Projet de programmation en C - Rapport

Échange de clés sur les graphes d'isogénies de courbes supersingulières

Mathilde CHENU - DE LA MORINERIE

Encadrants : Luca DE FEO Michael QUISQUATER

UNIVERSITÉ DE VERSAILLES ST-QUENTIN-EN-YVELINES

Table des matières

Table des matières **2**

Remerciements **3**

Introduction **4**

1 Fondements mathématiques **5**

2 Protocole d'échange de clés **6**

3 Optimisation du protocole **7**

3.1 Représentation des données 8

3.2 Accélération des calculs 8

4 Choix des nombres premiers **10**

4.1 Le cas des 2-isogénies 10

4.2 Le cas des 3-isogénies 11

4.3 Le cas général des isogénies de degré impair 11

5 Implémentation **12**

5.1 Structure du programme 12

5.2 Résultats 13

5.3 Difficultés rencontrées 15

Conclusion **16**

Références **17**

Remerciements

Merci à Luca De Feo de m'avoir donné l'opportunité d'approfondir ce sujet qui me tenait à cœur.

Merci également d'avoir répondu rapidement à mes questions plus ou moins urgentes, plus ou moins pertinentes, et plus ou moins tardives (voire matinales).

Introduction

La création d'un ordinateur fonctionnant avec les lois de la physique quantique serait un bouleversement en cryptographie, puisqu'un tel ordinateur pourrait résoudre les problèmes reposants sur la factorisation et le logarithme discret en temps polynomial. Ainsi, les algorithmes cryptographiques reposants sur ces deux problèmes, qui ne peuvent être résolus qu'en temps exponentiel jusqu'ici, deviendraient obsolètes.

Face à cette menace, le National Institute of Standards and Technology (NIST) a lancé un appel à contribution pour standardiser un ou plusieurs systèmes cryptographiques post-quantiques, c'est-à-dire résistants aux attaques par des ordinateurs quantiques. Ces systèmes sont essentiellement de cinq types :

- Les systèmes exploitant les *réseaux* et la difficulté de trouver un vecteur court, ou une base orthonormée sur un réseau euclidien.
- Les systèmes utilisant un *code correcteur d'erreur* comme clé secrète.
- Les systèmes basés sur les *systèmes de polynômes multivariés*, reposant sur la difficulté de trouver une solution, et la facilité de vérification d'une solution.
- Les systèmes reposants sur les *fonctions de hachage*.
- Les systèmes basés sur la difficulté de trouver un chemin dans les *graphes d'isogénies de courbes elliptiques supersingulières*.

Ces protocoles doivent non seulement être résistants aux attaques d'un ordinateur quantique, mais aussi efficaces et rapides pour pouvoir être déployés à grande échelle. C'est pourquoi une grande importance est accordée au choix des paramètres et à l'implémentation.

Dans la suite de ce rapport, on s'intéressera aux isogénies de courbes elliptiques supersingulières, en présentant tout d'abord les fondements mathématiques et le protocole d'échange de clés qui en découle, puis les techniques d'optimisation, avant d'étudier le choix des paramètres en fonction de la sécurité et de la rapidité du protocole. Pour finir, la section 5 donne des détails sur l'implémentation du protocole d'échange de clés réalisée sur des les graphes de 2 et 3 isogénies, puis sur les graphes d'isogénies de degré impair quelconque, conformément à l'objectif du projet.

1 Fondements mathématiques

Soit un corps fini $\mathbb{F}_{p^n}$.

Une *isogénies* est un homomorphisme de courbes elliptiques surjectif. Dans le cas d'isogénies séparables, le degré d'une isogénie est égal au cardinal son noyau. Ce noyau décrit l'isogénie de façon unique [15]. On écrira donc $E / < \ker \phi >$ pour désigner l'image E' d'une courbe elliptique E par une isogénie ϕ.

Si le noyau contient ℓ points, on parle dans ce cas de *ℓ-isogénie* [15]. Par ailleurs, toute isogénie de degré ℓ tel que ℓ ne divise pas p peut être écrite comme la composée d'isogénie de degré premier sur $\overline{\mathbb{F}}_{p^n}$ [5]. Deux courbes elliptiques E et E' sont dîtes ℓ-isogènes s'il existe une ℓ-isogénie ϕ entre E et E'. On peut alors définir une ℓ-isogénie duale ψ entre E' et E. L'isogénie duale est telle que $\phi \circ \psi = [\ell]$. La relation "être isogène" définit ainsi une relation d'équivalence sur les courbes elliptiques.

Une courbe E est dite *supersingulière* si l'anneau des endomorphismes de E, c'est-à-dire l'ensemble des isogénies de E dans E avec le morphisme nul, est isomorphe à une algèbre de quaternions.[12] Les courbes supersingulières sont toutes définies sur $\mathbb{F}_{p^2}$, et sont isogènes, à isomorphisme près. Il n'y a donc qu'une seule classe d'équivalence pour la relation "être isogène".

Le *graphe de ℓ-isogénie* est le graphe dont les points sont formés par les classes d'isomorphismes de courbes elliptiques. Deux points sont reliés dans le graphe s'il existe une ℓ-isogénie entre les courbes. Dans le cas des courbes supersingulières, ce graphe est $l+1$-régulier, et a la propriété d'être expanseur [14]. Cela signifie en particulier qu'une marche aléatoire suffisamment longue à la même probabilité d'arriver en chaque point. L'unicité de la classe d'isogénie dans le cas supesingulier implique que les graphes de ℓ-isogénies sont tous définit sur le même ensemble de points.

