Title Goes Here

Submitted by

Robert W.V. Gorrie
B.ASc. Computer Science (McMaster University)

Under the guidance of
Douglas Stebila

Submitted in partial fulfillment of
the requirements for the award of the degree of

Masters of Science
in
Computer Science

Department of Computing and Software
McMaster University
Hamilton, Ontario, Canada

Fall Semester 2017

Abstract

¡Abstract here¿

Contents

List of Figures

Chapter 1

Introduction

1.1 Background and Recent Research

1.1.1 ¡any sub section here¿

1.1.2 Literature Survey

1.2 Layout of Paper

¡Sub-subsection title¿

some text[1], some more text

¡Sub-subsection title¿

even more text[1], and even more.

1.3 Motivation

[1] ¡footnote here¿

Chapter 2

Technical Background

Over the course of the past decade, elliptic curve cryptography (ECC) has proven itself a mainstay in the wide world of applied cryptology. While isogeny based cryptography does build itself up from the same underlying field of mathematics as ECC, it simultaneously draws from a slightly more complicated space of algebraic notions. Much of this chapter will be dedicated to illuminating these notions in a manner that should be digestable for those without serious background in algebraic geometry, or abstract algebra in general.

This chapter will cover the following preliminary topics: isogenies and their relevant properties, supersingular isogeny Diffie-Hellman (SIDH) and related procotols, the Fiat-Shamir construction for digital signatures (and its quantum-safe adaptation), and finally the current landscape of isogeny based signature schemes.

Our discussion of isogenies will begin with some basic coverage of the underlying algebra. We will provide the material necessary for the remaining sections as we build up in the level of abstraction; working our way through groups, finite fields, elliptic curves, and finally isogenies and their properties.

Once we have presented the necessary algebra, we will illustrate the specifics of the supersingular isogeny Diffie-Hellman key-exchange protocol. We will spend most of this time dedicated to a modular deconstruction of the protocol, reviewing the underlying isogeny-level procedures and algorithms which will be necessary for understanding in detail the signature protocol to come. Another task of this section will be to introduce the SIDH C library released by Microsoft Research, on top of which the core contributions of this thesis are implemented. This subsection will end with a briefing and analysis of the closely related zero-knowledge proof of identity (ZKPoI) isogeny protocol proposed in the original De Feo et al. paper[ref], as it is necessary for understanding the isogeny based signature scheme presented by Yoo et al[ref].

In section 2.3 we will discuss the Fiat-Shamir transformation; a technique which, given a secure interactive proof of knowledge, creates a secure digital signature scheme. We will also look at the quantum-secure adaptation published by Unruh, for applying a non-quantum-resistant transform to a quantum-resistant primitive would be rather frivolous.

Finally, the last section of this chapter will be dedicated to covering current isogeny-based signature schemes - the topic of which this dissertation is mainly concerned. We will primarily discuss the signature scheme of Yoo et al., which is a near direct application of Unruh's Fiat-Shamir adaptation to the SIDH zero-knowledge proof of identity to be discussed at the end of section 2.2.

2.1 Algebraic Geometry & Isogenies

Groups & Varieties. A **group** is a 2-tuple composed of a set of elements and a corresponding group operation. Given some group defined by the set G and the group operation $\cdot$ (written as $(G, \cdot)$) it is typical to refer to the group simply as G. If $\cdot$ is equivalent to some rational mapping[footnote about rational mappings] $f_G : G \to G$, then the group $(G, \cdot)$ is said to form an *algebraic variety*[footnote about the inverse function]. A group which is also an algebraic variety is referred to as an **algebraic group**.

G is said to be an *abelian* group if, in addition to the four traditional group axioms (closure, associativity, existence of an identity, existence of an inverse), G satisfies the condition of commutitiviy. More formally, for some group G with group operation $\cdot$, we say G is an abelian group iff $x \cdot y = y \cdot x \ \forall x, y \in G$. An algebraic group which is also abelian is referred to as an **abelian variety**.

Definition 1 (Abelian Variety)**.** for some algebraic group G with operation $\cdot$, we say G is an abelian variety iff $x \cdot y = y \cdot x \ \forall x, y \in G$.

