Projet PX511 :

One round tripartite non-interactive key exchange

Sommaire

I Problématique

En cryptographie, l'échange chiffré entre deux parties peut être réalisé grâce à une clé de session partagée, établie par l'algorithme de Diffie–Hellman (DH). En revanche, avec trois parties ou plus, un émetteur devrait, en l'absence d'un protocole adapté, établir des clés bilatérales distinctes avec chaque destinataire, ce qui multiplie le nombre de clés et de négociations. Dans ce projet, nous visons à réaliser un échange de clé à trois participants, permettant à Alice, Bob et Charlie de dériver la même clé secrète K sur un canal non sécurisé.
Le but de ce projet est de fournir un démonstrateur qui choisit des clés aléatoires et vérifie, du point de vue de chacun des 3 participants que la clé obtenue soit identique.

1) Principe de la cryptographie

Pour commencer, définissons la cryptographie :
La cryptographie est une discipline ayant pour but de protéger des messages (assurant confidentialité, authenticité et intégrité) en s'aidant souvent de secrets ou clés. Elle permet de rendre un message supposément inintelligible à qui que ce soit autre ne possédant pas le moyen de le décrypter.

Désormais, présentons et expliquons pourquoi la cryptographie est omniprésente en informatique aujourd'hui.

a. Utilité

De nos jours, la cryptographie sert principalement à la protection des données.
En effet, elle intervient particulièrement lors d'échanges de données ou de messages entre plusieurs individus sur des canaux ouverts et publiques. Cela permet ainsi, de garantir plusieurs propriétés essentielles : la confidentialité, l'intégrité et l'authenticité.

Concrètement, de nombreux outils cryptographiques sont employés dans des usages quotidiens. Les fonctions de hachage cryptographiques (telles que SHA-256) sont utilisées pour vérifier l'intégrité des données, stocker des mots de passe ou identifier de manière unique un contenu. Le chiffrement symétrique (par exemple AES) permet de protéger le contenu des communications ou des fichiers stockés.

Pour mieux comprendre notre sujet, nous allons rapidement présenter les 2 types d'échanges chiffrés qui sont réalisés dans nos communications. Le premier est asymétrique, utilisé pour l'échange de clé secrète, le plus critique et celui permettant ensuite de passer en échange symétrique. Le second est le symétrique, utilisé une fois la clé secrète connue, qui chiffre simplement le message à l'aide de cette clé connue des communicateurs.

b. Échange symétrique

La cryptographie symétrique repose sur l'utilisation d'une clé unique partagée entre les participants. Cette clé sert à la fois au chiffrement et au déchiffrement des messages. Les algorithmes symétriques sont en général très efficaces et bien adaptés au chiffrement de volumes importants de données.

Cependant, le principal défi de ce modèle réside dans le partage initial de la clé secrète : si un adversaire parvient à intercepter cette clé, la sécurité de toutes les communications est compromise. Pour cette raison, la cryptographie symétrique est rarement utilisée seule dans des contextes ouverts.

En pratique, les systèmes modernes combinent cryptographie asymétrique et symétrique. Un protocole d'échange de clé (tel que Diffie–Hellman) permet d'établir de manière sécurisée une clé symétrique partagée, qui sera ensuite utilisée pour protéger les échanges de données de façon efficace.

c. Échange asymétrique

A l'inverse, la cryptographie asymétrique, aussi appelée cryptographie à clé publique, repose sur l'utilisation d'une paire de clés distinctes : une clé publique et une clé privée. La clé publique peut être diffusée librement, tandis que la clé privée doit rester secrète et n'être connue que de son propriétaire.

Dans un schéma classique de chiffrement asymétrique, un participant chiffre un message à l'aide de la clé publique du destinataire. Seul ce dernier est alors capable de déchiffrer le message à l'aide de sa clé privée

correspondante. Cette approche permet d'établir une communication sécurisée sans qu'un secret préalable n'ait été partagé entre les parties.

2) Notions Mathématiques

Les protocoles d'échange de clés étudiés dans ce rapport reposent sur des structures algébriques bien définies. Cette section introduit les notions mathématiques essentielles nécessaires à la compréhension de Diffie–Hellman, de ses variantes sur courbes elliptiques, et des constructions plus avancées basées sur des couplages.

a. Groupes et groupes cycliques

Un groupe est un ensemble G muni d'une opération interne $\circ : G \times G \to G$ vérifiant les propriétés suivantes :

- associativité :
$$\forall x, y, z \in G,$$
$$(x \circ y) \circ z = x \circ (y \circ z)\ ;$$
- existence d'un élément neutre :
$$\forall x \in G, e \in G$$
$$\text{tel que } x \circ e = e \circ x = x\ ;$$
- existence d'un inverse :
$$\forall x \in G, \exists x^{\{-1\}} \in G$$
$$\text{tel que } x \circ x^{\{-1\}} = e\ ;$$
- clôture :
$$\forall x, y \in G,$$
$$x \circ y \in G\ .$$

Un groupe est dit abélien si l'opération est commutative, c'est-à-dire

$$x \circ y = y \circ x$$

pour tous $x, y \in G$.

Un groupe G est cyclique s'il existe un élément $g \in G$, appelé générateur, tel que

$$G = \langle g \rangle = \{g^k \mid k \in \mathbb{Z}\}$$

Dans ce cas, tout élément du groupe s'exprime comme une puissance (ou itération) du générateur.

Les groupes cycliques finis sont centraux en cryptographie car ils permettent de définir des problèmes asymétriques : certaines opérations y sont faciles à calculer, tandis que leurs inverses sont supposées difficiles.

b. Groupe multiplicatif $\mathbb{Z}_p^*$

Un exemple fondamental est le groupe multiplicatif

$$\mathbb{Z}_p^* = \{1, 2, ..., p-1\}$$

, où p est un nombre premier, muni de la multiplication modulo p

Ce groupe est fini, abélien et cyclique. Il existe donc un générateur $g \in \mathbb{Z}_p^*$ tel que

$$\mathbb{Z}_p^* = \{g^k \bmod p \mid k \in \{0, ..., p-2\}\}$$

Dans ce contexte, l'opération directe est l'exponentiation modulaire

$$g^a \bmod p$$

qui se calcule efficacement. En revanche, l'opération inverse — retrouver a à partir de g et $g^a \bmod p$ — correspond au problème du logarithme discret, supposé difficile lorsque p est suffisamment grand.

