point multiplication using standard projective coordinates on short Weierstrass form of elliptic curve defined on the field F_{p_n}.

For the case of SIDH, computations are done on the curves defined over $F_{p_s^2}$, where length of the prime p_s is 503 bits and 751 bits. From this point onwards, in this paper we will refer to the curve used for SIDH implementation as C-503^2 and C-751^2 when length of the prime p_s corresponding to the underlying field $F_{p_s^2}$ is 503 bits and 751 bits respectively.

IV. Elliptic curve point addition and doubling

Affine coordinates are the most basic coordinate representation where a point P on the curve consists of the x and y coordinates that is (x_p, y_p). Whereas standard projective coordinates are represented as $P = (X : Y : Z)$ and $Z \neq 0$. In this case as the coordinate is in the form of ratio which implies there is no unique way to represent an affine point with projective coordinates. This also leads to some loss in information. However, in projective coordinates, there is no requirement of performing inversions of points on elliptic curves while performing point doubling or point addition,

which helps to reduce the cost of the operations to some extent, as each inversion computation takes more time than multiplication or squaring operation.

A. Elliptic curve point addition and doubling for ECDH

In the cryptoscheme of ECDH, the short Weierstrass curve (Equation 1) is used. In this section, we provide the formula required to perform elliptic curve point addition and point doubling on this curve. Let the point getting doubled be $P(x_p, y_p)$ and after doubling $Q(x_q, y_q) = 2 \cdot P$. Then (x_q, y_q) can be expressed as follows [9]:

$$x_q = \lambda^2 - 2 \cdot x_p$$
$$y_q = \lambda \cdot (x_p - x_q) - y_p$$

where $\lambda = \frac{3 \cdot x_p^2 + a}{2 \cdot y_p}$. In case of point addition, let the two distinct input points be $P(x_p, y_p)$ and $Q(x_q, y_q)$. After addition let $R(x_r, y_r) = P + Q$. Then (x_r, y_r) can be computed as follows.

$$x_r = \lambda^2 - x_p - x_q$$

$$y_r = \lambda \cdot (x_p - x_r) - y_p$$

where $\lambda = \frac{y_q - y_p}{x_q - x_p}$. Next, we present the formulae for point doubling using standard projective coordinates [17] on the same curve. Now while using standard projective coordinates, the elliptic curve which was in the short Weierstrass form can be represented as

$$Y^2 \cdot Z = X^3 + a \cdot X \cdot Z^2 + b \cdot Z^3 \tag{3}$$

Let the point getting doubled be $P(X_P, Y_P, Z_P)$ and after doubling be $Q(X_Q, Y_Q, Z_Q) = 2 \cdot P$. So, (X_Q, Y_Q, Z_Q) can be written as follows [18] :

$$X_Q = 2 \cdot Y_P \cdot Z_P[9 \cdot ({X_P}^2 - {Z_P}^2)^2 - 8 \cdot X_P \cdot {Y_P}^2 \cdot Z_P]$$
$$Y_Q = 3 \cdot ({X_P}^2 - {Z_P}^2) \cdot [12 \cdot X_P \cdot {Y_P}^2 \cdot Z_P - 9 \cdot ({X_P}^2 - {Z_P}^2)^2] - 8 \cdot {Y_P}^4 \cdot {Z_P}^2$$
$$Z_Q = 8 \cdot {Y_P}^3 \cdot {Z_P}^3$$

In case of point addition using the same coordinates, let the points getting added be $P(X_P, Y_P, Z_P)$ and $Q(X_Q, Y_Q, Z_Q)$, after adding $R(X_R, Y_R, Z_R) = P + Q$. Therefore, (X_R, Y_R, Z_R) can be expressed as [18]:

$$U = Y_Q \cdot Z_P - Y_P \cdot Z_Q$$
$$V = X_Q \cdot Z_P - X_P \cdot Z_Q$$
$$A = U^2 \cdot Z_P \cdot Z_Q - V^3 - 2 \cdot V^2 \cdot X_P \cdot Z_Q$$
$$X_R = V \cdot A$$
$$Y_R = U \cdot (V^2 \cdot X_P \cdot Z_Q - A) - V^3 \cdot Y_P \cdot Z_Q$$
$$Z_R = V^3 \cdot Z_P \cdot Z_Q$$

