Hardware Deployment of Hybrid PQC: SIKE+ECDH

Reza Azarderakhsh(✉), Rami Elkhatib, Brian Koziel, and Brandon Langenberg

PQSecure Technologies LLC, Boca Raton, FL, USA
{razarder,rami.elkhatib,brian.koziel,brandon.langenberg}@pqsecurity.com

Abstract. In this work, we present a small architecture for quantum-safe hybrid key exchange targeting ECDH and SIKE. This is the first known hardware implementation of ECDH/SIKE-based hybrid key exchange in the literature. We propose new ECDH and EdDSA parameter sets defined over the SIKE primes. As a proof-of-concept, we evaluate SIKEX434, a hybrid PQC scheme composed of SIKEp434 and our proposed ECDH scheme X434 over a new, low-footprint architecture. Both schemes utilize the same 434-bit prime to save area. With only 1663 slices on a small Artix-7 device, our SIKE architecture can compute an entire hybrid key exchange in 320 ms. This is the smallest SIKE architecture in the literature. The hybrid SIKEX434 adds approximately 16% communication overhead and 10% latency overhead over SIKEp434. The additional overhead to support multiple primes indicates the need for new standardized ECC parameters for area-efficient designs in the future.

Keywords: Hybrid cryptosystem · Isogeny-based cryptography · Elliptic curve cryptography · Field-programmable gate array

1 Introduction

Quantum computers will bring about a new paradigm in computing technology, producing huge advances in various technological, medical, and financial sectors. While seen as a great boon for society, the major downside is that quantum computers can also be used maliciously to break the foundational security in the internet and various digital applications. Within hours, a large-scale quantum computer can break cryptosystems such as RSA and elliptic curve cryptography through Shor's algorithm [41]. Other cryptosystems such as AES can be weakened through Grover's algorithm [22], but not fully broken.

Given the uncertainty of when a large-scale quantum computer will emerge and the threats it poses, the National Institute of Standards and Technology (NIST) has solicited and evaluated classical and quantum-safe (post-quantum) candidates in its post-quantum cryptography (PQC) standardization project [8]. Now in its third round of review, there are still no clear winners, as a variety

J. Garcia-Alfaro et al. (Eds.): SecureComm 2021, LNICST 399, pp. 475–491, 2021.
https://doi.org/10.1007/978-3-030-90022-9_26

of tradeoffs in performance, communication overhead, security foundation, side-channel resistance, and so on continue to be evaluated. Although this standardization process was initiated in 2017, the winner has not been announced and no standards have been created. Historically, adoption of new cryptosystems is slow as standards must be created, new cryptosystems must be deployed, and an entire infrastructure slowly upgrades, perhaps even taking decades.

With the impending emergence of a large-scale quantum computer, hybrid key exchanges are seen as a solution to ease the transition to a quantum-safe infrastructure. A hybrid key exchange is a cryptosystem that uses multiple key exchange algorithms to reach a shared secret. The benefit here is that now an attacker must break each key exchange algorithm rather than simply one, greatly strengthening the key exchange. For instance, one hybrid key exchange could be pairing an extensively studied classically-safe key exchange with a quantum-safe alternative. Another example could be pairing two or more quantum-safe schemes. There are several classes of known quantum-safe key exchanges and signature schemes, but, unfortunately, there are still many gaps in quantifying their resistance to attacks and more research is still needed. Hybrid key exchanges allow us to hedge our risks and confidence in quantum-safe schemes.

Here, we investigate a hybrid key exchange composed of the classically-safe elliptic curve Diffie-Hellman (ECDH) and the quantum-safe supersingular isogeny key encapsulation (SIKE) [24] mechanism. ECDH and other elliptic curve schemes have been the de facto standard for public-key cryptography because of their small key sizes and competitive performance. Among quantum-safe key exchanges, there are many tradeoffs in terms of security confidence, performance, and communication overhead. However, SIKE is a NIST PQC Round 3 Alternative candidate based on isogenies of elliptic curves that features the smallest public key sizes but suffers from slow performance. By addressing shared computational similarities between ECDH and SIKE, we present a low-area hardware implementation of hybrid ECDH+SIKE.

Contributions:

- We propose new RFC 7748-style ECDH parameter sets over SIKE primes, including X434, X503, and X610.
- We propose new RFC 8032-style EdDSA parameter sets over SIKE primes, including Ed434, Ed503, Ed610, and Ed751.
- We propose, implement, and evaluate a small hybrid PQC architecture that performs SIKEX434, a hybrid PQC composed of SIKEp434 and X434. This is the smallest NIST Security Level 1 SIKE implementation in the literature.

2 Elliptic and Isogeny-Based Cryptography

Elliptic curve cryptography centers on the cryptographic applications of elliptic curves defined over finite fields. The primary applications in our digital society have been key establishment through the elliptic curve Diffie-Hellman (ECDH) key exchange and authentication through the elliptic curve digital signature

algorithm (ECDSA). We define an elliptic curve over a finite field as the set of all points (x, y) over the algebraic closure of $\mathbb{F}_q$ which statisfy the short Weierstrass curve equation below. This set, together with the point at infinity, is a group, while pairs in $\mathbb{F}_q \times \mathbb{F}_q$ form a subgroup. The short Weierstrass equation is:

$$E/\mathbb{F}_q \ : \ y^2 = x^3 + ax + b$$

where $a, b, x, y \in \mathbb{F}_q$. This set forms an abelian group over addition for which point addition and doubling can be defined. A series of point addition and doublings can compute an elliptic curve point multiplication, $Q = kP$ where $k \in \mathbb{Z}$ and $P, Q \in E$. ECDH and ECDSA are protected by the elliptic curve discrete logarithm problem (ECDLP), where given P, Q it is infeasible to find k as the size of the abelian group becomes large. We note that we have defined an elliptic curve here using the "traditional" short Weierstrass curve. New constructions of elliptic curves have shown additional benefits when used in an implementation. In the following sections, we choose to use Montgomery [39] curves for ECDH and twisted Edwards [3,16] curves for EdDSA (an ECDSA alternative) for better performance, implementation simplicity, and conformity to existing RFC standards. Elliptic curve cryptosystems defined over these elliptic curves are still protected by the ECDLP. However, the ECDLP is only hard for classical computers, whereas quantum computers can take advantage of Shor's algorithm [41] to efficiently factor the point multiplication and find k.