Contrairement aux groupes habituellement utilisés dans les protocoles de Diffie-Hellman [6], il n'y a pas de groupe commutatif agissant sur les isogénies. En revanche, en se plaçant dans $\mathbb{F}_p^2$ avec :

- p un nombre premier de la forme $p = f \cdot \ell_a^{e_a} \ell_b^{e_b} \pm 1$, où ℓ_a et ℓ_b sont premiers, et f un cofacteur tel que p soit premier,
- E une courbe elliptique supersingulière de cardinal $(p \pm 1)^2 = (f \cdot \ell_a^{e_a} \ell_b^{e_b})^2$,
- G_a et G_b deux points de $E[\ell_a^{e_a}]$ et $E[\ell_b^{e_b}]$, les sous-groupes de $\ell_a^{e_a}$ et $\ell_b^{e_b}$ torsion respectivement, générant les noyaux des isogénies ϕ_a et ϕ_b,

le diagramme suivant commute :

$$\begin{array}{ccc} E & \xrightarrow{\phi_a} & E_a = E/ < G_a > \\ \Big\downarrow{\scriptstyle \phi_b} & & \Big\downarrow{\scriptstyle \phi_{ab}} \\ E_b = E/ < G_b > & \xrightarrow{\phi_{ba}} & E_b \Big/ < \phi_b(G_a) > = E_a \Big/ < \phi_a(G_b) > = E_{ab} \end{array}$$

C'est à la fois la commutativité du diagramme qui va permettre de réaliser un protocole à la Diffie-Hellman, et la non-commutativité des isogénies qui protège le protocole des attaques existantes utilisant des ordinateurs quantiques.

Ce diagramme est au coeur du protocole d'échange de clés présenté ci-dessous.

2 Protocole d'échange de clés

Pour pouvoir exploiter ce diagramme en cryptographie, il faut arriver à en extraire un secret partagé à partir d'une clé publique et d'une clé privée, sans que le protocole d'échange ne révèle la clé privée.

Pour faire apparaître ces éléments publiques et privés, on va décomposer les points G_a en $[m_a]P_a + [n_a]Q_a$ et G_b en $[m_b]P_b + [n_b]Q_b$, où :

- $\diamond$ m_a, n_a, m_b et n_b appartiennent à $\mathbb{Z}/\ell_a^{e_a}$ et $\mathbb{Z}/\ell_b^{e_b}$ respectivement et sont gardés secrets.
- $\diamond$ (P_a, Q_a), (P_b, Q_b) sont des points publiques, bases respectives de $E[\ell_a^{e_a}]$ et $E[\ell_b^{e_b}]$.

Comme les isogénies sont des homomorphismes, on obtient alors que

$$\phi_b(G_a) = [m_a]\phi_b(P_a) + [n_a]\phi_b(Q_a)$$

$$\phi_a(G_b) = [m_b]\phi_a(P_b) + [n_b]\phi_a(Q_b) \ .$$

Le protocole est le suivant : Alice va calculer les flèches horizontales du diagramme commutatif, et Bob les flèches verticales, grâce à leurs clés respectives et à l'échange de certains points auxiliaires.

Concrètement, à partir d'une clé publique P_a et Q_a, Alice choisit une clé secrète m_a et n_a, et calcule $G_a = [m_a]P_a + [n_a]Q_a$ sur la courbe E. Elle peut alors calculer sa clé publique $E_a = E/ < G_a >$, $\phi_a(P_b)$ et $\phi_a(Q_b)$ qu'elle envoie à Bob.

Bob procède de la même façon. A partir de la courbe et des points fournis par Alice et de sa clé privée, il peut alors calculer $E_a/ < \phi_a(G_b) >= E_a/ < [m_b]\phi_a(P_b) + [n_b]\phi_a(Q_b)$ et ainsi obtenir une courbe E_{ab}.

Alice procède de même et obtient une courbe isomorphe à E_{ab}. Le secret commun est alors le j-invariant des courbes obtenues par Alice et Bob respectivement.

Formellement, le protocole d'échange de clés est donc le suivant :

Paramètres publiques : $p = f \cdot \ell_a^{e_a} \ell_b^{e_b} \pm 1$,
E supersingulière de cardinal $(p \pm 1)^2 = (f \cdot \ell_a^{e_a} \ell_b^{e_b})^2$,
(P_a, Q_a) et (P_b, Q_b) bases respectives de $E[\ell_a^{e_a}]$ et $E[\ell_b^{e_b}]$.

$\mathcal{A}$lice		$\mathcal{B}$ob
Clé secrète :		**Clé secrète :**
$m_a, n_a \in_R \mathbb{Z}/\ell_a^{e_a}\mathbb{Z}$		$m_b, n_b \in_R \mathbb{Z}/\ell_b^{e_b}\mathbb{Z}$
Clé publique :		**Clé publique :**
$E_a = E \Big/ < [m_a]P_a + [n_a]Q_a >$		$E_b = E \Big/ < [m_b]P_b + [n_b]Q_b >$
$\phi_a(P_b), \phi_a(Q_b)$		$\phi_b(P_a), \phi_b(Q_a)$
	$E_a, \phi_a(P_b), \phi_a(Q_b)$ $\rightharpoonup$	
	$\leftharpoondown$ $E_b, \phi_b(P_a), \phi_b(Q_a)$	
Calcul du secret :		**Calcul du secret :**
$G_{ab} = [m_a]\phi_b(P_a) + [n_a]\phi_b(Q_a)$		$G_{ba} = [m_b]\phi_a(P_b) + [n_b]\phi_a(Q_b)$
$E_{ab} = E_b \Big/ < G_{ab} >$		$E_{ba} = E_a \Big/ < G_{ba} >$
Secret commun :		**Secret commun :**
$j(E_{ab})$		$j(E_{ba})$

De façon plus imagée, Alice se promène sur le graphe de l_a-isogénies en réalisant deux fois le même chemin : une fois à partir de la courbe initiale E, puis à partir de la courbe E_b de Bob. Celui-ci procède de même sur le graphe de l_b-isogénies. Le fait que ces deux graphes recouvrent les même points et la commutativité du diagramme ci-dessus assure qu'Alice et Bob arriveront à terme au même point du graphe.