For some group $(G, \cdot)$, some $x, y \in G$, and some rational mapping $f_G : G \to G$, let the following sequence of implications denote the classification of $(G, \cdot)$:

$$\text{group} \xrightarrow{x \cdot y = f_G(x,y)} \text{algebraic group} \xrightarrow{x \cdot y = y \cdot x} \text{abelian variety}$$

Morphisms. Let us again take for example some group $(G, \cdot)$. Let's also define some set $S_{(G,\cdot)}$ which contains every tuple (x, y, z) for group elements x, y, z which satisfy $x \cdot y = z$.

$$S_{(G,\cdot)} = \{x, y, z \in G | x \cdot y = z\}$$

Take also for example a second group $(H, *)$ and some map $\phi : G \to H$. ϕ is said to be *structure preserving* if the following implication holds:

$$(x, y, z) \in S_{(G,\cdot)} \Rightarrow (\phi(x), \phi(y), \phi(z)) \in S_{(H,*)}$$

A **morphism** is simply the most general notion of a structure preserving map. More specifically, in the domain of algebraic geometry, we will be dealing with the notion of a **group homomorphism**, defined as follows:

Definition 2 (Group Homomorphism)**.** For two groups G and H with respective group operations $\cdot$ and $*$, a group homomorphism is a structure preserving map $h : G \to H$ such that $\forall u, v \in G$ the following holds:

$$h(u \cdot v) = h(u) * h(v)$$

From this simple definition, two more properties of homomorphisms are easily deducible. Namely, for some homomorphism $h : G \to H$, the following properties hold:

- h maps the identity element of G onto the identity element of H, and
- $h(u^{-1}) = h(u)^{-1}, \forall u \in G$

Furthermore, an **endomorphism** is a special type of morphism in which the domain and the codomain are the same groups.

Definition 3 (Endomorphism)**.** For two groups G and H, an endomorphism is a morphism $\psi : G \to H$ wherein $G = H$

Fields & Field Extensions. An algebraic group G_a is defined over a field K if each element $e \in G_a$ is defined over K and the corresponding f_{G_a} is also defined over K. To show that a particular algebraic group G_a is defined over some field K we will henceforth denote the group/field pairing as $G_a(K)$.

Quotient Groups.

These algebraic structures are all important for building up to the concept of an *isogeny*. The lowest-level structure we will be concerned with when discussing the forthcoming isogeny-based protocols will typically be abelian varieties. The lowest-level structure in the SIDH C codebase is a finite field element.

2.1.1 Elliptic Curves

An elliptic curve is an algebraic curve defined over some field K, the most general representation of which is given by

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6.$$

This representation encapsulates elliptic curves defined over any field. If, however, we are in the dicussion of curves defined over some field K such that $\text{char}(K) > 3$[ref], then the more compact form $y^2 = x^3 + ax + b$ can be applied. In this dissertation we will default to this second representation, as the schemes with which we are concerned will always be defined over $\mathbb{F}_p$ for some large prime p.

Within algebraic geometry, it is common practice to define a group structure over the points of a given elliptic curve (or any other smooth cubic curve). If we wish to define a group in accordance to a particular curve, we do so with the following notation:

$$E : y^2 = x^3 + ax + b$$

Wherein E denotes the group in question, the elements of which are all the points (solutions) of the curve. The corresponding group operation + is better understood geometrically than algebraically. Consider the following.

Given two elements P and Q of some arbitrary elliptic curve group $E(\mathbb{F}_q)$ defined over some arbitrary finite field $\mathbb{F}_q$, we define + geometrically as follows: drawing the line formed by points P and Q, we follow this line to its third intersection on the curve, which we will denote as $R = (x_R, y_R)$. We then set $P + Q = -R$. $-R$ is necessarily defined as $(x_R, -y_R)$. Direct your attention to figure [fig] for an illustrated representation of this process.

The group operation + is referred to as *pointwise addition*. In order for E to properly form a group under pointwise addition, it must satisfy the four group axioms.