Isogeny-based cryptography has become a hot topic in the cryptographic community. In particular, we are focused on isogenies between elliptic curves. An isogeny can be thought of as a mapping between elliptic curves that preserves the point at infinity. Given elliptic curves E and E', it is difficult to find the isogeny ϕ such that $\phi \ : \ E \rightarrow E'$. This problem is still considered to be difficult for quantum computers.

Isogenies of elliptic curves were first proposed as a key-exchange over ordinary elliptic curves in 2006 in independent works by Couveignes [14] and Rostovtsev and Stolbunov [40]. Isogenies of supersingular elliptic curves were also proposed as a hash function in 2006 by Charles, Lauter, and Goren [7]. In 2010, Childs, Jao, and Soukharev [9] investigated the quantum resistance of isogenies over ordinary elliptic curves and found a new quantum subexponential time algorithm to compute isogenies. To counteract this quantum attack, Jao, De Feo, and Plût [15] proposed a new quantum-safe isogeny-based key exchange based on the difficulty to compute isogenies between *supersingular* elliptic curves, known as the supersingular isogeny Diffie-Hellman (SIDH) key exchange. Of known quantum-safe key exchange schemes, SIDH provided the smallest public key size at the cost of slower performance. Since the discovery of SIDH, the research of isogenies of elliptic curves in cryptography has continued to gain momentum, with new applications [20,25,44], security analyses [1,13,19,21,28,29,43], bandwidth reductions [2,11,12], and performance boosts [10,17,27,30–34,36].

Within the NIST PQC standardization project, the Supersingular Isogeny Key Encapsulation (SIKE) scheme was the only submitted isogeny-based cryptosystem. SIKE can be thought of as an upgrade to the SIDH key exchange, with

Table 1. Proposed ECDH parameter sets over SIKE primes

	SIKE			ECDH			
SIKE scheme	Security level	Public Key (Bytes)	Ciphertext (Bytes)	a coef	base x-coord	Pollard Rho attack (ops)	Public Key (Bytes)
$p434$	1	330	346	439,322	4	$2^{215.4}$	55
$p503$	2	378	402	308,290	2	$2^{249.8}$	63
$p610$	3	462	486	1,135,802	2	$2^{303.5}$	77
$p751$	5	564	596	624,450	3	$2^{374.2}$	94

better defense against active attacks and IND-CCA security. At this time, SIKE is currently in the third round of NIST PQC standardization as an alternative candidate. Among NIST PQC candidates, SIKE features by far the smallest public key sizes. The primary downsides are that its security problem is among the newest and that it is slow. Since SIKE's submission, the confidence in SIKE's hard problem has improved (with new and smaller parameter sets) and new implementations and optimizations continue to improve performance. SIKE is protected by the difficulty to compute isogenies between elliptic curves. Specifically, SIKE uses a slight variation of this problem over supersingular elliptic curves where a kernel defines the isogeny. Because of the breadth of theory in elliptic curve and isogeny-based cryptography, we point the reader to [18] for a more in-depth review of isogeny fundamentals.

3 Proposed Hybrid Key Exchange Based on Elliptic Curves

ECDH+SIKE. Our proposed hybrid key exchange is based on ECDH and SIKE which both relate to different hard problems based on elliptic curves defined over finite fields. Here, we define a Montgomery [39] curve over a finite field $\mathbb{F}_q$ as the collection of all points (x, y) and point at infinity that satisfy the Montgomery [39] form:

$$E/\mathbb{F}_q \ : \ by^2 = x^3 + ax^2 + x$$

SIKEp434. In the SIKE submission to the NIST PQC standardization project, there are 4 different parameter sets based on varying levels of security ranging from NIST Level 1 which is believed to be as hard as brute-forcing AES128 to NIST Level 5 which is believed to be as hard as brute-forcing AES256. For a low-area implementation, the NIST Level 1 parameters are the obvious pick. SIKEp434 is the parameter set, where all arithmetic is defined over a finite field $\mathbb{F}_q = \mathbb{F}_{p^2}$ with the 434-bit prime $p_{434} = 2^{216}3^{137} - 1$. The primary computations in SIKE include a double-point multiplication, large-degree isogeny, and the SHAKE256 hash. As a key encapsulation mechanism, SIKE includes three different phases: key generation, key encapsulation, and key decapsulation. Lastly, the starting elliptic curve is

$$E_0/\mathbb{F}_{p^2} \ : \ y^2 = x^3 + 6x^2 + x$$

New ECDH Schemes for SIKE Primes. In order to further reduce a hybrid PQC implementation's area footprint, we propose new parameter sets for ECDH, which we call X434, X503, and X610 that share the SIKE prime. The SIKE elliptic curves are specially crafted to be supersingular elliptic curves. These are necessary for supersingular isogeny-based cryptography, but are weak for classical elliptic curve schemes such as ECDH. To find appropriate curves for X434, X503, and X610, we searched for an elliptic curve with the smallest value $a_{24} = (a+2)/4$ such that the curve and its quadratic twist both have an order of the form four multiplied by a large prime (4·prime), similar to how Ed448 was found [23]. This a_{24} value is important for elliptic curve scalar multiplication over Montgomery curves. Our chosen curve over the $p434$ SIKE prime, called X434, has $a = 439322$ and $b = 1$, thus:

$$E/\mathbb{F}_p \ : \ y^2 = x^3 + 439322x^2 + x$$

The base point is at $x = 4$. This elliptic curve satisfies the Safecurves criteria, described on safecurves.cr.yp.to. For instance, the Pollard Rho attack would take approximately $2^{215.4}$ operations, which is close to the security strength of AES256. Since this curve was to be paired with the quantum-safe SIKE Level 1, we opted for a high-strength ECDH parameter set. We summarize each SIKE scheme and corresponding RFC 7748-style curve [35] in Table 1. Note that each curve has a Montgomery b coefficient of 1. We do not claim X751 as a contribution as it was originally found in [12].