La sécurité du protocole repose sur la difficulté conjecturée de trouver une isogénie entre deux courbes données. Ainsi, la connaissance de E et E_a par un attaquant ne lui permet pas de retrouver le générateur (secret) du noyau de l'isogénie, et ne lui permet donc pas de retrouver la clé secrète d'Alice, même en connaissant les points auxiliaires $\phi_a(P_b)$ et $\phi_a(Q_b)$. Cela implique également que la connaissance de E_a et E_b, et des points auxiliaires révélés, ne permet pas de calculer E_{ab}.

Il est intéressant de remarquer que ce même problème de calcul d'isogénie sur des courbes ordinaires peut-être résolu en temps subexponentiel avec un ordinateur quantique. Cependant, les algorithmes utilisent l'existence d'une action de groupe commutative dans le cas ordinaire, et ne peuvent donc pas être utilisés dans le cas supersingulier. Par ailleurs, le fait que le logarithme discret soit facile sur les courbes supersingulières ne diminue pas la sécurité du protocole décrit ci-dessus.

3 Optimisation du protocole

Le protocole décrit précédemment doit pouvoir être exécuté rapidement afin de pouvoir être utilisé à grande échelle. Pour cela, un certain nombre de techniques permettent d'optimiser les calculs effectués, et sont présentées dans cette section.

Ces techniques sont de deux types : la représentation efficace des données et l'optimisation des calculs.

3.1 Représentation des données

La façon de représenter les données dans le protocole décrit précédemment est redondante. En effet, le sous-groupe engendré par $[m_a]P_a + [n_a]P_b$ est le même que celui engendré par $[m_a n_a^{-1}]P_a + Q_a$. On peut donc remplacer la clé secrète m_a, n_a par une clé $s_a = m_a n_a^{-1}$, qui utilise deux fois moins de place.

De plus, la projection $(2 : 1)$ sur la ligne de Kummer qui à un point $(X : Y : Z) = (X/Z : Y/Z : 1) = (x, y)$ associe $(X : Z) = (X/Z : 1) = x$ permet de représenter deux points $P = (x_P, y_P)$ et $Q = (x_Q, y_Q)$ seulement grâce à x_P, x_Q et x_{P-Q}. La donné de la différence des deux points permet de différencier P et $-P$. Cette représentation plus compacte sera aussi utile pour accélérer les calculs.

Enfin, il est possible de retrouver rapidement les coefficients de l'équation d'une courbe elliptique à partir de l'abscisse des trois points P, Q et $R = P - Q$. Ainsi, en envoyant $x(\phi_a(P_b), x(\phi_a(Q_b)$ et $x(\phi_a(R)_b)$ (respectivement $x(\phi_b(P_a), x(\phi_b(Q_a)$ et $x(\phi_b(R)_a)$), il n'est plus nécessaire d'envoyer les coefficients de la courbe E_a (respectivement E_b), qui peuvent être retrouvés grâce aux points auxiliaires. De même, la donnée de la courbe initiale n'est pas nécessaire si les points P_a, Q_a, R_a et P_b, Q_b, R_b sont donnés.

Le protocole implémenté en pratique est donc de la forme :

Paramètres publiques : $p = f \cdot \ell_a^{e_a} \ell_b^{e_b} \pm 1$,
$(x(P_a), x(Q_a), x(R_a)), (x(P_b), x(Q_b), x(R_b))$

$\mathcal{A}$lice		$\mathcal{B}$ob
Clé secrète :		**Clé secrète :**
$s_a \in_R \mathbb{Z}/\ell_a^{e_a}\mathbb{Z}$		$s_b \in_R \mathbb{Z}/\ell_b^{e_b}\mathbb{Z}$
Clé publique :		**Clé publique :**
$x(\phi_a(P_b)), x(\phi_a(Q_b)), x(\phi_a(R_b))$		$x(\phi_b(P_a)), x(\phi_b(Q_a)), x(\phi_b(R_a))$
	$x(\phi_a(P_b))$ $x(\phi_a(Q_b))$ $x(\phi_a(R_b))$ $\rightarrow$	
	$\leftarrow$ $x(\phi_b(P_a))$ $x(\phi_b(Q_a))$ $x(\phi_b(R_a))$	
Calcul du secret :		**Calcul du secret :**
$x(G_{ab}) = s_a x(\phi_b(P_a)) + x(\phi_b(Q_a))$		$x(G_{ba}) = s_b x(\phi_a(P_b)) + x(\phi_a(Q_b))$
$E_{ab} = E_b / < G_{ab} >$		$E_{ba} = E_a / < G_{ba} >$
Secret commun :		**Secret commun :**
$j(E_{ab})$		$j(E_{ba})$

3.2 Accélération des calculs

Deux types de calculs doivent être réalisés par Alice et Bob en parallèle pour effectuer l'échange de clés. D'abord le calcul du sous-groupe noyau de l'isogénie, puis le calcul de l'image de la courbe et des points publiques par l'isogénie. Plusieurs techniques sont utilisées pour les réaliser le plus rapidement possible.