3) Courbes elliptiques

Les courbes elliptiques fournissent des groupes abéliens finis utilisés en cryptographie à base de logarithme discret. Soit $\mathbb{F}_q$ un corps fini (avec $q = p^m$) et car $\left(\mathbb{F}_q\right) \neq 2, 3$. Une courbe elliptique E est donnée par une équation de Weierstrass courte $y^2 = x^3 + ax + b$, où la condition de non-singularité $\Delta \neq 0$ garantit l'absence de points singuliers. L'ensemble

$$E\left(\mathbb{F}_q\right) = \left\{(x, y) \in \mathbb{F}_q^2 \mid y^2 = x^3 + ax + b\right\} \cup \{O\}$$

est muni d'une loi de groupe abélienne (addition de points) avec O comme élément neutre. En pratique, les primitives cryptographiques se placent dans un sous-groupe cyclique $< P >$ d'ordre n (souvent premier) et de cofacteur h, ce qui structure précisément le problème difficile exploité.

L'opération « directe » est la multiplication scalaire $[k]P$, calculée efficacement en $O(\log k)$ par des algorithmes comme double-and-add ou Montgomery ladder. L'opération inverse consiste à retrouver k à partir de P et $Q = [k]P$: c'est le logarithme discret sur courbe elliptique (ECDLP), supposé difficile pour des courbes correctement choisies. Introduite par Miller et Koblitz, la cryptographie ECC se décline en accord de clé (ECDH/ECDHE) et en signatures (ECDSA, EdDSA) [Kob87, Mil86].

L'avantage principal est un meilleur compromis sécurité/performance : à niveau de sécurité comparable, les paramètres ECC sont plus compacts que ceux de RSA et de certains groupes multiplicatifs, réduisant bande passante et coût de calcul. Pour des courbes recommandées, les meilleures attaques génériques connues sur l'ECDLP sont de type Pollard-rho et ont une complexité en $O(\sqrt{n})$, ce qui motive le choix d'un ordre n suffisamment grand et de paramètres standardisés ; les recommandations NIST [Bar20, Che+23] cadrent précisément ces tailles et paramètres de domaine.

II État de l'art : échange de clé tripartite

L'échange de clé est une brique fondamentale de la cryptographie moderne : il permet à plusieurs participants de se mettre d'accord sur une clé secrète partagée malgré une communication sur un canal potentiellement contrôlé par un adversaire. Historiquement, l'un des premiers mécanismes sûrs pour deux participants est l'échange de clé de Diffie–Hellman, proposé en 1976, qui permet à deux entités d'établir une clé conjointe sans secret préalable partagé [DH06] ; ce mécanisme sert encore de base à de nombreux protocoles standards.

Comme il est important de comprendre comment fonctionne l'échange de clé selon Diffie-Hellman dans le domaine de la cryptographie et pour la suite de notre rapport, voici son fonctionnement :

1) Présentation du protocole Diffie Hellman

L'échange de clé de Diffie–Hellman (DH) est un mécanisme permettant à deux participants (Alice et Bob) de dériver une même clé de session K en utilisant un canal public, sans avoir à partager un secret au préalable. Historiquement, DH est l'une des primitives les plus connues et les plus utilisées depuis sa proposition initiale (1976) [DH06].

Dans sa forme « classique », les deux parties se mettent d'abord d'accord sur des paramètres publics :
Un groupe cyclique G d'ordre q (par exemple un sous-groupe de $\mathbb{Z}_p{}^*$ ou un groupe de points sur une courbe elliptique), ainsi qu'un générateur g.
Alice tire un aléa $a \xleftarrow{\$} \{1, \ldots, q-1\}$ et envoie $A = g^a$;
Bob tire $b \xleftarrow{\$} \{1, \ldots, q-1\}$ et envoie $B = g^b$.
Chacun calcule alors le même secret Z via $Z = B^a = \left(g^b\right)^a = g^{ab}$ côté Alice et $Z = A^b = \left(g^a\right)^b = g^{ab}$ côté Bob.
La sécurité est généralement formulée via des hypothèses calculatoire : (i) CDH (Computational Diffie–Hellman), qui postule qu'il est difficile de calculer g^{ab} à partir de $\left(g, g^a, g^b\right)$, et (ii) DDH (Decisional Diffie–Hellman), qui postule qu'il est difficile de distinguer g^{ab} d'un élément aléatoire du groupe.

En pratique, Z n'est pas utilisé directement comme clé : il sert de matériau d'entrée à une fonction de dérivation (KDF) [KE10] qui produit une ou plusieurs clés de session, en liant le résultat au contexte (identités, transcripts, paramètres négociés) afin d'éviter des confusions de clés et de garantir des propriétés attendues par les standards. Les recommandations NIST [Bar+18] sur l'établissement de clé par logarithme discret décrivent précisément ces schémas et les exigences associées (choix de groupe, validation de clés publiques, dérivation, etc.).

Enfin, DH « nu » n'authentifie pas les parties : sans authentification (signature, PSK, certificat…), il est vulnérable à l'attaque de l'homme du milieu. C'est pourquoi les protocoles déployés (p. ex. TLS 1.3) utilisent des variantes éphémères (DHE/ECDHE) combinées à l'authentification, ce qui apporte notamment la confidentialité persistante (forward secrecy) : la compromission ultérieure d'une clé long-terme ne permet pas de retrouver les clés de session passées [Res18].