Our proposed and implemented X434 key exchange scheme was adapted from the X25519 and X448 key exchanges described in RFC 7748 [35]. Specifically, a private scalar is created by choosing a random 55 byte value, setting bit 433, and clearing bits 0, 1, 434, 435, 436, 437, 438, and 439. These bits are set and cleared to ensure that a proper public key is generated. The key exchange is composed of 2 different phases: the key generation and the key agreement phases, each requiring an elliptic curve scalar point multiplication. Upon receiving a public key from another party for the key agreement phase, bits 434 through 439 are cleared. Lastly, public and shared keys are 55 bytes long.

SIKEX434 Operation. From here on, we call the hybrid key exchange with SIKEp434 and X434 as "SIKEX434". Since SIKEp434 is a key encapsulation mechanism and X434 is a Diffie-Hellman key exchange, we combine the two into a key encapsulation mechanism. In Fig. 1, we illustrate an example of a hybrid key exchange with SIKEX434. In this example, a client wants to establish a secure channel with a server using X434. First, the client generates a public key for X434 and SIKEp434. Then, in the Client Hello message, the client concatenates the two public keys. X434 and SIKEp434 public keys are 55 and 330 bytes, respectively, so a combined public key is 385 bytes, which is about 17% larger than simply a SIKEp434 public key. Upon receiving the Client Hello message, the server continues the key exchange by performing a public key generation

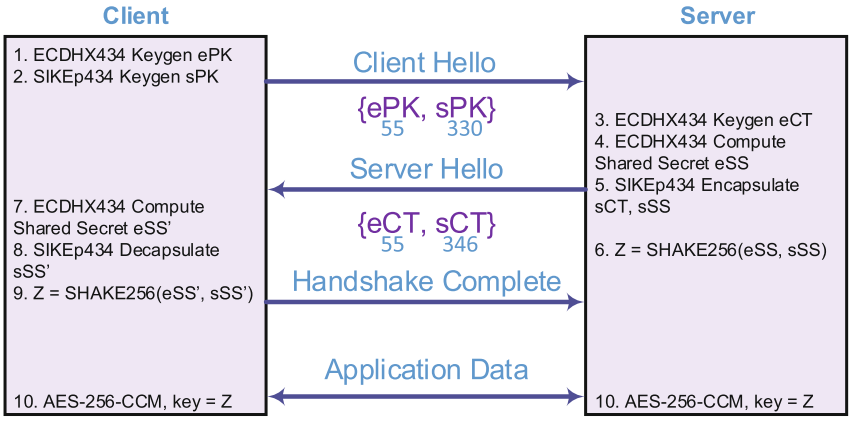

Fig. 1. Proposed SIKEX434 hybrid key exchange. 385 bytes are sent in Client Hello and 401 bytes are sent in Server Hello.

for X434, computing the X434 shared secret, and SIKEp434 encapsulating the SIKEp434 public key. For ease of notation, the output of X434 key generation on the server side is considered a ciphertext. The Server Hello is composed of the 55 byte X434 ciphertext and 346 byte SIKEp434 ciphertext, for a total 401 bytes. The Server can then use SHAKE256 (which is already included as part of SIKE) as a key derivation function by hashing the 55 byte X434 shared secret and 16 byte SIKEp434 shared secret. Upon receiving the Server Hello, the Client can proceed by computing the X434 shared secret and decapsulating the SIKEp434 ciphertext. Upon completion, the Client can similarly use the key derivation function to find the final shared key. If the hybrid key exchange was successful, both parties can use the shared key in AES to send and receive messages.

Key Derivation Methods. Since we are using two different public-key cryptosystems for SIKEX434, we must create a shared secret from the SIKEp434 shared key and X434 shared key. For this, we adapt the methods from [6] which combine the shared keys by concatenating or cascading the shared secrets before applying a pseudo-random function. For simplicity, we perform the concatenation key derivation function (CtKDF) combiner by concatenating the X434 shared key and SIKEp434 shared key and hashing them using the SHAKE256 function (which is already a part of SIKE). Further, if necessary, the concatenation KDF could include generated public keys, a context, and the length of the output as additional inputs in the SHAKE256 hash function. See [6] for further security analysis on generic hybrid key encapsulation methods.

The Case Against X25519 and X448. Here, we made the choice to go against Curve25519 and Curve448, two well-known curves for ECDH and EdDSA, to save area. X25519 uses a much smaller prime size and would have drastically sped up the classical cryptosystem used in a hybrid key exchange. Meanwhile, X448 uses a slightly larger, but better-shaped prime than X434, most likely resulting

Table 2. Proposed EdDSA parameter sets over SIKE primes

		SIKE			EdDSA		
SIKE scheme	SIKE/EdDSA prime	Security level (Bytes)	Public Key (Bytes)	Ciphertext (Bytes)	d coef	Public Key (Bytes)	Signature (Bytes)
$p434$	$2^{216}3^{137}-1$	1	330	346	109,831	55	110
$p503$	$2^{250}3^{159}-1$	2	378	402	77,073	63	126
$p610$	$2^{305}3^{192}-1$	3	462	486	283,951	77	144
$p751$	$2^{372}3^{239}-1$	5	564	596	156,113	94	188

in a similar timing profile. However, because SIKE is limited (for reasonable performance) to only smooth isogeny-friendly primes such as SIKEp434 we cannot use these standardized ECDH primes. Although we do not have results (at this time), pairing X25519 or X448 would mean that our modular multiplier as we describe in Sect. 5 would be much larger. For instance, consider taking X25519 with the Montgomery multiplier we describe in the following section. X25519's prime was specifically chosen for its pseudo-Mersenne prime shape that cannot be taken advantage of by SIKE primes. For Montgomery multiplication, this would result in more arithmetic logic (small multipliers) to compute the Montgomery q constant each computation. Furthermore, additional registers and control logic would be needed to switch between the X25519 multiplication reduction and the SIKEp434 multiplication reduction. Lastly, the latency of SIKE is significantly longer than ECDH, so including X25519 would only have a tiny improvement in total latency of a hybrid key exchange.

