SPIM

Thèse de Doctorat

école doctorale **sciences pour l'ingénieur et microtechniques**

UFC UNIVERSITÉ DE FRANCHE-COMTÉ

Algorithmes d'authentification et de cryptographie efficaces pour les réseaux de capteurs sans fil

Youssou FAYE

S2PIM

Thèse de Doctorat

UFC

école doctorale **sciences pour l'ingénieur et microtechniques**

UNIVERSITÉ DE FRANCHE-COMTÉ

THÈSE présentée par

Youssou FAYE

pour obtenir le

Grade de Docteur de

l'Université de Franche-Comté

et de

l'Université Cheikh Anta Diop (Sénégal)

Spécialité : **Informatique**

Algorithmes d'authentification et de cryptographie efficaces pour les réseaux de capteurs sans fil

Unité de Recherche :
FEMTO-ST DISC

Soutenue le 18/09/2014 en France devant le Jury :

Hervé GUYENNET	Directeur de thèse	Professeur à l'Université de Franche-Comté
Ibrahima NIANG	Co-Directeur de thèse	Professeur à l'Université Cheikh Anta Diop
Olivier FLAUZAC	Rapporteur	Professeur à l'Université de Reims
Hossam AFIFI	Rapporteur	Professeur à l'Institut Télécom SudParis
Laurent PHILIPPE	Examinateur	Professeur à l'Université de Franche-Comté
Vincent LECUIRE	Examinateur	Maître de Conférence à l'Université NANCY 1

Remerciements

Une thèse n'est pas le fruit d'un seul individu, je profite de l'occasion qui m'est donnée ici pour exprimer toute ma gratitude à toutes les personnes qui m'ont suivi et soutenu ces trois années.

En premier lieu, je tiens à remercier mes directeurs de thèse Hervé GUYENNET et Ibrahima NIANG pour l'investissement sans faille dont ils ont fait preuve dans mon encadrement tout au long de ces trois années de thèse. Ils auront été au cours de ces 3 ans bien plus que des simples directeurs de thèse et je ne les remercierai jamais assez. C'est grâce à leur confiance, leur disponibilité, leurs conseils et la liberté qu'ils ont su me donner que j'ai pu mener à bien ces recherches.

Je tiens à remercier Olivier FLAUZAC Professeur à l'Université de Reims et Hossam AFIFI Professeur à l'Institut Télécom SudParis pour l'intérêt qu'ils ont montré dans mes travaux en acceptant la charge de rapporteurs.

Je remercie également Laurent PHYLIPPE Professeur à l'Université de Franche Comté, et Vincent LECUIRE Maître de Conférence à l'Université NANCY 1, d'avoir bien voulu juger ces travaux en tant qu'examinateurs.

Merci au Professeur Pierre-Cyrille HEAM, de m'avoir apporté ses connaissances sur la problématique de l'authentification.

Je voudrais remercier tous les membres du DISC. En particulier les doctorants présents et passés, avec qui j'ai beaucoup appris, partagé et apprécié travailler.

Je voudrais également remercier tous les membres de l'Equipe de recherche Services Réseaux et Télécoms de l'UCAD.

J'adresse un remerciement aux collègues de l'équipe CARTOON et particulièrement à Yanbo SHOU.

Je remercie aussi chaleureusement mes parents, ma femme Thiara Thiandoum, mes enfants Mouhamed et Moustapha, mes frères et sœurs pour leur présence, leur soutient indéfectible et la confiance toujours renouvelée qu'ils ont su m'apporter. Je remercie également tous les membres de ma famille au sens africain du terme.

Mes amis, amies et tout ceux qui, de près ou de loin, m'ont apporté leur soutien ont également toute ma gratitude. La liste est longue, ils et elles se reconnaîtront.

Je remercie le Service de Coopération et d'Action Culturelle de l'Ambassade de France au Sénégal d'avoir financé partiellement ma thèse et m'avoir mis dans d'excellentes conditions de travail.

SOMMAIRE

Introduction générale

Ces dernières années, les progrès réalisés dans les domaines de la microélectronique, de la microtechnique, et des technologies de communication sans fil, ont permis de produire des composants de quelques centimètres à quelques millimètres cubes de volume appelés capteurs. Ces derniers intègrent : une unité de captage chargée de mesurer des grandeurs physiques (chaleur, humidité, vibrations) et de les transformer en grandeurs numériques, une unité de traitement informatique et de stockage de données et un module de transmission sans fil (wireless). Le déploiement des capteurs, parfois aléatoire, et souvent en grand nombre, dans des endroits plus ou moins hostiles en vue de collecter et de transmettre des données environnementales d'une manière autonome, forme un réseau de capteurs sans fil (RCSF).