2) Approches classiques pour plusieurs participants

Dans un contexte multipartite, on peut étendre le protocole Diffie–Hellman à trois parties en effectuant plusieurs tours d'échange entre les participants. Par exemple, la simple itération du protocole entre les trois utilisateurs peut fournir un secret partagé après plusieurs échanges intermédiaires de valeurs publiques. Cependant, ces constructions nécessitent typiquement plusieurs tours de communication et sont inefficaces en termes de latence lorsqu'on cherche à faire coïncider rapidement plusieurs clés individuelles.

Les méthodes basées uniquement sur la généralisation du schéma Diffie–Hellman aux trois parties n'offrent pas toujours un chemin direct vers une clé commune en un seul tour de communication. De plus, elles reposent sur des hypothèses comme la difficulté des problèmes de type Diffie–Hellman dans un groupe donné, ce qui impose des contraintes sur les performances et sur l'échelonnage du nombre de participants.

C'est dans ce contexte qu'émerge une nouvelle classe de protocoles exploitant des objets mathématiques plus puissants — les accouplements bilinéaires (pairings) — qui permettent simplement et directement de concevoir un partage de clé multipartite en un seul tour.

3) Cryptographie à base de couplages (pairings)

La cryptographie à base de couplages désigne l'utilisation de **fonctions bilinéaires non dégénérées** sur des groupes abéliens finis pour construire des primitives cryptographiques inhabituelles ou plus efficaces.

Un couplage cryptographique est une application $[e : G_1 \times G_2 \to G_T]$ entre trois groupes cycliques G_1, G_2 et G_T, telle que :

- e est bilinéaire, c'est-à-dire

[$e\big(g_1^a, g_2^b\big) = e(g_1, g_2)^{\{ab\}}$] pour tous entiers a, b et éléments $g_1 \in G_1, g_2 \in G_2$;

- e est non dégénérée : si g_1 et g_2 sont des générateurs, alors $e(g_1, g_2)$ est un générateur de G_T.

Cette bilinéarité donne une forme de multiplicativité croisée difficile à obtenir avec des opérations de groupe seules.

Plusieurs types de pairings sont utilisés en pratique :

- **Weil pairing** : défini à partir de fonctions rationnelles sur les points d'une courbe elliptique ;
- **Tate pairing** : généralement préféré en cryptographie pour son efficacité pratique et ses propriétés bilinéaires ;
- **Ate & variants** : modifications du Tate pairing optimisées pour des calculs plus rapides sur certaines classes de courbes (par exemple, l'Optimal Ate pairing).

Les méthodes de calcul de ces pairings reposent sur des algorithmes comme celui de Miller, qui permettent d'évaluer ces couplages efficacement sur des courbes elliptiques définies sur des corps finis.

Voici en détail l'explication mathématique de chacun des couplages :

(i) Weil

Le couplage de Weil est défini sur le r-torsion complet d'une courbe elliptique E : $e_{Weil} : E[r] \times E[r] \to \mu_r \subset \mathbb{F}_{p^k}^*$. Il provient de fonctions rationnelles dont les diviseurs sont multiples de r (idée : construire une fonction f_r, P de diviseur $r(P) - r(O)$ et l'évaluer "autour" de Q). Il est bilinéaire, alterné, et non dégénéré dès que les deux arguments appartiennent à $E[r]$. En pratique, on le calcule via l'algorithme de Miller (boucle de longueur $\simeq \log r$, composée d'additions/doublages et d'évaluations de droites) suivi d'une exponentiation de normalisation.

Sur le plan performance, le Weil est historiquement important mais plus coûteux : il exige typiquement que les deux points vivent dans l'extension $\mathbb{F}_{p^k}^*$ (ou sur le tordu) et la normalisation n'est pas plus simple que pour Tate/Ate. On le rencontre donc surtout à des fins pédagogiques ou pour des preuves. Pour notre NIKE tripartite, il suffit d'avoir un couplage bilinéaire non dégénéré ; le Weil fonctionne, mais n'est pas le meilleur choix en efficacité.

(ii) Tate

Le couplage de Tate (réduit) s'écrit $e_{Tate}(P, Q) = f_{r,P}(Q)^{\frac{p_k - 1}{r}}$, avec $P \in E[r]$ et $Q \in E\big(\mathbb{F}_{p^k}\big)$.
L'évaluation brute $f_{r,P}(Q)$ (Miller) n'est définie qu'à une puissance r —ième près ; l'exponentiation finale $\frac{p_k - 1}{r}$ projette dans μ_r et rend la valeur unique. Par rapport au Weil, le Tate est souvent plus rapide et plus souple : on place P et Q dans des sous-groupes G_1, G_2 convenables (Type-3 "asymétrique"), par exemple $P \in E\big(F_p\big)$ et Q sur un tordu défini sur $F_{p^{\frac{k}{d}}}$, ce qui réduit les coûts d'arithmétique.

Côté sécurité, la bilinéarité $e(aP, bQ) = e(P, Q)^{\{ab\}}$ et la non-dégénérescence tiennent sous des conditions standard : cofactor-clearing et choix d'un degré d'immersion k tel que $rm\,\mathrm{id}\big(p^k - 1\big)$. Le couplage de Tate couvre parfaitement l'usage NIKE tripartite de Joux (2000) [Jou04] — première construction "une ronde" basée sur un couplage ; on peut prendre $K_A = e(B_1, C_2)^a$, $K_B = e(C_1, A_2)^b$,

$K_C = e(A_1, B_2)^c$ (voir aussi la référence ajoutée en bibliographie). Il reste le coût de l'exponentiation finale, souvent décomposée en partie "facile" (facteurs cyclotomiques) et "difficile" (exposant résiduel).

(iii) Ate

Le couplage d'Ate accélère la phase Miller en remplaçant le paramètre r par un petit entier lié à la trace de Frobenius. Sur une courbe ordinaire (BN, BLS), si t est la trace, on pose $T = t - 1$ et on définit (schématiquement) $e_{Ate}(Q, P) = f_{T,Q}(P)^{\frac{p^k-1}{r}}$. La boucle de Miller parcourt alors $\log T$ au lieu de $\log r$, ce qui apporte un gain net quand $\mid T \mid \ll r$ (cas des courbes "pairing-friendly"). Les arguments sont "inversés" par rapport à Tate (on évalue une fonction attachée à Q sur P), mais l'objet final vit toujours dans μ_r.