- *Closure*: Because elliptics curves are polynomials of degree of 3, we know any given line passing through two points P and Q of E will pass through a third point R. The exceptions here are twofold. First, when $P = Q$ and thus our line is tangent to E, and second, when $Q = -P$ and our line is parallel with the y-axis. We resolve the first case nicely by defining $P + P$ as the resulting $-R$ given the second

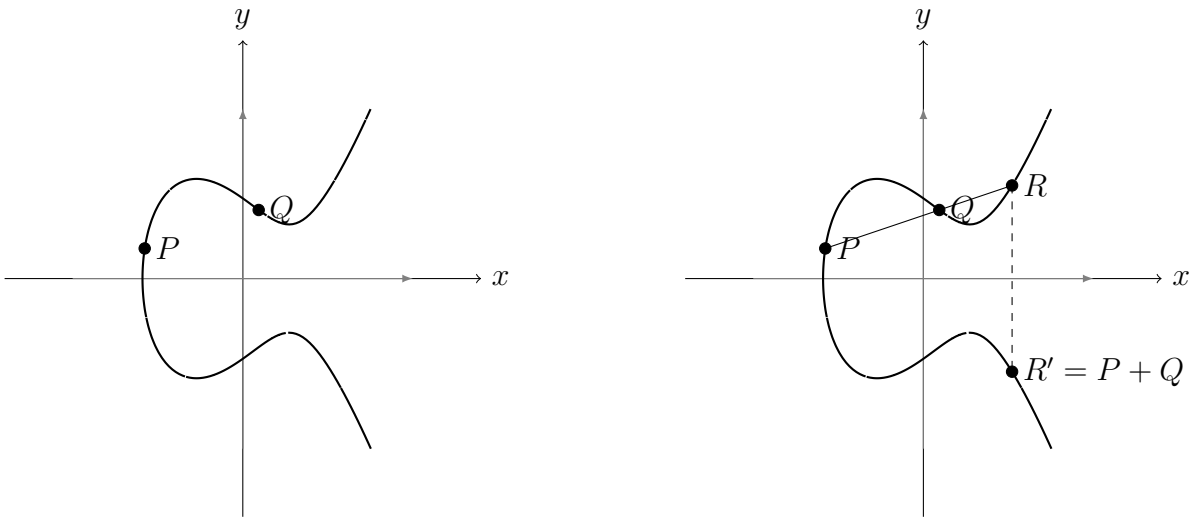


Figure 2.1: + acting over points P and Q of some curve E.

intersection, R, of the tangent line with E. In the second case, $P + (-P)$ should yield the identity element of the group. We will define this element and resolve this issue below.

- *Identity*:
- *Associativity*: Assumin
- *Inverse*: Because elliptic curves are symmetrical about the x-axis, we can take the inverse element of P to be $-P$, defined the same as above. It is easy to see that $P + (-P) = \mathcal{O}$

Additionally, we shorthand n-many applications of the group operation of a point P to itself as nP, analogous to scalar multiplication.

Moreover, because groups defined over elliptic curves in this fashion are commutitive, they also constitute abelian varieties[ref]. We will see later other possible algebraic representations of elliptic curves which have varying benefits. The point at infinity.

When referring to curves as abelian varieties defined over a field, we will write them as $E_\alpha(K)$, for some curve E_α and some field K. If we are only concerned with the geometric properties of the curve, or curves as distinct elements of some group structure, it will suffice to write E_α. Moving forward from here, we will assume all general curves discussed are capable of definition over some finite field $\mathbb{F}_q$.

Torsion Groups. The *r-torsion group* of E is the set of all points $P \in E(\overline{\mathbb{F}}_q)$ such that $[r]P = \mathcal{O}$. We denote the r-torsion group of some curve as $E[r]$.
Supersingular Curves. An elliptic curve can be either *ordinary* or *supersingular*. There are several equivalent ways to define supersingular curves (and thus the distinction between them and ordinary curves,)

For the remainder of this paper, unless otherwise noted, all elliptic curves in discussion will be of the supersingular variety.

2.1.2 Isogenies & Their Properties

Definition 4 (Isogeny)**.** Let G and H be algebraic groups[ref]. An isogeny is a morphism[ref] $h : G \to H$ possessing a finite kernel.