4 Proposed EdDSA Schemes over SIKE Parameters

In addition to ECDH for a hybrid key establishment, digital signatures are another strong use case for elliptic curves. Although elliptic curve-based digital signatures would also be broken by a sufficiently large quantum computer, the use cases and security longevity between key establishment and authentication are different. For instance, quantum-safe key establishment must be implemented and deployed well in advance of large-scale quantum computers as retroactive decryptions could break past secure communications. The concern here is that highly confidential communications including top-secret classified information that is encrypted with classical cryptosystems could be broken well before it was supposed to be declassified. One primary application of digital signatures is authentication. Retroactive authentication is not so much an issue in our digital infrastructure as authenticating a session is typically short-term. When large quantum computers are available, implementations using classical-safe digital signature schemes can make the switch to quantum-safe digital signature schemes and carry on business as usual.

Here, we see elliptic curve-base digital signatures as an important complement to hybrid PQC schemes. Similar to our parameter selection for RFC 7748-style [35] ECDH parameter sets, we can also create new parameter sets for RFC 8032-style [26] Edwards Digital Signature Algorithm (EdDSA). We chose to go

with EdDSA over ECDSA as EdDSA features improved performance, implementation simplicity, and no need to generate a nonce for each signature. Up until now, Ed25519 [4] and Ed448 [23] were the primary EdDSA curves. Although it uses a curve defined over a smaller prime, Ed25519 is less ideal for a small footprint implementation with SIKE as it uses SHA-512 whereas Ed448 uses SHAKE256. SHAKE256, used in all SIKE parameter sets, is based on the Keccak [5] sponge family of hashes and is standardized in SHA-3.

New EdDSA Schemes for SIKE Primes. Here, we propose new parameter sets for EdDSA, which we call Ed434, Ed503, Ed610, and Ed751 that share the primes used of the corresponding SIKE versions SIKEp434, SIKEp503, SIKEp610, and SIKEp751, respectively. All EdDSA schemes are defined over twisted Edwards curves of the following form:

$$E/\mathbb{F}_q \ : \ ax^2 + y^2 = 1 + dx^2y^2$$

To find appropriate curves for Ed434, Ed503, Ed610, and Ed751, we fixed $a = 1$ and searched for the smallest positive d value such that both the curve and its quadratic twist have 4·prime order and d was not a square in $\mathbb{F}_q$. If d is not a square, then the Edwards curve arithmetic is complete. We summarize our found curves in Table 2.

Another method to search for the Edward curve parameters could have been to apply an isomorphism from the Montgomery curves, similar to how Ed25519 was created as an isomorphism from X25519. For instance, applying a similar isomorphism from X434 to Ed434 would have created the curve, $E/\mathbb{F}_p \ : \ x^2+y^2 = 1+(109830/109831)x^2y^2$. Furthermore, the birational equivalence is further that $x_{Ed} = \sqrt{439324}x_{Mont}y_{Mont}$ and $y_{Ed} = (x_{Mont} - 1)(x_{Mont} + 1)$. The isomorphic base point is then $(x, \frac{3}{5})$, for which x is positive. We opted to find smaller twisted Edwards curves to save space when storing the curve and base point. Also, interestingly enough, the Edwards d coefficients of the EdYYY schemes are equal to the Montgomery a_{24} for each of the XYYYschemes. For instance, Ed434 uses a d coefficient of 109,831, which is equal to X434's $a_{24} = (439322+2)/4 = 109831$.

Ed434. Here, we describe the operation for Ed434, an RFC 8032-style [26] EdDSA scheme over the SIKE prime $p434$. This is depicted in Fig. 2. The EdDSA schemes over other SIKE primes are extremely similar. Ed434 is the collection of all points (x, y) that satisfy the twisted Edwards elliptic curve form (note that $a = 1$) over the finite field $p434 = 2^{216}3^{137} - 1$:

$$E/\mathbb{F}_p \ : \ x^2 + y^2 = 1 + 109831x^2y^2$$

Ed434 Key Generation. The Ed434 public key is generated by first collecting 55 bytes of random data. These 55 bytes are hashed using SHAKE256 and the output digest is the first 110 bytes. The lower 55 bytes of this output digest are pruned to an output s by setting bit 433, and clearing bits 0, 1, 434, 435, 436, 437, 438, and 439. This result is interpreted as a scalar and used to perform a scalar point multiplication over the base point $A = [s]B$. Finally, this point

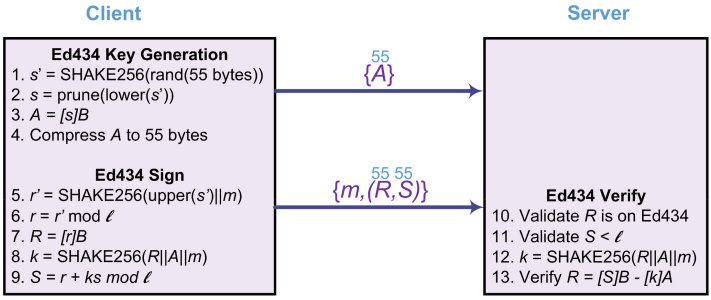

Fig. 2. Proposed Ed434 digital signature scheme.

result is compressed by encoding the first 55 bytes with the y-coordinate and the least significant bit of the x-coordinate is copied to the most significant bit of the final byte, resulting in a 55 byte public key.

Ed434 Sign. A message is signed by first SHAKE256 hashing the upper 55 bytes of the secret key hash with the message to a 110-byte r. r is reduced modulo the large prime order of the curve, ℓ, and interpreted as a scalar to perform a scalar point multiplication over the base point $R = [r]B$. Point R is represented in the compressed Edwards form as was done with the public key A. SHAKE256 is once again used to hash R, A, and the message to a 110-byte digest k. Lastly, the signature component S is computed by performing $S = r + k \cdot s \bmod \ell$. The final signature is the compressed Edwards point R and the scalar S, totaling to 110 bytes.