B. Elliptic curve point addition and doubling for SIDH

The supersingular elliptic curve used in the SIDH protocol was defined in Section II-B. The formulae for point doubling on affine coordinates are given below. Let the point getting doubled be $P(x_p, y_p)$ and the after doubling $Q(x_q, y_q) = 2 \cdot P$. The coordinates (x_q, y_q) can be computed as [19] :

$$x_q = B \cdot \lambda^2 - A - 2 \cdot x_p$$
$$y_q = (3 \cdot x_p + A) \cdot \lambda - B \cdot \lambda^3 - y_p$$

where $\lambda = \frac{3 \cdot x_p^2 + 2 \cdot A \cdot x_p + 1}{2 \cdot B \cdot y_p}$. In case of point addition where the two distinct input points be $P(x_p, y_p)$ and $Q(x_q, y_q)$. After point addition let $R(x_r, y_r) = P + Q$, then (x_r, y_r) can be expressed as [19]:

$$x_r = B \cdot \lambda^2 - A - x_p - x_q$$
$$y_r = (2 \cdot x_p + x_q + A) \cdot \lambda - B \cdot \lambda^3 - y_p$$

where $\lambda = \frac{y_q - y_p}{x_q - x_p}$. Kummer variety of Montgomery curves were used with the logic of avoiding the computations involving y-coordinates. Therefore using projective coordinates, let the point getting doubled be $P(X_P, Z_P)$ and after doubling be $Q(X_Q, Z_Q) = 2 \cdot P$.

$$X_Q = C \cdot (X_P - Z_P)^2 \cdot (X_P + Z_P)^2$$
$$Y_Q = 4 \cdot X_P \cdot Z_P \cdot [A \cdot 4 \cdot X_P \cdot Z_P + C \cdot (X_P - Z_P)^2]$$

This point doubling algorithm also takes input of two constants A and C, corresponding to the curve being used. And the resultant coordinates of point Q can be represented as shown above [20] Let the points getting added be $P(X_P, Z_P)$ and $Q(X_Q, Z_Q)$, after adding $R(X_R, Z_R) = P + Q$. This point addition algorithm takes an extra input that is coordinates of the point $P - Q$ ($X_{P-Q} : 1$), where the z-coordinate is assumed to be 1. The resultant point R's coordinates can be computed as follows [20] :

$$X_R = [(X_P + Z_P) \cdot (X_Q - Z_Q) + (X_P - Z_P) \cdot (X_Q + Z_Q)]^2$$
$$Y_R = (X_{P-Q}) \cdot [(X_P + Z_P) \cdot (X_Q - Z_Q) - (X_P - Z_P) \cdot (X_Q + Z_Q)]$$

Based on the above given formulae, point addition and doubling is implemented on both the curves. The corresponding results on energy efficiency are presented below.

V. Effect of coordinate systems on power and energy consumptions

In this section we discuss the power and energy consumption of elliptic curve point additions and point doublings corresponding to the curves used in the schemes of ECDH and SIDH for both the security levels of 128 and 192. The notations of the curves used in this section are denoted in Section II.

A. On ordinary elliptic curves P-256 and P-384

As mentioned earlier, in case of ECDH an elliptic curve in the short Weierstrass form [21] is used, defined by Equation 1. Using "Double and Add" algorithm, any scalar point multiplication performed on points defined on elliptic curve is basically point addition and point doubling. Table II provides the comparison on number of operations such as multiplications, inversions and squarings, that are required on F_{p_n} for point doubling and point addition using the affine and projective coordinates on the ordinary elliptic curve, to be used

TABLE III: Power and energy consumption using affine and projective coordinates on the ECDH curves P-256 and P-384

Operations	128 bits				192 bits			
	Power(Watts)		Energy(mJoules)		Power(Watts)		Energy(mJoules)	
	Affine	Projective	Affine	Projective	Affine	Projective	Affine	Projective
Doubling	25.67	25.34	253.32	3.87	26.83	26.05	592.67	8.94
Adding	26.35	25.98	256.75	4.02	27.22	25.95	594.02	10.32

TABLE IV: Power and energy consumption using affine and projective coordinates on the SIDH curves C-503^2 and C-751^2

Operations	128 bits				192 bits			
	Power(Watts)		Energy(mJoules)		Power(Watts)		Energy(mJoules)	
	Affine	Projective	Affine	Projective	Affine	Projective	Affine	Projective
Doubling	26.98	25.93	270.03	5.8	27.13	25.05	621.53	11.32
Adding	26.76	26.31	268.31	5.3	27.02	26.75	609.61	12.53

in ECDH. Here, p_n is a prime of size either 256 bits or 384 bits, as recommended in [13].