In the case of the above definition where G and H are abelian varieties (such as elliptic curves,) the isogeny h is homomorphic[ref] between G and H. Because of this, isogenies over elliptic curves (and other abelian varieties) inherit certain characteristics.
For an isogeny $h : E_1 \to E_2$ defined over elliptic curves E_1 and E_2, the following holds:

- $h(\mathcal{O}) = \mathcal{O}$, and
- $h(u^{-1}) = h(u)^{-1}, \forall u \in G$

We denote $End(E)$ to denote the ring formed by all the isogenies acting over E which are also endomorphisms. Note that m-repeated pointwise addition of a point with itself can equivalently be modelled by an endomorphism, we denote the application of such an endomorphism to a point P as $[m]P$, such that $[m] : E \to E$ and $[m]P = mP$.

2.2 Supersingular Isogeny Diffie-Hellman

This section will aim to accomplish two things. First, we will briefly explain the isogeny-level & key-exchange-level procedures of the SIDH protocol. Second, we will illuminate how these procedures map onto Microsoft Research's C implementation of SIDH. In this regard, this section can be considered an attempt to meld two domains of SIDH functions & procedures, in hopes of easing the navigation from the SIDH protocol to Microsoft's C implementation, and vice versa.

The original work of De Feo, Jao, and Plut outlines three different isogeny-based cryptographic primitives: Diffie-Hellman-esque key exchange, public key encryption, and the aforementioned zero-knowledge proof of identity. Because all three of these protocols require the same initialization and public parameters, we will begin by covering these parameters in detail. Immediately after, we will analyze the key exchange at a relatively high level. Our goal of this section is to explain in detail the algorithmic and cryptographic aspects of the ZKPoI scheme, as this forms the conceptual basis for the signature scheme we will be investigating. We begin with the key exchange protocol because its sub-routines are integral to the Yoo et. Al signature implementation.

For the discussion that follows, we will assume every instance of an SIDH protocol occurs between two parties, A and B (eg. Alice & Bob,) for which we will colorize information particular to A in blue and B in red. This will include private keys & public keys as well as the variables and constants used in their generation.

2.2.1 Public Parameters

As the name suggests, SIDH protocols work over supersingular curves (of a smooth order). Let $\mathbb{F}_q = \mathbb{F}_{p^2}$ be the finite field over which our curves are defined, $\mathbb{F}_{p^2}$ denoting the quadratic extension field of $\mathbb{F}_p$. p is a prime defined as follows:

$$p = \ell_A^{e_A} \ell_B^{e_B} \cdot f \pm 1$$

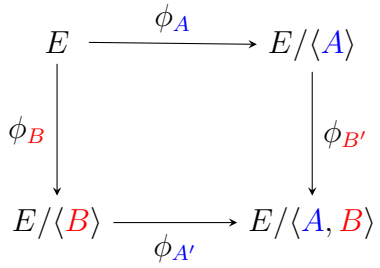
Wherein ℓ_A and ℓ_B are small primes (typically 2 & 3, respectively) and f is a cofactor ensuring the primality of p. We then define a global parameter curve E defined over $\mathbb{F}_q$

with cardinality $(\ell_A^{e_A}\ell_B^{e_B}f)^2$. Consequently, the torsion group $E[\ell_A^{e_A}]$ is $\mathbb{F}_q$-rational and has ${\ell_A}^{{e_A}-1}(\ell_A+1)$ cyclic subgroups of order $\ell_A^{e_A}$, with the analogous statement being true for $E[\ell_B^{e_B}]$.

Additionally, we include in the public parameters the bases $\{P_A, Q_A\}$ and $\{P_B, Q_B\}$, generating $E[\ell_A^{e_A}]$ and $E[\ell_B^{e_B}]$ respectively.

2.2.2 SIDH Key Exchange

This subsection will illustrate an SIDH key exchange run between party members Alice and Bob. The general idea of the protocol can be surmised by the diagram below. In the scheme, **private keys** take the form of isogenies[ref] defined with domain E, and **public keys** are the associated co-domain curve of said isogenies.