3.2.1 Les courbes et l'arithmétique de Montgomery

Les courbes de Montgomery, c'est-à-dire les courbes ayant pour équation $by^2 = x^3 + ax^2 + x$, possèdent une arithmétique efficace, en particulier sur la ligne de Kummer. Elles permettent notamment de calculer $[s_a]P_a + Q_a$ de façon optimisée, et résistante aux attaques par analyse de la consommation de courant. De plus, le choix d'une courbe de cardinal $(p \pm 1)^2$ assure l'existence d'un point d'ordre 4 sur la courbe ou sa tordue, ce qui implique l'existence d'un isomorphisme entre la courbe et une courbe de Montgomery. Comme les isogénies préservent le cardinal des courbes, on peut bien utiliser la représentation de Montgomery tout au long du protocole d'échange.

L'une des particularité des courbes de Montgomery est que leur j-invariant ne dépend pas de b. Ce b n'intervient pas non plus dans les formules pour calculer les images de courbes ou de points par les isogénies. En définitive, n'étant pas utile pour les calculs du protocole, on peut simplement l'ignorer. Une courbe de Montgomery sera donc seulement décrite par un paramètre a.

3.2.2 Décomposition des isogénies

Le calcul des isogénies repose sur l'approche suivante : puisque les isogénies peuvent être décomposées en isogénies de degré premier, les ℓ^e isogénies peuvent être calculées comme une suite de ℓ isogénies. En notant S un point de ℓ^e torsion, le calcul peut-être représenté de la façon suivante :

$$E_0 \xrightarrow{\phi_1} E_0 \Big/ [\ell^{e-1}]S \xrightarrow{\phi_2} \dots \xrightarrow{\phi_i} E_{i-1} \Big/ [\ell^{e-i}]\phi_i(S) \xrightarrow{\phi_{i+1}} \dots \xrightarrow{\phi_e} E_{e_1} \Big/ [\ell]\phi_{e-1}(S) = E_0 / < S >$$

où à chaque étape, $[\ell^{e-i}]\phi_i(S)$ est un point de ℓ torsion sur la courbe E_{i-1}, et où on a identifié ϕ_i comme étant la composée $\phi_i \circ \dots \circ \phi_1$ pour plus de lisibilité. Pour calculer la courbe image, on calcule donc la succession des e courbes images par les ℓ-isogénies. On procède de la même façon pour calculer l'image des points auxiliaires publiques.

Il est important de noter que dans l'étape de création de la clé publique à partir de la clé secrète, les images successives de la courbe initiale E_0 sont nécessaires aux calculs de $[\ell^{e-i}]\phi_i(S)$, bien que la courbe finale obtenue ne soit pas transmise.

A chaque étape i on effectue donc les calculs suivants, qui sont représentés sur le schéma ci-dessous :

- ◇ Calculer $[\ell^{e-1}]S_i$ où $S_i = \phi_i \circ \dots \circ \phi_1(S)$.
- ◇ En déduire $E_{i+1} = E_i \Big/ < [\ell^{e-i}]S_i >$ et ϕ_{i+1}.
- ◇ Calculer $S_{i+1} = \phi_{i+1}(S_i)$.
- ◇ Calculer

$$P_{i+1} = \phi_{i+1}(P_i), Q_{i+1} = \phi_{i+1}(Q_i), R_{i+1} = \phi_{i+1}(R_i) \ ,$$

où

$$P_i = \phi_i \circ \dots \circ \phi_1(P), Q_i = \phi_i \circ \dots \circ \phi_1(Q), R_i = \phi_i \circ \dots \circ \phi_1(R) \ ,$$

avec P, Q est la base des points de torsion, et $R = P - Q$.

$$\begin{array}{ccc} E_i & \xrightarrow{\phi_{i+1}} & E_{i+1} = E_i \Big/ [\ell^{e-i}]S \\ S_i & & \phi_{i+1}(S_i) \\ P_i, Q_i, R_i & & \phi_{i+1}(P_i), \phi_{i+1}(Q_i), \phi_{i+1}(R_i) \end{array}$$

Concrètement, ces calculs reposent sur des formules qui dépendent des nombres premiers ℓ_a et ℓ_b utilisés. Ces formules sont optimisées pour limiter le nombre de calculs, en particulier le nombre d'inversions et de racines carrées dans le corps $\mathbb{F}_{p^2}$ qui sont les plus coûteuses. A cette fin, le paramètre de la courbe de Montgomery a est stocké comme un couple $(A : C)$ tel que $a = A/C$. Cela permet de limiter le nombre d'inversions nécessaires. La même technique est utilisée pour les points $(X : Z)$.

Toujours dans le but d'optimiser les calculs, les courbes sont parfois représentées par $(A + 2C : 4C)$, $(A + 2C : A - 2C)$ voire $(A + 2C/4C : 1)$ plutôt que par $(A : C)$. En effet, certains calculs font appel aux quantités décrites ci-dessus, et il est plus efficace de conserver la courbe sous un tel format, plutôt que de convertir à chaque étape.

3.2.3 L'utilisation de *stratégies*

Dans les faits, les calculs des $[\ell^{e-i}]\phi_i(S)$ successifs sont très gourmands en temps, même avec l'arithmétique de Montgomery. Des *stratégies* peuvent être implémentées pour limiter le nombre de ces calculs, qui utilisent un compromis entre le nombre de multiplications et de calculs d'images de points [8]. Par manque de temps, ces stratégies n'ont pas été implémentées, et ne sont donc pas présentées ici.

4 Choix des nombres premiers

Pour obtenir 128-bits de sécurité sur le protocole, le nombre premier p doit avoir une taille d'au moins 503 bits [10]. Il reste à choisir les nombres ℓ_a et ℓ_b utilisés.