TABLE II: Number of prime field operations used in affine and standard projective coordinates based point addition and doubling for the ECDH curves.

Instructions	Affine Coordinate		Projective Coordinate	
	Double	Add	Double	Add
Mul	2	2	7	12
Squaring	2	1	3	2
Inv	1	1	0	0

The number of multiplications refer to only the finite field multiplications involved. Multiplications with constants are ignored in this table. Inversions are computed on the prime field by using Fermat's Little Theorem. Since affine coordinates involve inversions, they are naturally slower, which gets reflected in their overall energy consumption as well (refer to Table III). It can be seen that affine coordinates requires marginally more power for all its computations when compared to projective coordinate based computations. In terms of energy, it requires on average around 60 to 65 times more energy than what projective coordinate based computations consume over the curves P-256 and P-384. These figures of energy consumption are mostly affected by the clock cycles required for the respective computations.

B. On supersingular elliptic curves C-503^2 and C-751^2

All the computations are performed on points over Montgomery curves as used in SIDH scheme (refer to Equation 2). Affine points on elliptic curves are represented as $P = (x, y)$ where each x and y coordinate is an element of $F_{p_s^2}$. Similarly projective Kummer coordinates [12] are represented as $(X : Z)$, where each of the coordinate is an element of $F_{p_s^2}$.

Let an arbitrary element M of $F_{p_s^2}$ be represented as $m_0 + i \cdot m_1$ where $m_0, m_1 \in F_{p_s}$ and $i^2 + 1 = 0$. Then the cost of adding two elements of $F_{p_s^2}$ is two additions/subtractions on F_{p_s}. Using the Karatsuba scheme [22], the multiplication of two elements of $F_{p_s^2}$ involves the following arithmetic operations on F_{p_s}: three multiplications, three additions (one of size $2|p_s|$ and the other two of size $|p_s|$ each, where $|x|$ is the bit length of x) and two subtractions of size $2|p_s|$. Squaring M would be requiring two multiplications, two additions (one of size $2|p_s|$ and the other of size $|p_s|$) and one subtraction of size $|p_s|$. Lastly, inverting M would need one inversion, two multiplications, two squarings and one addition of size $2|p_s|$. In other words, if the operations on the field $F_{p_s^2}$ are translated to the field F_{p_s} they turn out to be as shown below in Table V.

TABLE V: Number of prime field operations used in affine and projective coordinates based point addition and doubling for the SIDH curve

Instructions	Affine Coordinate		Projective Coordinate	
	Double	Add	Double	Add
Mul	15	10	10	13
Squaring	2	2	0	0
Inv	1	1	0	0

Since the comparisons are done with ECDH only point doubling and addition is focussed in this paper. However point tripling (not generally required in ECDH) is also an important and time consuming operation in SIDH, which was optimised in [23]. Table IV provides the power and energy consumed for these operations. Here also it can be seen that energy consumption in elliptic curve point operations using affine coordinates is around 45 to 50 times more compared to that using projective coordinates on the curves of C-503^2 and C-751^2. So, Table III and Table IV′s data further validates the energy efficiency of standard projective coordinates compared to affine coordinates.

VI. Energy consumption of ECDH and SIDH

As per the previous section's deductions, in order to achieve better energy efficiency, all the implementations of the algorithms in both the cryptosystems of SIDH and ECDH are done using standard projective coordinates. The basic elliptic curve point operations use the same formulae as mentioned in Section IV. Table VI compares the energy and power consumption between the crypto schemes. SIDH consumes slightly more power than ECDH for both the cases. However the energy consumption between them has a huge difference. The comparison has been done at the four major steps of key exchange that is initial key generation by both the parties, followed by the generation of shared key secret.