Key Generation. Alice chooses two random numbers $m_A, n_A \in \mathbb{Z}/\ell_A^{e_A}\mathbb{Z}$ such that $(\ell_A \nmid m_A) \vee (\ell_A \nmid n_A)$. Following this, Alice computes the isogeny $\phi_A : E \to E_A$ with kernel $K_A := \langle [m_A]P_A, [n_A]Q_A\rangle$.

PK Exchange. In addition to sending their respective isogenies,

Secret Agreement.

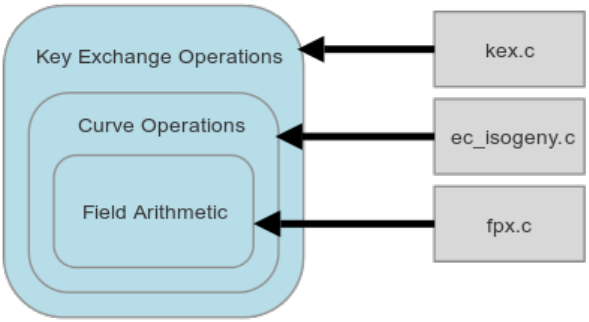


Figure 2.2: Relationship between SIDH key exchange & MR SIDH C library

2.2.3 Modular Breakdown

Figure [ref] illustrates the relationship between abstraction levels of the SIDH protocol and modules of the SIDH C library.

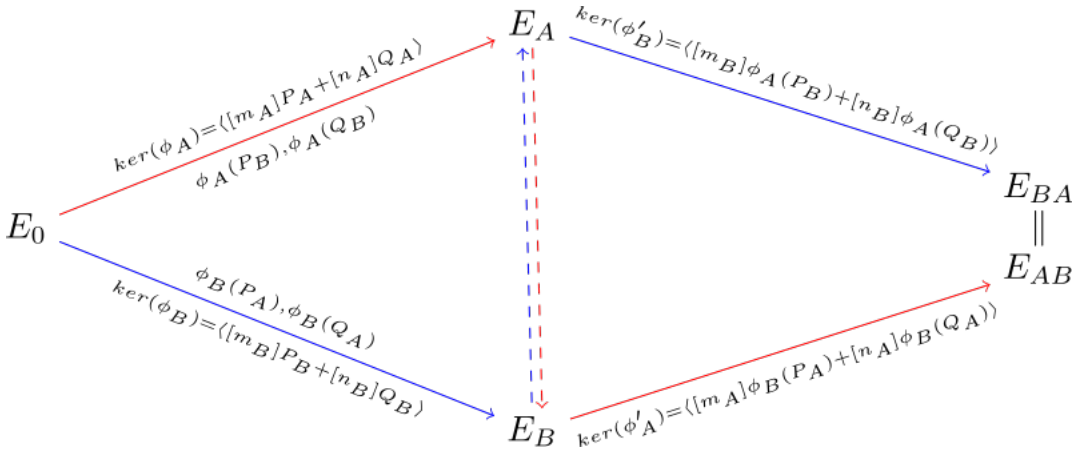


Figure 2.3: SIDH key exchange between Alice & Bob

2.2.4 Security Assumptions

2.2.5 Zero-Knowledge Proof of Identity

Recall the notion of a simple identification scheme:

2.3 Fiat-Shamir Construction

The Fiat-Shamir construction (sometimes referred to as the Fiat-Shamir heuristic or transform) is used

2.3.1 Unruh's Post-Quantum Adaptation

2.4 Isogeny Based Signatures

Now that we've introduced the zero-knowledge proof of identity scheme from [REFERENCE] as well as Unruh's quantum-safe Fiat-Shamir adaption, the isogeny based signature scheme presented by Yoo et. Al is a near-trivial application of the latter to the former.