Des variantes modernes — R-ate, optimal Ate, optimal-Ate sur BN/BLS12 — combinent plusieurs itérations liées au Frobenius pour raccourcir encore la boucle, tout en conservant la même exponentiation finale (optimisée par décomposition algébrique). En pratique, les bibliothèques actuelles (bplib, mcl, relic, etc.) implémentent un optimal-Ate Type-3 sur BN/BLS, qui est aujourd'hui le meilleur compromis performance/simplicité. Pour notre NIKE, l'Ate (ou optimal-Ate) est le choix recommandé : même garantie de bilinéarité/non-dégénérescence, mais temps de calcul sensiblement moindre que Tate/Weil.

4) Théorie pour l'échange de clé en tripartite (Joux)

Le protocole proposé par Joux constitue une généralisation directe de l'échange de clé de Diffie–Hellman à trois participants, permettant à Alice, Bob et Charlie d'établir une clé secrète commune en un seul tour de communication. Cette propriété est rendue possible grâce à l'utilisation de couplages bilinéaires définis sur des groupes issus de courbes elliptiques.

a. Cadre mathématique

Soient :

- G_1 et G_2 deux groupes cycliques additifs d'ordre premier q ;
- G_T un groupe cyclique multiplicatif d'ordre q ;
- $e : G_1 \times G_2 \to G_T$ un couplage bilinéaire non dégénéré.

Dans la plupart des constructions pratiques, on considère le cas symétrique où $G_1 = G_2$, ce qui simplifie l'implémentation sans perte de généralité.

La bilinéarité du couplage implique que, pour tout $P, Q \in G_1$ et pour tous $a, b \in \mathbb{Z}_q$: [e(aP, bQ) = e(P, Q)^{a b}.]

b. Description du protocole

Les trois participants se mettent d'accord sur des paramètres publics :

- un groupe elliptique G d'ordre q ;
- deux points indépendants $P, Q \in G$;
- un couplage bilinéaire $e : G \times G \to G_T$.

Chaque participant choisit ensuite un secret aléatoire :

- Alice tire $a \xleftarrow{\$} \mathbb{Z}_q$;
- Bob tire $b \xleftarrow{\$} \mathbb{Z}_q$;
- Charlie tire $c \xleftarrow{\$} \mathbb{Z}_q$.

Ils calculent alors leurs valeurs publiques respectives :

- Alice envoie (aP, aQ) ;
- Bob envoie (bP, bQ) ;
- Charlie envoie (cP, cQ).

Cette diffusion constitue l'unique tour de communication du protocole.

c. Calcul de la clé partagée

À partir des valeurs reçues, chaque participant peut calculer la clé secrète commune :

- Alice calcule :

$$K = e(bP, cQ)^a = e(P, Q)^{\{abc\}}.$$

- Bob calcule :

$$K = e(aP, cQ)^b = e(P, Q)^{\{abc\}}.$$

- Charlie calcule :

$$K = e(aP, bQ)^c = e(P, Q)^{\{abc\}}.$$

Par bilinéarité du couplage, les trois valeurs sont identiques, ce qui garantit que les participants dérivent exactement la même clé $K \in G_T$.

d. Propriétés de sécurité

La sécurité du protocole repose sur l'hypothèse de difficulté du Tripartite Diffie–Hellman Problem (TDH), qui consiste à calculer $e(P, Q)^{\{abc\}}$ à partir des seules informations publiques : [(P, Q, aP, aQ, bP, bQ, cP, cQ).]

Cette hypothèse est considérée comme difficile dans les groupes adaptés aux couplages, même lorsque le problème DDH est rendu trivial par la présence du pairing. En pratique, la clé K n'est pas utilisée directement, mais injectée dans une fonction de dérivation (KDF) afin d'obtenir une clé de session cryptographiquement sûre.

Comme pour le Diffie–Hellman classique, ce protocole n'assure pas l'authentification des participants et reste vulnérable aux attaques de l'homme du milieu. Il constitue néanmoins une brique fondamentale pour la construction de protocoles tripartites authentifiés ou de schémas non interactifs comme l'Identity-Based Encryption.

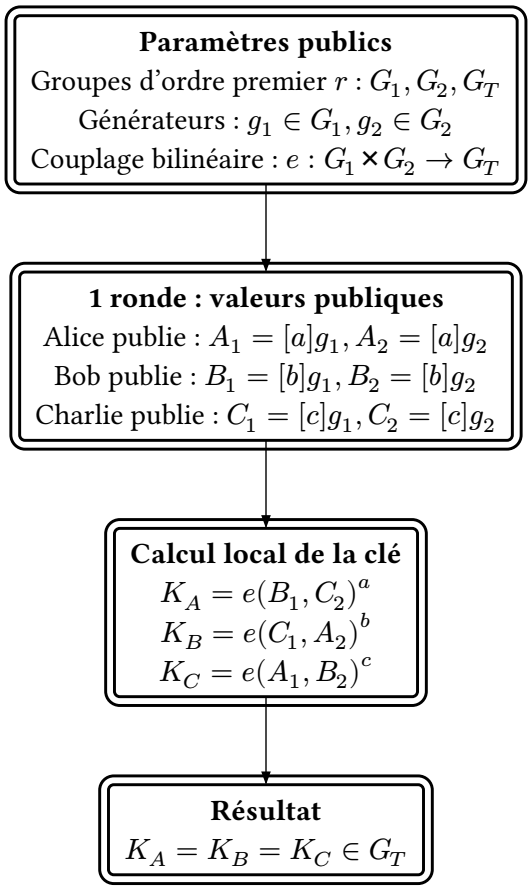
Ce protocole illustre ainsi l'apport fondamental des couplages bilinéaires : permettre des constructions cryptographiques multipartites efficaces, impossibles à réaliser avec les seuls outils classiques du logarithme discret.