Ed434 Verify. A message and signature is verifed by first validating the point R and scalar S are of the correct form. This entails checking that R decodes to a valid Ed434 point and that the scalar S is less than the large prime order of Ed434. Note that decoding an Ed434 Edwards point involves a few exponentiations in a similar manner to Ed448, but no square roots are needed (see Sect. 5.2.3 of RFC 8032 [26]). Upon validating signature components R and S, SHAKE256 is once again used to hash R, A, and the message to a 110-byte digest k. Lastly, this k value is used to verify the group equation $[S]B = R + [k]A$. If the left and right side are equal, then the signature is valid. Note that similar to [4], one can instead check that the encodings of the left and right hand side of $R = [S]B - [k]A$ match to avoid decoding point R.

5 ECC/SIKE for Small Devices

Here, we present some details and results of our small footprint architecture to accelerate SIKE and ECDH over the 434-bit prime. Our core components and area numbers are shown in Fig. 3.

Field Addition. At the lowest level, ECDH and SIKE are composed of modular addition and multiplication over p_{434}. One primary reason to create the parameter set for X434 was so that the field arithmetic unit could be reused between

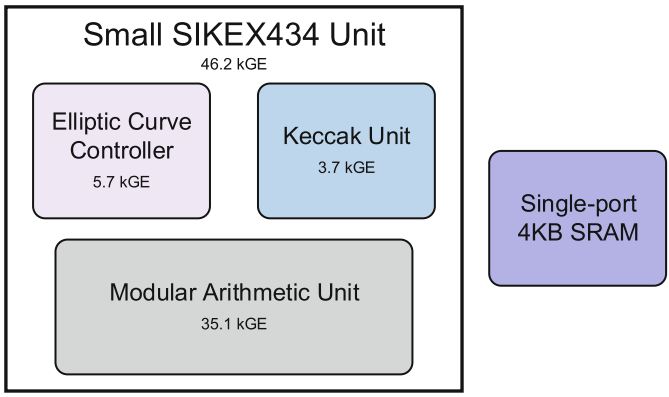

Fig. 3. High-level components of the small SIKEX434 unit. The numbers below the component indicate its size in Gate Equivalents (GE).

the two key exchanges. To keep interface and arithmetic overhead low, all data in our architecture is stored by 32-bit word in a single-port 4KB SRAM. Each intermediate $\mathbb{F}_p$ value requires 55 bytes or 14 words. For a given cycle we can only perform one read or write operation of 32-bit data. $\mathbb{F}_p$ addition performs $c = a + b$ where $a, b, c \in \mathbb{F}_p$. Thus, if $a + b > p$, then we perform a reduction and the result is $c = a + b - p$. $\mathbb{F}_p$ subtraction is similar, but a reduction is performed by adding p if the result is negative. For addition and subtraction, we first load one full operand, perform the addition or subtraction, perform a reduction, and store the result. Our architecture can internally store two full $p434$ registers, so we store the addition/subtraction result in register 1 and store the conditional reduction result in register 2. We choose the correct register to store back in our SRAM by checking the negative bit of operand 2 for $\mathbb{F}_p$ addition and the negative bit of operand 1 for $\mathbb{F}_p$ subtraction. An $\mathbb{F}_p$ addition/subtraction requires 48 cycles.

Field Multiplication. The modular multiplier is the largest piece of this architecture. It is an intense computation which is used extensively for both SIKEp434 and X434. Since the multiplication of two similarly sized values produces a result double the size, an expensive reduction is also needed. Unfortunately, efficient modular multiplication with $p434$ is limited to Barrett or Montgomery multiplication algorithms. Our modular multiplier is adapted from the carry-save adder Montgomery multiplier from [42] with a digit size of 4-bits. Montgomery multiplication [38] converts expensive division operations into shift operations, which are extremely cheap in hardware. In each cycle, this systolic modular multiplier computes 4 bits of a modular multiplication result. Since the final result is in two different buffers, one for sum and one for carry, we finish the computation by adding these two buffers with our field addition unit. By limiting this modular multiplier to only support $p434$, we can optimize away many gates in this multiplier architecture, such as any AND gate connected to the modulus regis-

ters. For $p434$, this is 434 AND gates. If multiple primes were to be supported, then there would have to be some storage to swap between primes, these AND gates would have been included, and there would be a larger critical path. An $\mathbb{F}_p$ multiplication requires 158 cycles.

Controller. The controller handles all sequencing necessary for ECDH point multiplications and SIKE isogenies. The controller reads from a program ROM that contains all elliptic curve-related subroutines. Each subroutine is composed of a sequence of $\mathbb{F}_p$ addition, subtraction, and multiplication, and $\mathbb{F}_{p^2}$ addition, subtraction, multiplication, and squaring. There are 1058 instructions total in our program ROM, for which the Fermat's Little Theorem-based $\mathbb{F}_p$ inversion required almost 600. ECDH used the classical Montgomery ladder. The isogeny and point arithmetic required for SIKE are the fastest known formulas in the literature that can be found in [24]. The large-degree isogeny requires a number of pivot points for efficient computation and our architecture can store up to 6 pivot points.

Keccak Unit. SIKE utilizes SHAKE256 as a hash function throughout its operation. However, the latency of SIKE's isogeny arithmetic dominates the computations, so we opted for an extremely small SHAKE256 module. SHAKE256 is based on the Keccak sponge family of hashes. Most fast implementations utilize 1600 registers to represent the full Keccak internal state. Our implementation simply reuses the single-port 4KB SRAM to hold the 1600-bit state. Our implementation utilizes various shifts and arithmetic functions as part of the Keccak sponge family, but performed over a single word at a time. Depending on the size of a SHAKE256 hash, generally each permutation operation takes slightly over 220,000 cycles, which is still significantly less than a SIKE operation.