The isogeny based signature scheme presented by Yoo et. Al is defined, in the traditional manner, by a tuple of algorithms. Namely, the scheme is defined by the tuple (KeyGen, Sign, Verify) with each algorithm loosely defined as follows:
KeyGen(): Select a random point S of order $\ell_A^{e_A}$, compute the isogeny $\phi : E \to E/\langle S\rangle$. Return (pk, sk) where pk $= (E/\langle S\rangle, \phi(P_B), \phi(Q_B))$ and sk $= S$.
Sign():
Verify():

2.4.1 Modular Breakdown

Shortly after, the following, more in-depth algorithms are given as definitions:

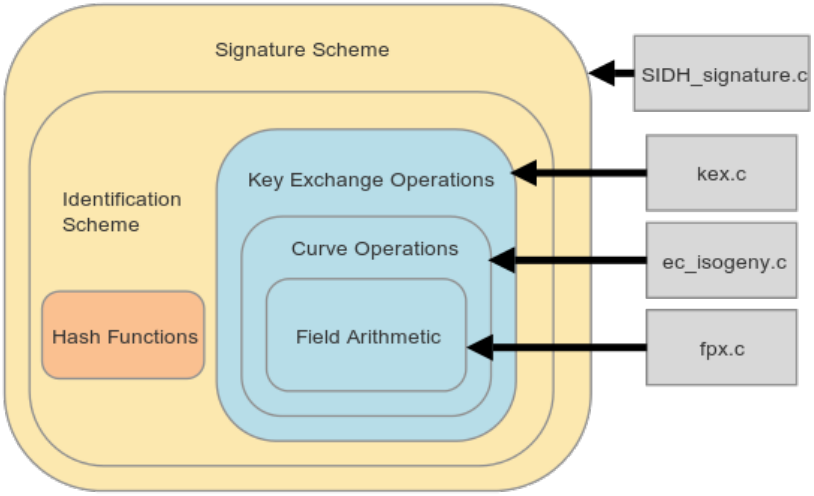


Figure 2.4: Relationship between SIDH based signatures & Our fork of the SIDH C library

Algorithm 1 KeyGen(λ)

Pick a random point S of order $\ell_A^{e_A}$
Compute the isogeny $\phi : E \to E/\langle S \rangle$
pk $\leftarrow (E/\langle S \rangle, \phi(P_B), \phi(Q_B))$
sk $\leftarrow S$
return (pk,sk)

Algorithm 3 Verify(pk, m, σ)

$J_1 \parallel ... \parallel J_{2\lambda} \leftarrow H(pk, m, (\mathtt{com}_i)_i, (\mathtt{ch}_{i,j})_{i,j}, (h_{i,j})_{i,j})$
for `i = 0..2`λ **do**
    check $h_{i,J_i} = G(\mathtt{resp}_{i,J_i})$
    if $\mathtt{ch}_{i,J_i} = 0$ **then**
        Parse $(R, \phi(R)) \leftarrow \mathtt{resp}_{i,J_i}$
        check $(R, \phi(R))$ have order $\ell_B^{e_B}$
        check R generates the kernel of the isogeny $E \to E_1$
        check $\phi(R)$ generates the kernel of the isogeny $E/\langle S \rangle \to E_2$
    else
        Parse $\psi(S) \leftarrow \mathtt{resp}_{i,J_i}$
        check $\psi(S)$ has order $\ell_A^{e_A}$
        check $\psi(S)$ generates the kernel of the isogeny $E_1 \to E_2$
if all checks succeed **then**
    return 1

If we transcribe the above to the language of the Microsoft SIDH API, we have in essense the following:

Algorithm 2 Sign(sk, m)

for `i = 1..2`λ **do**
  Pick a random point R of order $\ell_B^{e_B}$
  Compute the isogeny $\psi : E \to E/\langle R \rangle$
  Compute either $\phi' : E/\langle R \rangle \to E/\langle R, S \rangle$ or $\psi' : E/\langle S \rangle \to E/\langle R, S \rangle$
  $(E_1, E_2) \leftarrow (E/\langle R \rangle, E/\langle R, S \rangle)$
  $\texttt{com}_i \leftarrow (E_1, E_2)$
  $\texttt{ch}_{i,0} \leftarrow_R \{0, 1\}$
  $(\texttt{resp}_{i,0}, \texttt{resp}_{i,1}) \leftarrow ((R, \phi(R)), \psi(S))$
  if $\texttt{ch}_{i,0} = 1$ **then**
    $\texttt{swap}(\texttt{resp}_{i,0}, \texttt{resp}_{i,1})$
  $h_{i,j} \leftarrow G(\texttt{resp}_{i,j})$
$J_1 \parallel ... \parallel J_{2\lambda} \leftarrow H(pk, m, (\texttt{com}_i)_i, (\texttt{ch}_{i,j})_{i,j}, (h_{i,j})_{i,j})$
return $\sigma \leftarrow ((\texttt{com}_i)_i, (\texttt{ch}_{i,j})_{i,j}, (h_{i,j})_{i,j}, (\texttt{resp}_{i,J_i})_i)$