III Implémentation de l'échange de clé tripartite en 1 round

Cette section présente une implémentation du protocole de Joux « one-round tripartite Diffie–Hellman » : trois participants (Alice, Bob, Charlie) obtiennent une même clé de session en une seule diffusion de messages, en s'appuyant sur un couplage bilinéaire entre deux groupes elliptiques. Le protocole original est proposé par Joux [Jou04].

1) Schéma

Dans un premier temps, voici un schéma pour expliquer le déroulement du programme :



2) Principes mathématiques utilisés

La bibliothèque `bplib` [Dca] modélise un triplet de groupes (G_1, G_2, G_T) et fournit :

- `g1 = G.gen1()` générateur de G1, `g2 = G.gen2()` générateur de G2 ;
- `G.pair()` un couplage, ici Ate sera utilisé, e : G1 × G2 ⟶ GT ;
- `G.order()` l'ordre premier r du sous-groupe (dans notre exécution : r = 16798108731015832284940804142231733909759579603404752749028378864165570215949).

Dans le code, les opérations sont cohérentes avec la convention standard :

- G1 et G2 sont manipulés additivement (points elliptiques) : [x]P = multiplication scalaire ;
- GT est manipulé multiplicativement : z^x = exponentiation dans GT.

On suppose un couplage bilinéaire non dégénéré

$$e : G_1 \times G_2 \to G_T$$

sur des groupes d'ordre premier r.

La bilinéarité signifie que pour tous $P \in G_1, Q \in G_2$ et $\alpha, \beta \in Z_r$:

$$e([\alpha]P, [\beta]Q) = e(P, Q)^{\alpha\beta}$$

La non-dégénérescence garantit que si $P \neq O$, alors il existe Q tel que $e(P, Q) \neq 1$, ce qui empêche le couplage d'être trivial.

Dans le code, on fixe $g_1 \in G_1$, $g_2 \in G_2$, puis chaque participant choisit un secret $a, b, c \in \mathbb{Z}_r^*$ et publie

$$A_1 = [a]g_1,; A_2 = [a]g_2, \quad B_1 = [b]g_1,; B_2 = [b]g_2, \quad C_1 = [c]g_1,; C_2 = [c]g_2.$$

Alors, en utilisant la bilinéarité :

$$K_A = e(B_1, C_2)^a = e([b]g_1, [c]g_2)^a = \left(e(g_1, g_2)^{\{bc\}}\right)^a = e(g_1, g_2)^{\{abc\}},$$

$$K_B = e(C_1, A_2)^b = e([c]g_1, [a]g_2)^b = \left(e(g_1, g_2)^{\{ca\}}\right)^b = e(g_1, g_2)^{\{abc\}},$$

$$K_C = e(A_1, B_2)^c = e([a]g_1, [b]g_2)^c = \left(e(g_1, g_2)^{\{ab\}}\right)^c = e(g_1, g_2)^{\{abc\}}.$$

Ainsi $K_A = K_B = K_C$, ce qui justifie mathématiquement le test d'égalité effectué dans G_T. Ce mécanisme correspond ainsi à l'argument de correction du protocole tripartite de Joux [Jou04].

a. Correspondance avec le protocole original de Joux

Dans la présentation originale de Joux [Jou04], le protocole est formulé à partir d'un unique groupe elliptique G et de deux points indépendants $P, Q \in G$, avec un couplage symétrique

$$e : G \times G \to G_T$$

Dans notre implémentation, cette construction est adaptée au modèle moderne des couplages asymétriques (type 3), dans lequel :

- G_1 et G_2 jouent le rôle de deux copies logiquement indépendantes de G ;
- les générateurs $g_1 \in G_1$ et $g_2 \in G_2$ correspondent respectivement aux points P et Q de la description originale ;
- le couplage est de la forme $e : G_1 \times G_2 \to G_T$.

Cette adaptation ne modifie ni la correction ni la sécurité du protocole : elle montre simplement les contraintes et optimisations des bibliothèques de couplages modernes, qui n'implémentent plus de couplages symétriques efficaces pour des raisons de performance et de sécurité.

b. Courbe Elliptique utilisée (bn254)

Dans le script, le groupe est instancié par `G = bp.BpGroup()` sans paramètre. Or, dans bplib [Dca, Dcb], le constructeur `BpGroup.__init__` prend par défaut `nid=NID_fp254bnb`. Autrement dit, le choix "par défaut" correspond à la courbe nommée Fp254BNb (souvent référencée comme bn254).

La courbe Fp254BNb/bn254 est une courbe de Weierstrass sur un corps premier $\mathbb{F}_p$ d'environ 254 bits, avec des paramètres (a, b) (dans la base de courbes : $a = 0$, $b = 2$) et un sous-groupe cyclique d'ordre premier r (cofacteur $= 1$), ce qui correspond bien à l'affichage Group order (bits): 254 dans la sortie.

Elle est "pairing-friendly" : son degré d'immersion (embedding degree) est $k = 12$, donc les valeurs du couplage vivent naturellement dans un sous-groupe de $\mathbb{F}_{p^{12}}^*$ (souvent noté $G_T \subset \mathbb{F}_{p^{12}}^*$).

c. Couplage utilisé

Dans notre implémentation, l'appel `G.pair(P, Q)` calcule un couplage bilinéaire non dégénéré

$$e : G_1 \times G_2 \to G_T$$

où G_1 et G_2 sont deux sous-groupes (en général distincts) d'une courbe elliptique pairing-friendly, et G_T est un sous-groupe multiplicatif d'un corps d'extension (typiquement $G_T \subset \mathbb{F}_{p^k}^*$). Cette séparation G_1 / G_2 correspond au modèle asymétrique (type 3), couramment utilisé en pratique pour des raisons d'efficacité.