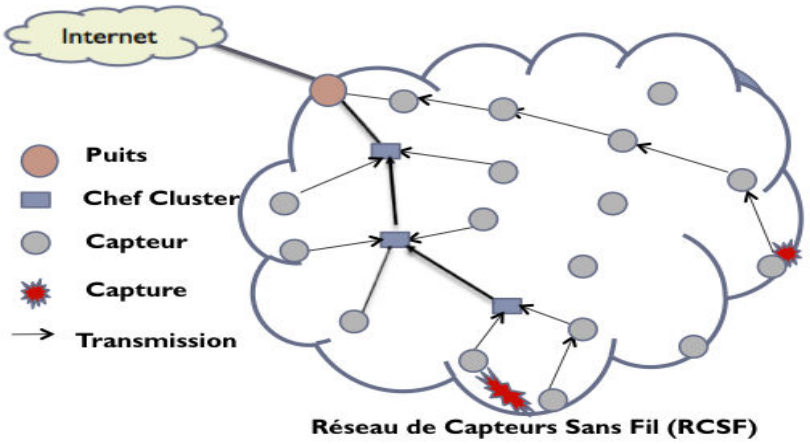


Figure 1 – Schéma résumé d'un réseau de capteurs sans fil : architecture, modèle de collecte et de transmission de données

Les réseaux de capteurs sans fil ont un intérêt particulier pour les applications militaires, environnementales, domestiques, médicales, et bien sûr les applications liées à la surveillance des infrastructures critiques. Le modèle de livraison de données ou trafic de données va essentiellement dépendre de l'application autrement dit de la raison pour laquelle le réseau a été déployé. Nous pouvons distinguer trois types de trafic de données qui peuvent être utilisés seuls ou hybrides :

– *Le modèle orienté temps (time-driven)* : c'est un schéma dans lequel les nœuds vont transmettre à intervalle de temps régulier les données obtenues des capteurs (avec ou sans agrégation des valeurs). La fréquence de l'envoi des données va dépendre des besoins de l'application du RCSF, elle peut aller de quelques millisecondes à plusieurs heures voire plusieurs jours.
– *Le modèle orienté requêtes (query-driven)* : un utilisateur ou une application (distante ou installée sur la station de base) va envoyer une requête à un nœud ou à un groupe de nœuds pour obtenir les informations de leurs capteurs.
– *Le modèle orienté événements (event-driven)* : ce modèle de communication est basé sur les événements contrôlés par le RCSF. Généralement, les nœuds vont récupérer périodiquement les

données de leurs capteurs et les analyser. Un événement ou phénomène physique détecté par un nœud à partir de ces analyses, est ensuite transmis à la station de base.

A l'heure actuelle, nous constatons une floraison d'applications pour ces réseaux et une évolution vers l'Internet des objets (Internet of Things, IoT). Ainsi, pour des applications militaires ou médicales, le besoin d'apporter une solution de sécurité fiable paraît important voir crucial. Or, de par les caractéristiques des réseaux de capteurs sans fil (absence d'infrastructure topologie dynamique, nombre important de capteurs, sécurité physique limitée, modes et lieux de déploiement offrant de multiples possibilités d'attaques) et des contraintes inhérentes aux nœuds capteurs (l'énergie, les capacités de calcul, de stockage et de transmission limitées), la sécurisation des réseaux de capteurs est à la source, aujourd'hui, de beaucoup de recherches scientifiques et techniques. Des travaux de recherche antérieurs ont d'ailleurs démontré que les solutions de sécurité proposées pour les réseaux sans fil (mobiles et adhoc), surtout celles basées sur l'utilisation de la cryptographie à clé publique ne peuvent pas être appliquées directement dans les réseaux de capteurs sans fil du fait des coûts de calcul importants engendrés.