Algorithm 4 KeyGen(λ)

(pk, sk) $\leftarrow$ `KeyGeneration_B()`
return (pk,sk)

Algorithm 5 Sign(sk, m)

for `i = 1..2`λ **do**
  $(, R, \psi) \leftarrow$ `KeyGeneration_A(E)`
  $E_1 \leftarrow E/\langle R \rangle$
  $(E_2, E/\langle R, S \rangle) \leftarrow$ `SecretAgreement_B()`
  $(E_1, E_2) \leftarrow (E/\langle R \rangle, E/\langle R, S \rangle)$
  $\texttt{com}[i] \leftarrow (E_1, E_2)$
  $\texttt{ch}[i] \leftarrow_R \{0, 1\}$
  $(\texttt{resp}[i]_0, \texttt{resp}[i]_1) \leftarrow ((R, \phi(R)), \psi(S))$
$J_1 \parallel ... \parallel J_{2\lambda} \leftarrow H(pk, m, (\texttt{com}_i)_i, (\texttt{ch}_i)_i, (h_{i,j})_{i,j})$
return $\sigma \leftarrow ((\texttt{com}_i)_i, (\texttt{ch}_{i,j})_{i,j}, (h_{i,j})_{i,j}, ((\texttt{resp})[J_i])$

Chapter 3

Batching Operations for Isogenies

3.1 Batching Procedure in Detail

One of our main contributions is the embedding of a low-level $\mathbb{F}_{p^2}$ procedure into Microsofts pre-existing SIDH library. The procedure in question reduces arbitrarily many unrelated/potentially parallel $\mathbb{F}_{p^2}$ inversions to a sequence of $\mathbb{F}_p$ multiplications & additions, as well as one $\mathbb{F}_p$ inversion.

More specifically, the procedure takes us from n $\mathbb{F}_{p^2}$ inversions to:

- $2n$ $\mathbb{F}_p$ squarings
- n $\mathbb{F}_p$ additions
- 1 $\mathbb{F}_p$ inversion
- 3(n-1) $\mathbb{F}_p$ multiplications
- $2n$ $\mathbb{F}_p$ multiplications

The procedure is as follows:

3.1.1 Projective Space

Because the work of Yoo et al. was built on top of the original Microsoft SIDH library, all underlying field operations (and isogeny arithmetic) are performed in projective space. Doing field arithmitic in projective space allows us to avoid many inversion operations. The downside of this (for our work) is that the number opportunities for implementing the batched inversion algorithm becomes greatly limited.

3.1.2 Remaining Opportunities

There are two functions called in the isogeny signature system that perform a $\mathbb{F}_{p^2}$ inversion: j_inv and inv_4_way. These functions are called once in SecretAgreement and KeyGeneration operations respectively. SecretAgreement and KeyGeneration are in turn called from each signing and verification thread.

Algorithm 6 Batched Partial-Inversion

```
1: procedure PARTIAL_BATCHED_INV(F_{p^2}[ ] VEC, F_{p^2}[ ] DEST, INT N)
2:     initialize F_p den[n]
3:     for i = 0..(n-1) do
4:         den[i] ← a[i][0]^2 + a[i][1]^2
5:     a[0] ← den[0]
6:     for i = 1..(n-1) do
7:         a[i] ← a[i-1]*den[i]
8:     a_inv ← inv(a[n-1])
9:     for i = n-1..1 do
10:        a[i] ← a_inv * dest[i − 1]
11:        a_inv ← a_inv * den[i]
12:    dest[0] ← a_inv
13:    for i = 0..(n-1) do
14:        dest[i][0] ← a[i] * vec[i][0]
15:        vec[i][1] ← −1 * vec[i][1]
16:        dest[i][1] ← a[i] * vec[i][1]
```

This means that in the signing procedure there are 2 opportunities for implementing batched partial-inversion with a batch size of 248 elements. In the verify procedure, however, there are 3 opportunities for implementing batched inversion with a batch size of roughly 124 elements.