Concernant la variante exacte (Tate réduit vs Ate vs optimal-Ate), l'API Python de bplib ne la nomme pas explicitement. Cependant, la bibliothèque bplib est documentée comme étant basée sur OpenPairing, une implémentation de couplages sur courbes de Barreto–Naehrig (BN) s'appuyant sur OpenSSL [Ac].

De plus, le livrable PANORAMIX [LTM18] (section d'évaluation) indique explicitement que bplib utilise l'Ate pairing et précise que la courbe employée est Fp254BNb. Ainsi, dans le cadre de ce rapport, nous adoptons l'hypothèse raisonnable que `G.pair` encapsule un couplage de type Ate (probablement une version optimisée, comme c'est standard sur BN), suivi de l'exponentiation finale.

3) Résultats (démonstrateur)

Pour illustrer le NIKE tripartite basé sur couplage, nous exécutons le prototype dont le code source est donné en (Annexe 1). Le bloc ci-dessous montre la sortie typique d'une exécution (égalité des clés dans G_T et clé de session exportée en hexadécimal) :

Le script affiche trois tests d'égalité dans G_T (tous doivent être vrais) :

```
[*] Group order (bits): 254
a = 97959392386571372070014149464232683189843674598595383663734639609015029054l5
b = 10134203465698003418838719594144630324906411920066739450740846534179623762673
c = 98247287285094207950795992295402612045970855322749823744344032707683709432l0

[*] Checking equality in GT...
K_A == K_B ? True
K_B == K_C ? True
K_A == K_C ? True

[+] Success: all three parties share the same pairing-based key.

Session key (GT element, hex):

151180a480e4707008b7500415d1a1de3cde60b10c0fac459fb7c4fb8f3d906013be043c9f5f18eb9433d3
139914096bf366d71ee8846e4eaad276a33218b2dc0a42a9743ee0f618a0f3519aa19cb0033eade8aa5ed5
1dbc25baa0754101633d139cbab51138833bcdf5a28fff434d558f3944d688f197e87945407d112c4c7514
777a3873fa4cd93f89ea516d7395adae64a5b4f08eb910a2fe6d883c4833b015b4e106d090a553cbc8f32c
a0846c2750be8c21df65ce3249e83541151b72f212ff9a46051e0914648324ddf8b662de29cf419406209c
a1f9a0fc7b5a45db680ff00d378d4c0fc65a62321a6fededce0c1cfcec27bd65a2d67cca2b33f937ee2017
57c5dcf2ce7778d7e44b275a6ccb1496ea5322faa08c336a6a4b5033b2380c96c2455d4b98f1b298769ef5
25ca780f5c140ca1fe0704e767d3fdc4c776f31c2d6ac4d4e9b403b488df6438d5a9de1244dad08adff8bb
23cd3c2ec7cfd1f201f78a94801381f3a5a9d0039696074870c28cc4f96f455612bea97772ecc6a3
```

Ainsi, bien que formulée différemment au niveau des structures algébriques, l'implémentation présentée correspond au protocole tripartite de Joux.

IV Ouverture (présentation des pistes pour n participants)

Pour la suite du projet, il pourrait être intéressant d'essayer d'étendre le sujet au dela de 3 participants. Pour cela, on a rassemblé quelques pistes pour étudier ce sujet. En annexe (Annexe 2), un tableau regroupant toutes celles trouvées.

1) Les variantes récentes de GGH15

Parmi les constructions héritées de la lignée GGH13–GGH15, la variante proposée par Bartusek, Guan, Ma et Zhandry (2018) constitue l'une des approches les plus prometteuses pour obtenir un schéma d'échange de clé non interactif multiparti réellement robuste. Contrairement aux versions originelles GGH13 et GGH15 — aujourd'hui considérées comme brisées en raison d'attaques "zeroizing" et de la rupture des hypothèses MDDH — cette variante introduit des mécanismes d'aléatorisation supplémentaires et s'appuie sur des hypothèses de sécurité plus structurelles, comme la Branching-Program Un-Annihilatability (BPUA). Bien que ces hypothèses soient fortes et encore peu étudiées, elles permettent d'établir des preuves de résistance contre toutes les attaques connues visant les maps multilinéaires basées lattices. En ce sens, cette famille représente la première tentative sérieuse de fournir une alternative « réparée » à GGH15, tout en conservant la capacité essentielle de réaliser un NIKE multiparti.

2) Les constructions basées sur iO

Les travaux d'Albrecht et al. (2020) basés sur l'obfuscation indistinguable (iO) constituent une autre piste prometteuse, bien que largement impraticable aujourd'hui. Cette approche ne dépend pas des mécanismes traditionnels des graded encodings et échappe donc aux vulnérabilités qui ont conduit à casser GGH13, GGH15 ou CLT13. En construisant des maps multilinéaires à partir d'iO, de NIZK dual-mode et parfois de FHE, les auteurs parviennent à obtenir une classe de maps pour lesquelles une version multilinéaire de DDH tient par construction. Ceci rend ces systèmes extrêmement robustes en théorie, sans vulnérabilités structurelles connues. Leur principal défaut réside dans les coûts prohibitifs de l'iO, ce qui limite tout usage pratique. Cependant, du point de vue de la recherche, ces modèles représentent probablement la direction la plus durable pour obtenir des NIKE multiparti pleinement prouvés et conceptuellement sûrs.

3) Les schémas plus récents et idéalistes (Salimi 2021, weak maps, etc.)

Enfin, les constructions plus récentes — telles que le GES de Salimi (2021) ou les versions à degré constant basées sur CLT13 proposées par Ma et Zhandry — méritent également l'attention. Elles cherchent principalement à contourner les attaques ayant compromis les graded encodings classiques en modifiant profondément les mécanismes de zéro-test ou en restreignant les fonctionnalités disponibles. Le schéma de Salimi élimine totalement les encodages de zéro test, souvent responsables des attaques zeroizing, et propose même un NIKE multiparti ou à identité. Cependant, aucun examen cryptanalytique poussé n'a encore été mené. Elles ne réparent donc pas réellement les failles connues : elles contournent plutôt les mécanismes vulnérables, mais au prix d'une grosse incertitude quant à leur solidité. Elles peuvent être vues comme prometteuses sur le plan conceptuel, mais ne peuvent pas encore rivaliser avec les constructions fondées sur iO en termes de garanties cryptographiques.