The aspect of energy consumption is dependent on both the power and time of execution. SIDH consumes a lot more energy than ECDH, where mostly the execution time corresponding to the individual steps in the schemes influence this difference. The isogeny evaluation and computation is one

TABLE VI: Comparison of Power (in Watts) and Energy (in Joules) consumption between ECDH and SIDH for 128 and 192 bit security levels

Cryptographic operations	128 bits				192 bits			
	ECDH		SIDH		ECDH		SIDH	
	Power	Energy	Power	Energy	Power	Energy	Power	Energy
Alice's Public Key	25.14	0.299	26.38	12.66	25.83	0.819	27.18	42.88
Bob's Public Key	25.08	0.275	26.46	11.12	26.12	0.821	26.96	38.36
Alice's Shared Secret	25.09	0.301	26.56	10.85	26.09	0.822	27.06	37.35
Bob's Shared Secret	25.13	0.287	26.33	9.15	25.96	0.808	27.13	35.45

of the most time consuming steps. Table VII provides the ratio of average power and energy used by SIDH cryptosystem to the ECDH. The comparison is done here on the basis of energy consumed by each operation in both ECDH and SIDH. An ECDH operation refers to a public key generation or a shared secret key generation at any of the node (Alice or Bob), involved in the secured communication. In case of ECDH, it is a point multiplication which involves a series of point doubling and point additions. ECDH operations at both the ends of Bob and Alice depend upon the private key which is being used for the point multiplication. Mostly, the time and energy consumed for such public key generation is similar at both the ends.

However in SIDH, when Alice and Bob compute their public keys, evaluation and computation of isogenies of different degrees are involved. In this particular implementation Alice computes isogeny using a kernel, which is a point of order 4 on the supersingular elliptic curve while Bob uses a kernel of order 3 on the same curve. As mentioned before the prime used in this scheme is of the form $p = {l_A}^{e_A} \cdot {l_B}^{e_B} - 1$. Therefore Alice evaluates isogenies of degree ${l_A}^{e_A}$ and Bob evaluates isogenies of degree ${l_B}^{e_B}$ respectively. So e_A isogenies of degree l_A and e_B isogenies of degree l_B are computed. The finite field operations involved in this isogeny computations of different degrees are also different. As a result unlike ECDH operations, Alice and Bob end up requiring different amounts of time and energy in their public key and shared secret key generation as shown in Table VI. On the field $F_{p_s^2}$ for the above mentioned isogeny computations [12], [15], Alice computes 13 multiplications and 8 squarings whereas Bob computes 9 multiplications and 5 squarings only.

VII. Battery exhaustion experiment

A practical experiment was performed to find out the number of SIDH and ECDH operations that can be performed on a laptop till its battery gets exhausted. We have used an HP Pavilion Notebook with the specification of the processor as given in Section III, with a battery capacity of 60 Watt hours or 216 KJoules. Since the public key generation and shared secret key generation in SIDH require different computations involving slightly different execution times, an average computation time is considered as each SIDH operation. On average, the number of ECDH operations that could be performed until the battery power of the laptop was exhausted is around 40 times more than that of SIDH. This finding is consistent with the Intel Power Gadget based results reported in the previous section. Also as the classical bit security level changes from 128 to 192, the number of operations in either ECDH or SIDH that could be performed decreases by around one third.

TABLE VII: Ratios of power and energy consumption for each SIDH operation compared to each ECDH operation

Bit Security	SIDH/ECDH	
	Power	Energy
128	1.05	37.67
192	1.07	47.10

VIII. Conclusion

In this paper, we have presented the energy efficiency of standard projective coordinate based elliptic curve computations compared to affine coordinates based similar computations. Standard projective coordinates have been found to be significantly more energy efficient than the affine coordinates. We have also reported energy consumption of SIDH and ECDH cryptoscheme implementation. Our results indicate that SIDH consumes about 40 times more energy than ECDH. Our findings also suggest that SIDH based on C-751[2] will consume three times more energy than the one based on C-503[2]. We should however note that, while SIDH is considered quantum-safe, ECDH is not. In addition to the relative energy (in)efficiency of SIDH compared to ECDH, we have reported their actual energy consumption values. To the best of our knowledge this is the first time that such values are reported for SIDH. These values could be an important consideration in the mode of deployment of SIDH in battery operated or energy constrained systems such as hand-held devices, remote sensors and space satellites.