Verification. SIKE and ECDH were tested separately and then put together for hybrid key exchange. SIKE was tested against the SIKE submission's Known Answer Tests and ECDH was tested against python-generated key generation and key agreement test vectors. Our hybrid key exchange generated a shared secret by using the concatenation key derivation function method from [6]. Essentially, a shared secret was produced by hashing a concatenation of the SIKEp434 and X434 shared secrets with SHAKE256 and using the first 256 bits of the result.

6 Hybrid Architecture Results

Here, we describe our small SIKEX434 hardware implementation results. For FPGA results, we used the NIST PQC recommended Artix-7 FPGA. We synthesized the SIKEX434 core with Xilinx Vivado 2019.2 to a Xilinx Artix-7 xc7a200tffbg676-2 device. On the ASIC side, we synthesized using Synopsis Design Compiler Q-2019.12 with the TSMC 65nm library tcbn65lptc. All results were obtained after place-and-route.

Table 3. FPGA results of small SIKEX434 accelerator on a Xilinx Artix-7.

Frequency	Area				
MHz	# FFs	#LUTs	# Slices	# DSPs	#BRAMs
195	1,942	5,841	1,663	0	1

Table 4. ASIC results of small SIKEX434 accelerator on 65nm tcbn65lptc. Note that this excludes the single-port 4KB SRAM.

Frequency	Critical path	Area	
MHz	ns	um^2	GE
278	6.41	66,578	46,235

Area. We present our FPGA results in Table 3 and ASIC results in Table 4. On the FPGA side, we used a total of 1663 slices for our entire design and one 36k BRAM. This equated to 1942 flip-flops and 5841 look-up tables. The maximum frequency for this configuration was 195 MHz. For the ASIC, we synthesized at 100 MHz, which the Synopsys Compiler easily beat, giving a slack of 6.41 ns. The critical path is thus 3.59 ns, which corresponds to a frequency of 278 MHz. The total area of the design was 66,578 um^2 which we converted to gate equivalents (GE) by dividing by 1.44, which was the size of a NAND gate in um^2. Note that this excludes the single-port 4KB SRAM. We further break down the size of the 46.2 kGE in Fig. 3. The finite field accelerator took the majority of the design at 35.1 kGE. This area could continue to go down notably by reducing the digit size of the systolic multiplier from 4 down to 1 or 2 at the expense of almost quadrupling or doubling the resulting time, respectively.

Timing. The latencies between operations are identical between the two device targets, which are presented in Table 5. The "E + D" column includes the key encapsulation and decapsulation time as was done in the SIKE submission. Since key generation only needs to be performed a single time by a party, it is expected that the encapsulation and decapsulation timings are the customer-felt latency. As the timings show, the SIKEX434 E + D time for ASIC is 175.2 ms and the time for an entire SIKEX434 hybrid key exchange is 224.1 ms. Note that given more aggressive timing for the synthesis that these numbers can continue to improve at the cost of increased area. However, higher frequencies greatly impact the resulting power and energy consumption.

SIKEX434 Overhead. Comparing SIKEp434 to SIKEX434 shows a 9.5% increase of latency for the additional hybrid computations. These hybrid computations include four additional X434 operations as well as two additional SHAKE256 KDF operations. On the communication side, SIKEX434 requires 16.3% additional overhead. A timing breakdown of each cryptosystem's primitives are presented in Fig. 4.

Table 5. Latency of operations on our SIKEX434 accelerator. Note that the latency is identical for Artix-7 and ASIC.

Scheme	Latency	Time (ms)	
	(cc×10^6)	Artix-7	ASIC
		@195 MHz	@278 MHz
X434 Keygen	1.2	6.2	4.3
X434 KeyAgree	1.2	6.2	4.3
SIKEp434 Keygen	12.4	63.6	44.6
SIKEp434 Encap	21.4	109.8	77.0
SIKEp434 Decap	23.1	118.5	83.1
SIKEp434 E + D	44.5	228.2	160.1
SIKEp434 Total	56.9	291.8	204.7
SIKEX434 Keygen	13.6	69.7	48.9
SIKEX434 Encap	24.1	123.6	86.7
SIKEX434 Decap	24.6	126.2	88.5
SIKEX434 E + D	48.7	249.7	175.2
SIKEX434 Total	62.3	319.5	224.1

Table 6. Area comparison of isogeny architectures on a Artix-7 at approximately NIST security level 1 (SIKEp434).

Work	# FFs	# LUTs	# Slices	# DSPs	#BRAMs
Massolino *et al.* [37] (128)[1,2]	7,202	11,943	3,491	57	21
Massolino *et al.* [37] (256)[1,2]	11,661	22,673	7,329	162	37
Koziel *et al.* [27]	24,328	21,946	8,006	240	26.5
This work	**1,942**	**5,841**	**1,663**	**0**	**1**

1. Implementation also includes SIKEp503, SIKEp610, and SIKEp751
2. (128) and (256) refer to 128-bit and 256-bit multipliers, respectively

Comparison. A fair comparison with other implementations is difficult as their focus has generally been on performance. Since our focus is on NIST Security Level 1, we summarize Artix-7 area and timing comparison results in Tables 6 and 7, respectively. The SIKE submission's hardware implementation is based on the high performance implementation from [27]. For SIKEp434, this implementation requires 8006 slices, 240 DSPs, and 26.5 BRAMs to compute SIKEp434 in 14.4 ms on the Artix-7. This is 16 times faster than our implementation for 5 times as many slices and significantly more BRAMs and DSPs. The more area-efficient implementation from [37] supports all four SIKE parameter sets using a hardware software co-design methodology. Their smaller implementation implements SIKEp434 in just over 50 ms with 3415 slices, 57 DSPs, and 21 BRAMs. This is over 4 times faster at the cost of 2 times as many slices and again significantly more BRAMs and DSPs. Unfortunately, this disparity in BRAMs and

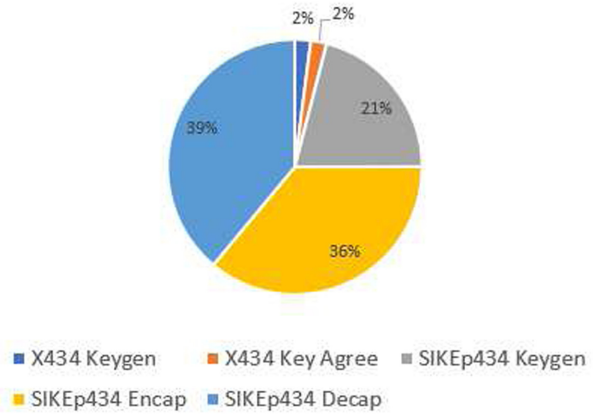

Fig. 4. Timing breakdown of SIKEp434 and X434 primitives. Note that X434 operations are performed twice in a full SIKEX434 operation.