Les formules pour les calculs d'images de points et de courbes par une isogénie dépendent fortement des ℓ_a et ℓ_b choisis, et l'efficacité de l'algorithme dépend en partie de ces formules. Les différents cas possibles sont présentés ci-dessous.

4.1 Le cas des 2-isogénies

Dans le cas de 2^e-isogénies, on calcule plutôt une succession de 4-isogénies à partir des points de 4-torsion. En notant $P_4 = (X_4 : Z_4)$, [8] et [3] donnent les formules suivantes pour calculer les paramètres $(A' : C')$ de la courbe image, et les coordonnées $(X' : Z')$ des points images :

$$(A' : C') = \left(2(2X_4^4 - Z_4^4) : Z_4^4\right)$$

$$(X' : Z') = \big(X\big(2X_4Z_4Z - X(X_4^2 + Z_4^2)\big)(X_4X - Z_4Z)^2 : Z\big(2X_4Z_4X - Z(X_4^2 + Z_4^2)\big)(Z_4X - X_4Z)^2\big)$$

En pratique, le cas $p_a = 2$ est le plus rapide à calculer, car il ne demande que cinq élévation au carré et sept additions $(5S+7a)$ pour le calcul de la courbe, et neuf multiplications, une élévation au carré et six additions $(9M + 1S + 6a)$ pour les images de points.

4.2 Le cas des 3-isogénies

Dans le cas de 3-isogénies, on calcule une succession d'isogénies de degré 3 par les formules de [8] et [3] donnent :

$$(A' : C') = \big(Z_3^4 + 18X_3^2Z_3^2 - 27X_3^4 : 4X_3Z_3^3\big)$$

$$(X' : Z') = \big(X(X_3X - Z_3Z)^2 : Z(Z_3X - X_3Z\big)^2\big)$$

Ces formules peuvent être calculées en trois multiplications, trois élévations au carré et huit additions $(3M + 3S + 8a)$ pour la courbe, et six multiplications, deux élévations au carré et deux additions $(6M + 2S + 2a)$ pour les points.

En pratique, le choix de nombres premiers de la forme $2_2^e3_3^e \pm 1$ est le plus efficace, et c'est celui qui est utilisé dans le protocole SIKE (*supersingular isogeny key exchange*) [10].

4.3 Le cas général des isogénies de degré impair

Craig Costello et Hüseyin Hisil ont récemment démontré dans [2] une formule pour le calcul d'isogénies de degré impair qui ne dépend que des abscisses des points du noyau de l'isogénie.

Le noyau d'une isogénie de degré impair contient $2d + 1$ points. En réalité, l'un de ces points est l'infini $(0 : 1 : 0)$, et les autres sont identifiés deux à deux sur la ligne de Kummer, puisque $P = -P$. Ainsi, seuls d points sont nécessaires pour décrire le noyau.

Pour les corps de caractéristique différente de 2, et en notant $P = (X_P : Z_P)$ le générateur du noyau de l'isogénie, l'image $(X' : Z')$ d'un point $(X : Z)$ est ([2] théorème 1) :

$$(X' : Z') = \left(X \prod_{i=1}^{d} \left(\frac{X \cdot X_{[i]P} - 1}{X - X_{[i]P}}\right)^2 : 1\right)$$

Un aspect intéressant des isogénies impaires est que les points de deux torsion sont conservés après application d'une isogénie, c'est-à-dire que $[2]\phi((X_T : Z_T)) = 0$ si $[2]T = 0$. De plus, les paramètres de la courbes peuvent être retrouvés à partir d'un point de deux torsion en utilisant :

$$(a:1) = (A:C) = (X_T^2 + Z_T^2 : -X_T Z_T)$$

où $T = (X_T, Z_T)$ est un point de deux torsion différent de $(0:1)$.

Ainsi il n'est plus nécessaire de calculer l'image de la courbe, il suffit de calculer l'image d'un point de deux torsion par la formule précédente, puis de retrouver la courbe correspondante.

Afin d'optimiser les calculs, il est plus utile de calculer les paramètres de la courbe sous la forme $(A+2C:4C)$, car c'est cette forme qui est utilisée pour accélérer les multiplications de points. En pratique on utilise donc :

$$(A+2C:4C) = \left((X_T - Z_T)^2 : (X_T - Z_T)^2 - (X_T + Z_T)^2\right)$$

qui ne coûte que deux élévations au carré et trois additions $(2S + 3a)$.

Cette nouvelle formule ainsi que l'utilisation des points de deux torsion rend l'utilisation de nombres premiers autres que 2 et 3 envisageable. L'un des buts de ce projet était de comparer les performances de premiers impairs avec l'utilisation usuelle de 2 et 3. L'implémentation est les résultats sont décrits dans la section suivante.

5 Implémentation

Le code contient une implémentation du protocole SIKE basée sur la spécification correspondante [10] utilisant les nombres premiers 2 et 3, ainsi qu'une généralisation du protocole à tout entier impair implémentée à partir des travaux de Craig Costello et Hüseyin Hisil [1].

Le code a été réalisé en C sur Ubuntu 16.04.03, en utilisant la bibliothèque gmp. Le fichier README inclus explique comment le compiler à partir du Makefile.

Cette section donne d'abord une description succincte de la structure du programme et de ses différentes composantes, puis présente les résultats obtenus, avant de revenir sur certaines difficultés rencontrées.