Cependant, même si le contexte a évolué, de la machine non connectée aux réseaux filaires puis sans fil, l'objectif de la sécurité a toujours été globalement le même, à savoir assurer les services de sécurité de base tels que l'authentification, le contrôle d'accès, la confidentialité, l'intégrité, la disponibilité etc... Si dans les autres réseaux et particulièrement dans les réseaux filaires, la solution consiste à augmenter toujours plus la puissance des chiffrements, la problèmatique posée par la faiblesse de calcul et le besoin d'économiser l'énergie des capteurs amènent à se poser des questions nouvelles sur les méthodes de sécurité à utiliser. Il est important que les solutions de sécurité à mettre en œuvre soient les moins coûteuses en consommation de ressources et visent à réduire les temps, le nombre de transmissions, et l'occupation de la bande passante. En l'occurrence, ce sont les solutions apportant une sécurité maximale tout en préservant la durée de vie du capteur, c'est-à-dire en utilisant peu la puissance de calcul et l'énergie des capteurs, qui doivent permettre de répondre aux problèmes de sécurité dans ce type de réseau.

Dans cette thèse, nous nous sommes intéressés aux problèmes de sécurité dans les RCSFs, nous cherchons à définir des mécanismes et solutions peu coûteux en énergie pour les réseaux de capteurs sans fil qui prennent en compte la relative faiblesse de défense d'un réseau autonome. Dans cette optique nous appliquons des solutions de cryptographie symétrique peu gourmandes en énérgie basée sur les fonctions de hachage, et des mécanismes de cryptographie asymétrique basés sur les courbes elliptiques.

Cette thèse se compose de deux parties : une partie état de l'art et une partie contribution. La premiere partie constituée d'un seul chapitre décrit l'état de l'art. Il est consacré à l'introduction de la problématique générale de la thèse.

Dans un premier temps nous présentons le contrôle d'accès et les particularités de la problématique de sécurité dans les RCSFs. Ensuite un focus sur différents mécanismes de sécurité légers en énergie et basés notamment sur les fonctions et arbres de hachage, les chaînes de clés à sens unique, les formes cryptographiques et plus spécifiquement les courbes elliptiques, permettra une mise en évidence des points forts et faibles utiles pour les RCSFs. Enfin, nous comparons et examinons différents algorithmes et protocoles pour mettre en évidence leurs objectifs, dispositifs, complexité, limites, etc.

La seconde partie comprend les chapitres 2, 3 et 4 qui constituent nos contributions. Dans la première contribution du chapitre 2, nous décrivons quelques protocoles d'authentification par mot de passe pour ensuite améliorer la sécurité et les performances de ces protocoles existants. La solution proposée utilise les mêmes mécanismes des fonctions de hachage, et bénéficie en plus d'une validation par simulation et implémentation.

La deuxième contribution constitue une suite logique de la première, oú après avoir introduit le nouveau concept de probabilité de risque, nous déterminons pour différentes architectures de déploiement, la consommation énergétique de la solution proposée en comparaison avec celles existantes. Enfin nous procédons à une vérification automatique de quelques propriétés de base relatives à des services de sécurité à l'aide d'outils de validation.

Le chapitre 3 est consacré à la multiplication scalaire sur les courbes elliptiques, l'opération de base de tous les protocoles reposant sur ces objets mathématiques. Nous présentons brièvement quelques solutions existantes pour accélérer la multiplication scalaire sur les courbes elliptiques définies dans des corps finis premiers. Nous présentons notre troisième contribution qui décrit un nouveau mécanisme basé sur l'opposé et l'ordre d'un point et qui réduit le nombre d'opérations de points. Ce dernier présente en plus l'avantage de pouvoir être combiné avec toutes les techniques existantes. Les performances du mécanisme proposé ont été évaluées en utilisant les paramètres recommandés par un organisme standardisant les courbes elliptiques NIST-192 (National Institute of Standards and Technology).

Dans le dernier chapitre, qui constitue notre quatrième contribution, nous nous sommes intéressés à l'accélération des points résultants du partitionnement du scalaire qui génèrent des coûts additionnels de calcul et de stockage mémoire durant les calculs parallèles des partitions. Nous comparons différentes formules de points existantes afin de mettre en évidence leur efficacité, et ce avec l'objectif de les utiliser dans un algorithme distribué que nous mettons en place.

Nous concluons cette thèse en résumant les avantages, les inconvénients et les limites des solutions que nous avons proposées. Les perspectives envisagées pour ces solutions innovantes complètent cette partie.

I

ETAT DE L'ART

1

Etat de l'art

Sommaire

1.1/ Le contrôle d'accès dans les réseaux de capteurs sans fil (RCSFs)

Le contrôle d'accès est un service de sécurité critique dans les réseaux de capteurs sans fil (RCSFs). C'est un problème ancien dans les réseaux classiques, mais qui n'a pas reçu assez d'attention dans les RCSFs.