Acknowledgements

This research was supported in part through an NSERC grant awarded to Dr. Hasan. A slightly different version of the manuscript will appear in the proceedings of the IEEE TrustCom 2018 conference.

References

[1] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of computation*, vol. 48, no. 177, pp. 203–209, 1987.

[2] V. S. Miller, "Use of elliptic curves in cryptography," in *Conference on the theory and application of cryptographic techniques*, pp. 417–426, Springer, 1985.

[3] J. W. Bos, J. A. Halderman, N. Heninger, J. Moore, M. Naehrig, and E. Wustrow, "Elliptic curve cryptography in practice," in *International Conference on Financial Cryptography and Data Security*, pp. 157–175, Springer, 2014.

[4] D. Stebila and J. Green, "Elliptic-curve algorithm integration in the secure shell transport layer," 2009.

[5] S. Blake-Wilson, B. Moeller, V. Gupta, C. Hawk, and N. Bolyard, "Elliptic curve cryptography (ECC) cipher suites for transport layer security (TLS)," 2006.

[6] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[7] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[8] D. Jao and L. De Feo, "Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies," in *International Workshop on Post-Quantum Cryptography*, pp. 19–34, Springer, 2011.

[9] D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to elliptic curve cryptography*. Springer Science & Business Media, 2006.

[10] J. H. Silverman, *The arithmetic of elliptic curves*, vol. 106. Springer Science & Business Media, 2009.

[11] P. G. Shah, "Investigating effects of co-ordinate system on execution time of elliptical curve protocol in wireless sensor networks," in *Future Communication Networks (ICFCN), 2012 International Conference on*, pp. 153–158, IEEE, 2012.

[12] C. Costello, P. Longa, and M. Naehrig, "Efficient algorithms for supersingular isogeny diffie-hellman," in *Annual Cryptology Conference*, pp. 572–601, Springer, 2016.

[13] P. Fips, "186-2. Digital Signature Standard (DSS)," *National Institute of Standards and Technology (NIST)*, vol. 20, p. 13, 2000.

[14] J. Vélu, "Isogénies entre courbes elliptiques," *CR Acad. Sc. Paris.*, vol. 273, pp. 238–241, 1971.

[15] D. Jao, R. Azarderakhsh, M. Campagna, C. Costello, A. Jalali, B. Koziel, B. LaMacchia, P. Longa, M. Naehrig, J. Renes, *et al.*, "Supersingular isogeny key encapsulation november 30, 2017," 2017.

[16] A. Menezes, F. A. Menezes, M. Qu, S. Vanstone, and K. J. Sutherland, "Elliptic curve systems," in *IEEE P1363, Part 4: Elliptic Curve Systems*, Citeseer, 1995.

[17] D. J. Bernstein and T. Lange, "Faster addition and doubling on elliptic curves," in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 29–50, Springer, 2007.

[18] H. Cohen, A. Miyaji, and T. Ono, "Efficient elliptic curve exponentiation using mixed coordinates," in *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 51–65, Springer, 1998.

[19] K. Okeya, H. Kurumatani, and K. Sakurai, "Elliptic curves with the montgomery-form and their cryptographic applications," in *International Workshop on Public Key Cryptography*, pp. 238–257, Springer, 2000.

[20] P. L. Montgomery, "Speeding the pollard and elliptic curve methods of factorization," *Mathematics of computation*, vol. 48, no. 177, pp. 243–264, 1987.

[21] W. Castryck, S. D. Galbraith, and R. R. Farashahi, "Efficient arithmetic on elliptic curves using a mixed edwards-montgomery representation.," *IACR Cryptology ePrint Archive*, vol. 2008, p. 218, 2008.

[22] A. Karatsuba and Y. Ofman, "Multiplication of many-digital numbers by automatic computers," in *Doklady Akad. Nauk SSSR*, vol. 145, p. 85, 1962.

[23] A. Faz-Hernández, J. López, E. Ochoa-Jiménez, and F. Rodríguez-Henríquez, "A faster software implementation of the supersingular isogeny diffie-hellman key exchange protocol," *IEEE Transactions on Computers*, 2017.