5.1 Structure du programme

La structure du code est la suivante :

- ◇ Le fichier *Header.h* contient les déclarations et les descriptions des fonctions utilisées dans le projet.
- ◇ Le fichier *fp.c* contient les fonctions nécessaires pour allouer, manipuler et désallouer les éléments de $\mathbb{F}_p^2$. Ces éléments sont représentés comme $s_0 + is_1$, où $i^2 = -1$.
- ◇ Le fichier *curve_point.c* contient les fonctions nécessaires pour allouer, manipuler et désallouer les courbes et les points. Les courbes sont représentées comme $(A:C), A, C \in \mathbb{F}_p^2$, tel que A/C soit le coefficient a d'une courbe de Montgomery. Les points sont représenté comme $(X:Z)$ sur la ligne de Kummer.

1. Le code est disponible à l'adresse : `/github.com/MathildeChenuLM/SIKE_generalized/`

- ◇ Le fichier *montgomery.c* contient les fonctions implémentant l'arithmétique efficace sur les courbes de Montgomery.
- ◇ Le fichier *isogeny.c* contient les fonctions utilisées dans le calcul d'image de courbes et de points par des isogénies de degré 2, 3, et de degré impair quelconque.
- ◇ Le fichier *pk_sk_param.c* contient les fonctions nécessaires pour allouer, manipuler et désallouer les structures définissant les clés publiques, privées, et les paramètres publiques. Il contient également les fonctions pour établir la clé publique à partir de la clé privée, et les fonctions implémentant l'échange de clés.
- ◇ Le fichier *Main.c* contient les fonctions encapsulant la création et l'échange de clés. Ce fichier contient également un ensemble de paramètres possibles en fonction des nombres premiers utilisés, et un test de vérification du protocole d'échange de clés.
- ◇ Le fichier *Makefile* permet de compiler l'ensemble des fichiers précédents, tout en les profilant en vue de l'analyse de performances.

Chacun de ces fichier contient également un jeux de tests inclus dans une fonction *main* qui peut être exécutée pour vérifier les résultats de chaque fonction. Ces jeux de tests ont été en grande partie réalisés avec Sage afin de vérifier les calculs.

5.2 Résultats

Les temps d'exécution pour la création de la clé publique et l'échange de clés ont été mesurés par profilage du code pour des nombres premiers de 503 bits (pour 128 bits de sécurité) issus de tous les couples de nombres premiers impairs entre 3 et 19, puis comparés au cas $2, 3$.

Ces nombres premiers ont été généré en utilisant Sage, de façon à obtenir $\ell_a^{e_a} \simeq \ell_b^{e_b}$, et $p = f \cdot \ell_a^{e_a} \ell_b^{e_b} - 1$, tel que $p = -1 \mod 4$. Cela assure que la courbe $y^2 = x^3 + x$ soit supersingulière, et de cardinal $(p+1)^2$, conformément au protocole.

Les bases de $\ell_a^{e_a}$ et $\ell_b^{e_b}$ torsion respectivement ont ensuite été générés [2] en s'inspirant largement du code Sage disponible à l'adresse : **github.com/defeo/ss-isogeny-software** [7].

Les résultats obtenus sont présentés sur les graphiques suivants.

Le premier tableau donne les temps obtenus en profilant la création de clé, l'échange de clés, et le protocole dans son intégralité, en fonction des nombres premiers utilisés. Pour la création de clés, le temps mesuré est celui nécessaire à la création de la clé d'Alice et de Bob. Pour l'échange de clés en revanche, le temps calculé est une moyenne entre le temps nécessaire à Alice ou à Bob.

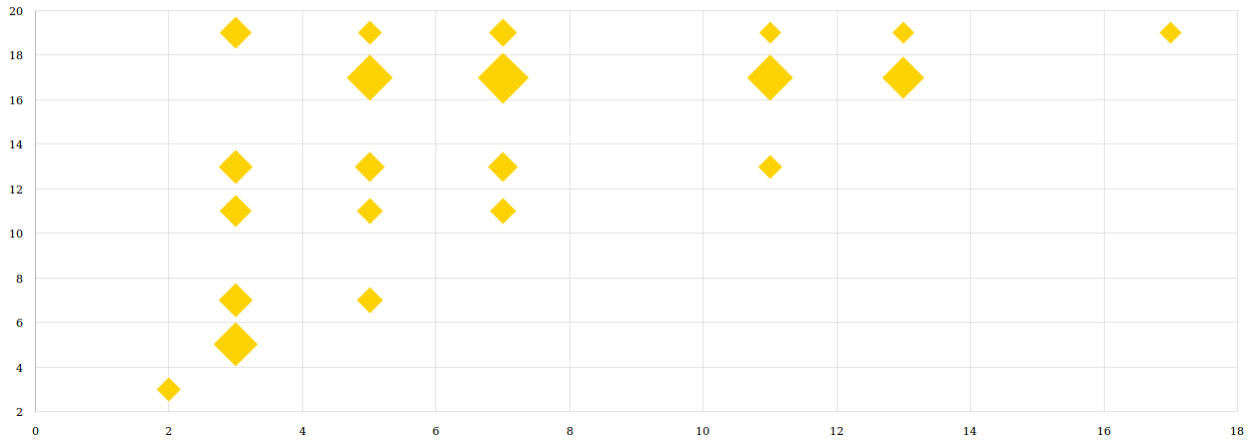
2. Sauf pour les couples de nombres premiers $(3, 11)$ et $(7, 13)$, pour lesquels le code Sage utilisé n'a pas trouvé les points de torsion nécessaire.

Durées de la création de clé (k.g.), de l'échange de clé (k.e.) et de l'ensemble du protocole en fonction de
Les résultats sont exprimés en secondes.