Table 7. Timing comparison of isogeny architectures on a Artix-7 at approximately NIST security level 1 (SIKEp434). Note that SIKE total time includes key encapsulation and decapsulation.

Work	Freq. (MHz)	Cycles ($cc \times 10^6$)	Total time (ms)
Massolino *et al.* [37] (128)	145.1	27.3	52.8
Massolino *et al.* [37] (256)	109.1	8.6	31.7
Koziel *et al.* [27]	132.2	1.91	14.4
This work	**163.1**	**44.5**	**228.2**

DSPs make a fair area-time comparison difficult. Hopefully there will be more ASIC area results for SIKE implementations in the future for more accurate comparisons of resource cost. Nonetheless, this implementation shows a lower area bound for SIKEp434 and SIKEX434 hardware implementations.

7 Conclusion

In this work, we proposed the hybrid key exchange SIKEX434 composed of SIKEp434 and X434 as well as new ECDH and EdDSA schemes over SIKE parameters SIKEp434, SIKEp503, SIKEp610, and SIKEp751. We presented the smallest SIKE architecture in the literature for this design that is 1663 slices in FPGA and 46 kGE and 4KB SRAM in ASIC. This design can accomplish the entire SIKEX434 on ASIC in less than 250 ms. An expected segment of a key exchange will also take less than a tenth of a second. SIKEX434 adds approximately 16% communication overhead and 10% latency overhead. With a 385 byte public key and 401 byte ciphertext, this hybrid key exchange is still 50% smaller than the next smallest key in the NIST PQC contest. As research

continues into quantum computer attacks on cryptosystems, adapting and innovating hybrid key exchange schemes will be necessary as we transition to a fully quantum-safe infrastructure.

Acknowledgment. This work has been funded by a DoD contract W911NF2010328 granted to PQSecure Technologies.

Intellectual Property Disclosure. Some of these techniques may be covered by US and/or international patents.

References

1. Adj, G., Cervantes-Vázquez, D., Chi-Domínguez, J.J., Menezes, A., Rodríguez-Henríquez, F.: On the cost of computing isogenies between supersingular elliptic curves. In: Cid, C., Jacobson Jr., M.J. (eds.) Selected Areas in Cryptography - SAC 2018–25th International Conference, Calgary, AB, Canada, August 15–17, 2018, Revised Selected Papers, Lecture Notes in Computer Science, vol. 11349, pp. 322–343. Springer, Heidelberg (2018). https://doi.org/10.1007/978-3-030-10970-7_15
2. Azarderakhsh, R., Jao, D., Kalach, K., Koziel, B., Leonardi, C.: Key compression for isogeny-based cryptosystems. In: Proceedings of the 3rd ACM International Workshop on ASIA Public-Key Cryptography, pp. 1–10 (2016)
3. Bernstein, D.J., Birkner, P., Joye, M., Lange, T., Peters, C.: Twisted edwards curves. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 389–405. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68164-9_26
4. Bernstein, D.J., Duif, N., Lange, T., Schwabe, P., Yang, B.-Y.: High-speed high-security signatures. J. Cryptogr. Eng. **2**(2), 77–89 (2012)
5. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G., Van Keer, R.: Keccak Implementation Overview (2012)
6. Campagna, M., Petcher, A.: Security of hybrid key encapsulation. Cryptology ePrint Archive, Report 2020/1364 (2020). https://eprint.iacr.org/2020/1364
7. Charles, D.X., Lauter, K.E., Goren, E.Z.: Cryptographic hash functions from expander graphs. J. Cryptol. **22**(1), 93–113 (2007). https://doi.org/10.1007/s00145-007-9002-x
8. Chen, L., et al.: Report on Post-Quantum Cryptography. NIST IR 8105 (2016)
9. Childs, A.M., Jao, D., Soukharev, V.: Constructing elliptic curve isogenies in quantum subexponential time. J. Math. Cryptol. **8**(1), 1–29 (2014)
10. Costello, C., Hisil, H.: A simple and compact algorithm for SIDH with arbitrary degree isogenies. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10625, pp. 303–329. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70697-9_11
11. Costello, C., Jao, D., Longa, P., Naehrig, M., Renes, J., Urbanik, D.: Efficient compression of SIDH public keys. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10210, pp. 679–706. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56620-7_24
12. Costello, C., Longa, P., Naehrig, M.: Efficient algorithms for supersingular isogeny diffie-hellman. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9814, pp. 572–601. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53018-4_21