V Conclusion

Dans ce projet, nous avons étudié et implémenté un protocole d'échange de clé tripartite en un seul tour, fondé sur l'utilisation de couplages bilinéaires sur courbes elliptiques, tel que proposé initialement par Joux [Jou04]. L'objectif était de dépasser les limitations des échanges de clé classiques de type Diffie–Hellman, qui nécessitent plusieurs tours de communication ou des négociations bilatérales multiples dès lors que le nombre de participants dépasse deux.

Après avoir rappelé les principes fondamentaux de l'échange de clé Diffie–Hellman et leurs limites dans un contexte multipartite, nous avons introduit les couplages bilinéaires comme outil mathématique central permettant de construire efficacement une clé commune entre trois participants. La propriété de bilinéarité du couplage rend possible la dérivation d'un secret partagé de la forme $e(P, Q)^{\{abc\}}$, inaccessible à un attaquant malgré la diffusion publique des valeurs intermédiaires.

Nous avons ensuite détaillé la théorie du protocole tripartite de Joux, en montrant comment la correction du schéma repose exclusivement sur les propriétés algébriques des couplages et sur la difficulté du problème de Diffie–Hellman tripartite. Cette analyse met en évidence le rôle fondamental des pairings dans la cryptographie moderne, non seulement comme outils cryptanalytiques, mais aussi comme primitives constructives permettant des protocoles auparavant inaccessibles.

La seconde partie du travail a été consacrée à l'implémentation concrète du protocole à l'aide de la bibliothèque `bplib`. Nous avons montré que, bien que la formulation originale de Joux repose sur un couplage symétrique et deux points indépendants dans un même groupe elliptique, l'implémentation moderne repose naturellement sur un modèle asymétrique (type 3), utilisant deux groupes distincts G_1 et G_2.

Les résultats expérimentaux confirment que les trois participants dérivent effectivement une clé identique dans le groupe G_T, validant ainsi l'implémentation et illustrant concrètement le fonctionnement du protocole en un seul tour de communication. Cette approche met en évidence l'intérêt pratique des courbes elliptiques pairing-friendly, telles que les courbes de Barreto–Naehrig, dans la conception de protocoles cryptographiques avancés.

Enfin, ce travail souligne que, bien que le protocole de Joux ne fournisse pas d'authentification et reste vulnérable aux attaques de l'homme du milieu, il constitue une brique fondamentale pour de nombreuses constructions modernes, notamment les protocoles tripartites authentifiés.

VI Annexe

1) CODE - Three-party NIKE

```py
from bplib import bp
from petlib.bn import Bn

def random_scalar(order: Bn) -> Bn:
    """Return a random scalar in [1, order-1]."""
    rnd = order.random()
    if rnd == 0:
        return random_scalar(order)
    return rnd

def main():
    # --- Setup bilinear pairing group ---
    G = bp.BpGroup()
    g1, g2 = G.gen1(), G.gen2()
    order = G.order()

    print("[*] Group order (bits):", order.num_bits())

    # Secrets for Alice, Bob, Charlie (Bn)
    a = random_scalar(order)
    b = random_scalar(order)
    c = random_scalar(order)

    print(f"a = {int(a)}")
    print(f"b = {int(b)}")
    print(f"c = {int(c)}")

    # Public points
    A1 = g1 * a
    B1 = g1 * b
    C1 = g1 * c

    A2 = g2 * a
    B2 = g2 * b
    C2 = g2 * c

    # Alice: K_A = e(B1, C2)^a
    e_BC = G.pair(B1, C2)
    K_A = e_BC ** a

    # Bob: K_B = e(C1, A2)^b
    e_CA = G.pair(C1, A2)
    K_B = e_CA ** b

    # Charlie: K_C = e(A1, B2)^c
    e_AB = G.pair(A1, B2)
    K_C = e_AB ** c
```

```
48
49      print("\n[*] Checking equality in GT...")
50      print("K_A == K_B ?", K_A == K_B)
51      print("K_B == K_C ?", K_B == K_C)
52      print("K_A == K_C ?", K_A == K_C)
53
54      if K_A == K_B == K_C:
55          print("\n[+] Success: all three parties share the same pairing-based key.")
56      else:
57          print("\n[-] Keys do not match. Something went wrong.")
58
59      key_bytes = K_A.export()
60      print("\nSession key (GT element, hex):")
61      print(key_bytes.hex())
62
63  if __name__ == "__main__":
64      main()
```

2) Tableau : Applications multilinéaires cryptographiques (> 3 personnes)