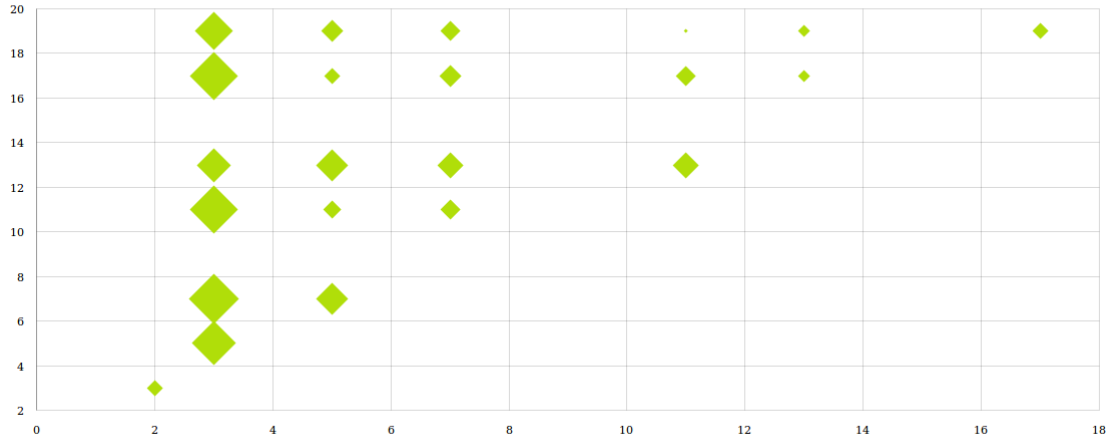
	3			5			7			11			13			17			19		
	k.g.	*k.e.*	*tot.*	*k.g.*	*k.e.*	*tot.*	*k.g.*	*k.e.*	*tot.*	*k.g.*	*k.e.*	*tot.*	*k.g.*	*k.e.*	*tot.*	*k.g.*	*k.e.*	*tot.*	*k.g.*	*k.e.*	*tot.*
2	0.4	0.18	0.76																		
3				0.74	0.29	1.31	0.58	0.31	1.19	0.54	0.3	1.13	0.59	0.25	1.09	0.65	0.3	1.24	0.53	0.26	1.05
5							0.46	0.24	0.94	0.43	0.19	0.81	0.52	0.24	1	0.42	0.18	0.78	0.41	0.21	0.82
7										0.44	0.2	0.83	0.51	0.22	0.95	0.46	0.21	0.87	0.49	0.2	0.88
11													0.41	0.22	0.85	0.41	0.2	0.8	0.38	0.14	0.66
13																0.37	0.17	0.71	0.36	0.17	0.7
17																			0.39	0.18	0.74
19																					

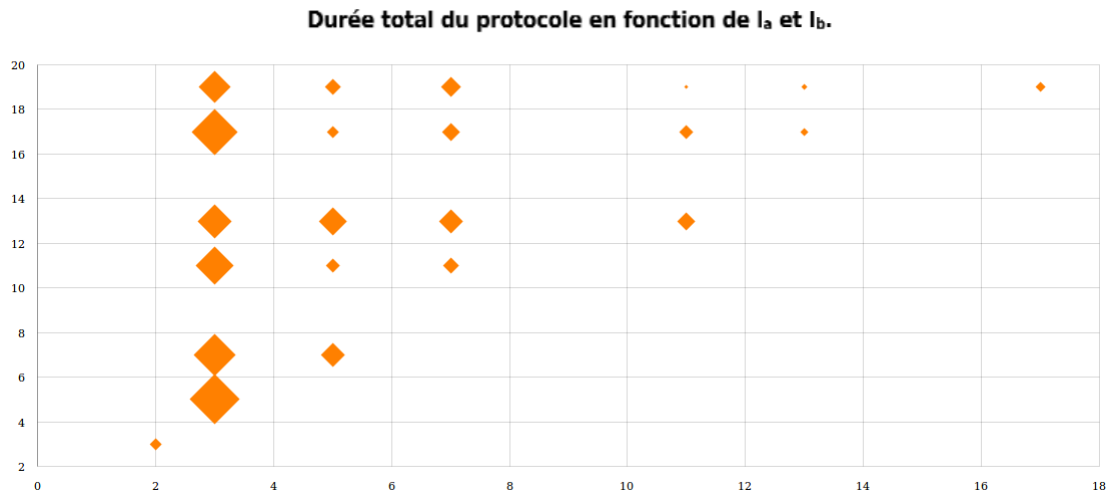
Les graphiques suivants donnent une comparaison de ces différents temps à l'aide de diagrammes à bulles. La taille des points aux coordonnées (ℓ_a, ℓ_b) est proportionnelle au temps nécessaire pour la création de clés, l'échange de clés et la réalisation de l'ensemble du protocole respectivement. Cela permet d'avoir un aperçu visuel de la variation des performances en fonction du nombre premier utilisé.

Durée de la création de clé publique en fonction de l_a et l_b.



Durée de l'échange de clé en fonction de l_a et l_b.





Comme attendu, les nombres premiers 2 et 3 font partie des plus performants, même en l'absence de stratégie. En revanche, contrairement à ce qu'on pourrait attendre, les performances s'améliorent globalement lorsque les nombres premiers utilisés augmentent. Cela pourrait provenir du fait que l'exposant e diminue lorsque ℓ augmente, à nombre de bits équivalent, et donc que le nombre de ℓ-isogénies à calculer diminue quand ℓ augmente.

On remarque aussi que les durées de création de clés et d'échange de clés ne sont pas proportionnelles. Certains couples de nombres premiers, comme 5 et 17 ont une création de clés très lente relativement aux autres couples, mais un échange de clés plutôt plus rapide. Comme l'échange de clés demande moins de calculs d'images de points que la création, cela met en relief les coûts variables des calculs d'images points par rapport aux calculs d'images courbes en fonction des nombres premiers choisis, et l'amélioration des performances qui peut être apportée par l'utilisation de stratégies.