13. Costello, C., Longa, P., Naehrig, M., Renes, J., Virdia, F.: Improved classical cryptanalysis of SIKE in practice. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020. LNCS, vol. 12111, pp. 505–534. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45388-6_18
14. Couveignes, J.M.: Hard Homogeneous Spaces. Cryptology ePrint Archive, Report 2006/291 (2006)
15. De Feo, L., Jao, D., Plût, J.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. J. Math. Cryptol. **8**(3), 209–247 (2014)
16. Edwards, H.M.: A normal form for elliptic curves. Bull. Am. Math. Soc. **44**, 393–422 (2007)
17. Faz-Hernández, A., López, J., Ochoa-Jiménez, E., Rodríguez-Henríquez, F.: A faster software implementation of the supersingular isogeny diffie-hellman key exchange protocol. IEEE Trans. Comput. **67**(11), 1622–1636 (2018)
18. Feo, L.D.: Mathematics of Isogeny Based Cryptography. CoRR, abs/1711.04062 (2017)
19. Galbraith, S.D., Petit, C., Shani, B., Ti, Y.B.: On the security of supersingular isogeny cryptosystems. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10031, pp. 63–91. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53887-6_3
20. Galbraith, S.D., Petit, C., Silva, J.: Identification protocols and signature schemes based on supersingular isogeny problems. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10624, pp. 3–33. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70694-8_1
21. Gélin, A., Wesolowski, B.: Loop-abort faults on supersingular isogeny cryptosystems. In: Lange, T., Takagi, T. (eds.) PQCrypto 2017. LNCS, vol. 10346, pp. 93–106. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59879-6_6
22. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, STOC 1996, pages 212–219, New York, NY, USA. Association for Computing Machinery (1996)
23. Hamburg, M.: Ed448-goldilocks, a new elliptic curve. Cryptology ePrint Archive, Report 2015/625 (2015). https://eprint.iacr.org/2015/625
24. Jao, D., et al.: Supersingular isogeny key encapsulation. Submission to the NIST Post-Quantum Standardization Project (2017)
25. Jao, D., Soukharev, V.: Isogeny-based quantum-resistant undeniable signatures. In: Mosca, M. (ed.) PQCrypto 2014. LNCS, vol. 8772, pp. 160–179. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11659-4_10
26. Josefsson, S., Liusvaara, I.: Edwards-curve digital signature algorithm (eddsa). RFC **8032**, 1–60 (2017)
27. Koziel, B., Ackie, A.B., Khatib, R.E., Azarderakhsh, R., Kermani, M.M.: Sike'd up: fast hardware architectures for supersingular isogeny key encapsulation. IEEE Trans. Circ. Syst. I Reg. Papers **67**(12), 4842–4854 (2020)
28. Koziel, B., Azarderakhsh, R., Jao, D.: An exposure model for supersingular isogeny diffie-hellman key exchange. In: Smart, N.P. (ed.) CT-RSA 2018. LNCS, vol. 10808, pp. 452–469. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-76953-0_24
29. Koziel, B., Azarderakhsh, R., Jao, D.: Side-channel attacks on quantum-resistant supersingular isogeny Diffie-Hellman. In: Adams, C., Camenisch, J. (eds.) SAC 2017. LNCS, vol. 10719, pp. 64–81. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-72565-9_4

30. Koziel, B., Azarderakhsh, R., Jao, D., Mozaffari-Kermani, M.: On fast calculation of addition chains for isogeny-based cryptography. In: Chen, K., Lin, D., Yung, M. (eds.) Inscrypt 2016. LNCS, vol. 10143, pp. 323–342. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-54705-3_20
31. Koziel, B., Azarderakhsh, R., Mozaffari-Kermani, M.: Fast hardware architectures for supersingular isogeny diffie-hellman key exchange on FPGA. In: Dunkelman, O., Sanadhya, S.K. (eds.) INDOCRYPT 2016. LNCS, vol. 10095, pp. 191–206. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-49890-4_11
32. Koziel, B., Azarderakhsh, R., Mozaffari-Kermani, M.: A high-performance and scalable hardware architecture for isogeny-based cryptography. IEEE Trans. Comput. **67**(11), 1594–1609 (2018)
33. Koziel, B., Azarderakhsh, R., Mozaffari-Kermani, M., Jao, D.: Post-quantum cryptography on FPGA based on isogenies on elliptic curves. IEEE Trans. Circ. Syst. I Reg. Papers **64**(1), 86–99 (2017)
34. Koziel, B., Jalali, A., Azarderakhsh, R., Jao, D., Mozaffari-Kermani, M.: NEON-SIDH: efficient implementation of supersingular isogeny diffie-hellman key exchange protocol on ARM. In: Foresti, S., Persiano, G. (eds.) CANS 2016. LNCS, vol. 10052, pp. 88–103. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48965-0_6
35. Langley, A., Hamburg, M., Turner, S.: Elliptic curves for security. RFC **7748**, 1–22 (2016)
36. Liu, W., Ni, J., Liu, Z., Liu, C., O'Neill, M.: Optimized modular multiplication for supersingular isogeny diffie-hellman. IEEE Trans. Comput. **68**(8), 1249–1255 (2019)
37. Massolino, P.M.C., Longa, P., Renes, J., Batina, L.: A compact and scalable hardware/software co-design of SIKE. Cryptology ePrint Archive, Report 2020/040 (2020). https://eprint.iacr.org/2020/040
38. Montgomery, P.L.: Modular multiplication without trial division. Math. Comput. **44**(170), 519–521 (1985)
39. Montgomery, P.L.: Speeding the pollard and elliptic curve methods of factorization. In: Mathematics of Computation, pp. 243–264 (1987)
40. Rostovtsev, A., Stolbunov, A.: Public-key cryptosystem based on isogenies. Cryptology ePrint Archive, Report 2006/145 (2006)
41. Shor, P.W.: Algorithms for quantum computation: discrete logarithms and factoring. In: 35th Annual Symposium on Foundations of Computer Science (FOCS 1994), pp. 124–134 (1994)
42. Sutter, G., Deschamps, J.P., Imaña, J.L.: Modular multiplication and exponentiation architectures for fast RSA cryptosystem based on digit serial computation. IEEE Trans. Ind. Electroni. **58**(7), 3101–3109 (2011)
43. Ti, Y.B.: Fault attack on supersingular isogeny cryptosystems. In: Lange, T., Takagi, T. (eds.) PQCrypto 2017. LNCS, vol. 10346, pp. 107–122. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59879-6_7
44. Yoo, Y., Azarderakhsh, R., Jalali, A., Jao, D., Soukharev, V.: A post-quantum digital signature scheme based on supersingular isogenies. In: Kiayias, A. (ed.) FC 2017. LNCS, vol. 10322, pp. 163–181. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70972-7_9