3.2 Implementation

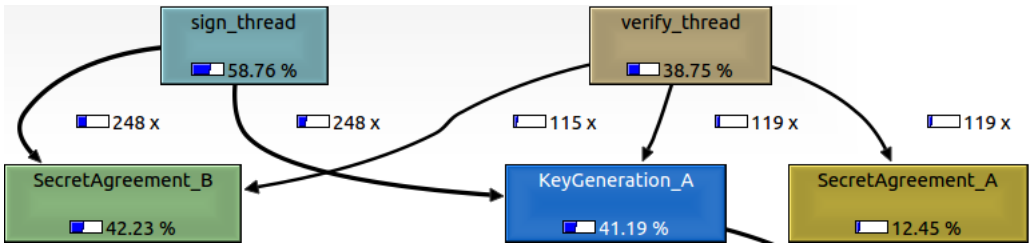


Figure 3.1: ¡Caption here¿

3.3 Results

Two different machines were used for benchmarking. System A denotes a single-core, 1.70 GHz Intel Celeron CPU. System B denotes a quad-core, 3.1 GHz AMD A8-7600.

The two figures below provide benchmarks for KeyGen, Sign, and Verify procedures with both batched partial inversion implemented (in the previously mentioned locations) and not implemented. All benchmarks are averages computed from 100 randomized sample runs. All results are measured in clock cycles.

Procedure	System A Without Batching	System A With Batching
KeyGen	68,881,331	68,881,331
Signature Sign	15,744,477,032	15,565,738,003
Signature Verify	11,183,112,648	10,800,158,871

Procedure	System B Without Batching	System B With Batching
KeyGen	84,499,270	84,499,270
Signature Sign	10,227,466,210	10,134,441,024
Signature Verify	7,268,804,442	7,106,663,106

System A: With inversion batching turned on we notice a 1.1 % performance increase for key signing and a 3.5 % performance increase for key verification.

System B: With inversion batching turned on we a observe a 0.9 % performance increase for key signing and a 2.3 % performance increase for key verification.

3.3.1 Analysis

It should first be noted that, because our benchmarks are measured in terms of clock cycles, the difference between our two system clock speeds should be essentially ineffective.

In the following table, "Batched Inversion" signifies running the batched partial-inversion procedure on 248 $\mathbb{F}_{p^2}$ elements. The procedure uses the binary GCD $\mathbb{F}_p$ inversion function which, unlike regular $\mathbb{F}_{p^2}$ montgomery inversion, is not constant time.

Procedure	Performance
Batched Inversion	1721718
$\mathbb{F}_{p^2}$ Montgomery Inversion	874178

Do performance increases observed make sense?

Chapter 4

Compressed Signatures

4.1 Compression of Public Keys

We discussed rejection sampling A values from signature public keys until we found an A that was also the x-coord of a point. After some simple analysis, however, we found that it was extremely unlikely for A to be a point on the curve.

4.1.1 ¡Sub-section title¿

4.1.2 ¡Sub-section title¿

some text[2], some more text

4.1.3 ¡Sub-section title¿

4.1.4 ¡Sub-section title¿

Refer figure 3.1.

4.1.5 ¡Sub-section title¿

4.2 Implementation

4.3 Results

Chapter 5

Discussion & Conclusion

5.1 Results & Comparisons

5.2 Additional Opportunities for Batching

5.3 Future Work

¡Conclusion here¿

Acknowledgments

¡Acknowledgements here¿

¡Name here¿

¡Month and Year here¿
National Institute of Technology Calicut

References

[1] ¡Name of the reference here¿, `<urlhere>`

[2] ¡Name of the reference here¿, `<urlhere>`