Enfin, l'amélioration des performances lorsque les premiers augmentent n'est pas tout à fait linéaire. On peut noter le cas du couples $(11, 19)$ qui donne des résultats très performants, que ce soit pour la création ou l'échange de clés.

Rappelons que ces résultats ont été obtenu sans utiliser de stratégie optimale, et donnent donc une idée des résultats en pire cas. L'utilisation de stratégies a de fortes chances de changer les valeurs présentés ici. De plus, il est possible que ces résultats puissent varier en fonction du nombre premier $p = f \cdot \ell_a^{e_a} \ell_b^{e_b} \pm 1$, même à ℓ_a et ℓ_b fixés. D'autres tests sont nécessaires pour clarifier ce point.

On peut mentionner que Joppe Bos and Simon Friedberger ont obtenu dans [1] une amélioration significative en utilisant les premiers 2 et 19. Comme le code ne concerne que les isogénies de degré impairs, cette situation n'a pu être reproduite ici.

5.3 Difficultés rencontrées

Les paramètres du problème de l'échange de clés sont représentés par des structures complexes en C reposants sur gmp. L'allocation et la libération de la mémoire sont donc également complexes, ce qui m'a permis d'être, à terme, beaucoup plus à l'aise avec la gestion de la mémoire et des pointeurs en C.

La complexité des formules utilisées dans les calculs demande d'avoir recours à Sage, ou tout autre logiciel de calcul, pour générer des tests afin de vérifier que le code s'exécute correctement. Cela m'a appris les subtilités d'utilisation des corps finis et des courbes elliptiques sur Sage, notamment l'absence de fonctions relatives aux courbes de Montgomery et l'importance du polynôme modulaire dans la définition d'un corps fini.

Les paramètres publiques utilisés dans le protocole d'échange de clés sont très précis, et dépendent des nombres premiers utilisés. Ils demandent d'avoir un nombre premier de la forme $f \cdot \ell_a^{e_a} \ell_b^{e_b} - 1$, une courbe de cardinal précis $(p-1)^2$, et une base des points de ℓ^e torsion. Ces paramètres sont fournis par la spécification dans les cas des premiers 2 et 3, mais doivent être générés avant de pouvoir tester le cas des isogénies impaires. Sage permet à nouveau de pallier cette difficulté, à condition d'avoir les bons algorithmes. Une adaptation Sage du code Python/Cython/C [**?**] m'a permis de générer efficacement ces paramètres, et d'avoir un aperçu de l'utilisation de Cython.

Enfin, l'installation d'Ubuntu en vue de l'utilisation de gmp m'a aussi familiarisée avec les BIOS, les relations de voisinages avec Windows, les drivers wifi et les caprices de clés USB. Bien qu'éloignée de la programmation C, ce fut aussi une expérience intéressante.

Conclusion

Ce projet a été l'occasion d'étudier en détails et d'implémenter un protocole d'échange de clés utilisant les graphes d'isogénies supersingulières, d'abord à partir de la spécification [10], puis à partir de l'article de [2] dans un cas plus général.

En comparant les deux approches, il apparaît bien que les premiers 2 et 3 sont ceux qui donnent les calculs les plus rapides. Après une chute des performances pour les petits premiers impairs, elles s'améliorent ensuite globalement lorsque des nombres premiers ℓ_a et ℓ_b augmentent pour rejoindre des temps comparables à 2 et 3.

En revanche, l'absence d'utilisation de stratégies dans l'implémentation ne permet pas de quantifier la pénalité imposé par l'utilisation de premiers arbitrairement plus grands en situation réelle.

Références

[1] Joppe W. Bos and Simon Friedberger. Fast arithmetic modulo 2xpy± 1. In *IACR Cryptology ePrint Archive*, 2016.

[2] Craig Costello and Hüseyin Hisil. A simple and compact algorithm for SIDH with arbitrary degree isogenies. In *Advances in Cryptology - ASIACRYPT 2017, Proceedings*.

[3] Craig Costello, Patrick Longa, and Michael Naehrig. Efficient algorithms for supersingular isogeny diffie-hellman. In *Advances in Cryptology - CRYPTO 2016 Proceedings*, 2016.

[4] Craig Costello and Benjamin Smith. Montgomery curves and their arithmetic. *CoRR*, 2017.

[5] Jean Marc Couveignes. Hard homogeneous spaces. *IACR Cryptology ePrint Archive*, 2006.

[6] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theor.*, September 2006.

[7] Luca De Feo. https ://github.com/defeo/ss-isogeny-software.

[8] Luca De Feo, David Jao, and Jérôme Plût. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. *J. Mathematical Cryptology*, 2014.

[9] Steven D. Galbraith, Christophe Petit, Barak Shani, and Yan Bo Ti. On the security of supersingular isogeny cryptosystems. In *Advances in Cryptology - ASIACRYPT 2016*, 2016.

[10] David Jao, Reza Azarderakhsh, Matthew Campagna, Craig Costello, Luca De Feo, Basil Hess, Amir Jalali, Brian Koziel, Brian LaMacchia, Patrick Longa, Michael Naehrig, Joost Renes, Vladimir Soukharev, and David Urbanik. Supersingular isogeny key exchange, 2017. https ://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-1-Submissions.

[11] David Jao and Luca De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011*, 2011.

[12] David Kohel. Endomorphism rings of elliptic curves over finite fields. 1996.

[13] Peter L. Montgomery. Speeding the pollard and elliptic curve methods of factorization. 48, 1987.

[14] Arnold Pizer. Ramanujan graphs. 1998.

[15] Joseph H. Silverman. *The Arithmetic of Elliptic Curves*.