Approche	Année	Auteurs	Hypothèses / Fondements	Forces	Faiblesses	NIKE
CLT13 (Coron et al.)	2013	Coron, Lepoint, Tibouchi	Encodages d'entiers à ordre composite (style DGHV) ; difficulté supposée de type "DLIN multilinéaire".	Pratique : démonstration d'un DH à 7 participants en ≈40 s. Conception à base d'entiers.	Cassé : l'hypothèse DDH multilinéaire (MDDH) a été rompue par Cheon **et al.** (2015). La version CLT15 a également été cassée.	Oui (conçu pour du DH multiparti)
GGH13 (Garg–Gentry–Halevi)	2013	Garg, Gentry, Halevi	Dureté de réseaux idéaux (style NTRU) : encodages gradués basés lattices.	Premier schéma de "graded map" à base de lattices ; conceptuellement général ; un exemple d'échange de clé existe.	Cassé : la dureté MDDH rompue par Hu & Jia (2016) ; autres attaques "zeroizing" (Albrecht **et al.**, Cheon **et al.**).	Oui (le schéma supportait le DH multiparti)
GGH15 (Garg–Gentry–Halevi)	2015	Garg, Gentry, Halevi	Encodages multilinéaires induits par des graphes sur des instances LWE.	Nouveau schéma asymétrique basé LWE ; structure plus riche pour l'obfuscation et l'ABE.	Cassé : la dureté MDDH rompue par Albrecht **et al.** (CLLT16).	Oui (support multiparty DH, similaire à GGH13)
Variante GGH15 de Bartusek et al.	2018	Bartusek, Guan, Ma, Zhandry	Variante de GGH15 avec davantage d'aléa ; prouvé sûr sous les hypothèses Branching-Program Un-Annihilatability et hypothèses associées.	Résiste de manière prouvée à toutes les attaques "zeroizing" publiées (analyse dans un nouveau modèle).	Repose sur des hypothèses fortes/non établies (BPUA, PRF dans NC^1, etc.) ; sécurité garantie seulement dans un modèle idéalisé.	Oui (map utilisable pour l'échange de clé, même si le focus est l'obfuscation)
Salimi 2021 GES	2021	Majid Salimi	Nouveau schéma d'encodages gradués où les utilisateurs encodent des éléments secrets aléatoires.	Élimine les encodages de zéro test publiés ; construction efficace et un MP-NIKE (même ID-based).	Tout récent, conception heuristique ; aucune preuve formelle ni réduction connue (non vérifié).	Oui (schéma MP-NIKE / ID-NIKE explicite proposé)
CLT13 degré constant (Ma–Zhandry)	2018	Fermi Ma, Mark Zhandry	Basé sur CLT13 dans un modèle de "weak map", avec de fortes hypothèses sur la complexité algébrique.	Premier encodage gradué de degré constant avec sécurité idéale dans le modèle ; aucune attaque connue sur sa fonctionnalité limitée.	Fonctionnalité limitée (pas de réutilisation avec rerandomisation complète) ; repose sur des hypothèses algébriques non prouvées.	Oui (extension de CLT13, donc supporte le DH multiparti)
Maps basées iO (Albrecht et al.)	2020	Albrecht, Farshim, Han, Hofheinz, Larraia, Paterson	Construit des maps multilinéaires à partir d'iO probabiliste, NIZK dual-mode, FHE, etc.	Construction complètement prouvée ; les analogues DDH multilinéaires tiennent par conception	Impraticable : nécessite des primitives très lourdes (iO, NIZK, FHE) ; κ doit être choisi à l'avance.	Oui (κ-DDH multilinéaire, permettant le NIKE classique)

VII Bibliographie

[Mil86] V. S. Miller, « Use of Elliptic Curves in Cryptography », in *Advances in Cryptology – CRYPTO '85*, in Lecture Notes in Computer Science, vol. 218. Springer, 1986, p. 417-426. doi: 10.1007/3-540-39799-X_31.

[Kob87] N. Koblitz, « Elliptic curve cryptosystems », *Mathematics of Computation*, vol. 48, nº 177, p. 203-209, 1987, doi: 10.1090/S0025-5718-1987-0866109-5.

[Bar20] E. Barker, « Recommendation for Key Management: Part 1 – General ». National Institute of Standards, Technology, 2020. doi: 10.6028/NIST.SP.800-57pt1r5.

[Che+23] L. Chen, D. Moody, K. Randall, A. Regenscheid, et A. Robinson, « Recommendations for Discrete Logarithm-based Cryptography: Elliptic Curve Domain Parameters ». National Institute of Standards, Technology, 2023. doi: 10.6028/NIST.SP.800-186.

[DH06] W. Diffie et M. Hellman, « New directions in cryptography », *IEEE Trans. Inf. Theor.*, vol. 22, nº 6, p. 644-654, sept. 2006, doi: 10.1109/TIT.1976.1055638.

[KE10] H. Krawczyk et P. Eronen, « HMAC-based Extract-and-Expand Key Derivation Function (HKDF) ». [En ligne]. Disponible sur: https://www.rfc-editor.org/info/rfc5869

[Bar+18] E. Barker, L. Chen, A. Roginsky, A. Vassilev, et R. Davis, « Recommendation for pair-wise key-establishment schemes using discrete logarithm cryptography: »., National Institute of Standards, Technology, Gaithersburg, MD, 2018. doi: https://doi.org/10.6028/NIST.SP.800-56Ar3.

[Res18] E. Rescorla, « The Transport Layer Security (TLS) Protocol Version 1.3 ». [En ligne]. Disponible sur: https://www.rfc-editor.org/info/rfc8446

[Jou04] A. Joux, « A One Round Protocol for Tripartite Diffie–Hellman », *Journal of Cryptology*, vol. 17, nº 4, p. 263-276, sept. 2004, doi: 10.1007/s00145-004-0312-y.

[Dc] G. Danezis et contributors, « bplib: A bilinear pairing library for petlib (based on OpenPairing) ». [En ligne]. Disponible sur: https://pypi.org/project/bplib/

[Dc] G. Danezis et contributors, « bplib ». [En ligne]. Disponible sur: https://github.com/gdanezis/bplib/tree/master

[Ac] D. F. Aranha et contributors, « OpenPairing: Implementation of a bilinear pairing over a Barreto–Naehrig curve using OpenSSL ». [En ligne]. Disponible sur: https://github.com/dfaranha/OpenPairing

[LTM18] P. Louridas, G. Tsoukalas, et D. Mitropoulos, « Integrated Service – Deliverable D5.3 », technical report, avr. 2018. [En ligne]. Disponible sur: https://panoramix-project.eu/wp-content/uploads/2019/03/D5.3.pdf

[14] D. K. Gillmor, « Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for Transport Layer Security (TLS) ». [En ligne]. Disponible sur: https://www.rfc-editor.org/info/rfc7919

[15] M. Kojo et T. Kivinen, « More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE) ». [En ligne]. Disponible sur: https://www.rfc-editor.org/info/rfc3526