Le nœud capteur a généralement une faible capacité de communication, de calcul, de mémorisation et d'énergie. Les RCSFs sont déployés pour une variété d'applications, souvent de surveillance et de collecte de données : surveillance militaire, médicale, contrôle de l'environnement etc.. Les modèles de livraison des informations peuvent être périodiques, évennementiels ou basés sur des requêtes.

Des facteurs comme le déploiement en grande quantité de nœuds dans des environnements souvent hostiles notamment ceux du domaine militaire ou publique, la communication sans fil, l'interaction

physique avec l'environnement etc. rendent ces réseaux très vulnérables à beaucoup d'attaques. Ainsi, le contrôle d'accès devient un vrai challenge dans les RCSFs. Parfois les données collectées ne sont pas critiques, comme par exemple une requête pour la collection des températures d'une zone. Cependant, dans des applications critiques, les données doivent être protégées contre tout accès non autorisé. Dans les applications temps réel, les données doivent non seulement être disponibles à la demande de l'utilisateur, mais aussi accessibles depuis n'importe quel endroit grâce aux capteurs en mode ad hoc[3].
Les mécanismes de contrôle d'accès définissent les politiques selon lesquelles les capteurs ou utilisateurs doivent accéder ou envoyer des requêtes au réseau. Dans le contexte des RCSFs, il est essentiel de limiter l'accès du réseau seulement aux nœuds ou utilisateurs éligibles, tandis que les messages ou requêtes provenant des nœuds externes (non autorisés) ne devraient ni être transmis, ni traités dans le réseau. En plus l'interception, la modification ou la suppression d'un paquet peuvent être détectées.

Le contrôle d'accès devient particulièrement difficile en présence d'un nœud compromis et des attaques telles que le déni de service (DdS) et la répétition. Dans les environnements hostiles, non seulement les capteurs, mais les utilisateurs peuvent être compromis via une manipulation distante les amenant à avoir un comportement malveillant. L'authentification des paquets à travers une communication multisauts utilisant une clé symétrique est un vrai défi, parce que les nœuds intermédiaires ont besoin de la clé pour authentifier les paquets. Par conséquent, un attaquant qui compromet ou capture un nœud obtient la clé. Ainsi, les solutions de sécurité dans ce domaine ne doivent pas être liées à un seul capteur [85]. Par ailleurs la limitation des capteurs en termes de ressources restreint l'utilisation des mécanismes de sécurité gourmands en ressources.

Dans cette première partie, nous proposons une étude des mécanismes et solutions de contrôle d'accès dans les réseaux de capteurs sans fil. Nous examinons différents algorithmes pour mettre en évidence leurs objectifs, dispositifs, complexités, limites, etc.. Nous comparons également ces algorithmes de contrôle d'accès basés sur un certain nombre de paramètres jugés utiles dans l'environnement des RCSFs.

1.2/ Le contrôle d'accès : un service de sécurité

Un réseau de capteurs est généralement constitué d'un grand nombre de nœuds répartis dans un espace géographiquement limité avec une ou plusieurs stations de base (SB). Dans ce cas, tous les capteurs font confiance à la station de base. Sans perte de généralités, il est important pour une meilleure étude du contrôle d'accès dans les RCSFs, de définir l'architecture physique du réseau de capteurs (plate ou hiérarchique), et le modèle de contrôle d'accès qui, souvent varie en fonction des services fournis par le réseau de capteurs et de la façon dont il est exploité : agrégation, communication interne, communication avec des utilisateurs externes etc..

1.2.1/ Architecture de contrôle d'accès

Dans la suite, nous allons considérer deux types de RCSFs en fonction des types de communications et des services du réseau comme le montrent les Figures 1.1 et Figure 1.2. Comme présenté sur la Figure 1.1, nous avons un réseau de capteurs qui offre des services aux utilisateurs où la station de base sert de point d'accès pour la gestion et l'administration du réseau. Les capteurs servent de points d'accès aux utilisateurs (ordinateur portable, PDA, téléphone mobile). Seuls les utilisateurs qui souscrivent aux services du réseau peuvent accéder aux données et le modèle de livraison des données se fait à la demande des utilisateurs. La Figure 1.2 montre un RCSF sans utilisateur (excepté la station de base) où les données sont envoyées vers la station de base. Le modèle d'acquisition et de livraison de données dépend de l'application utilisée, il peut être *continu* (collection

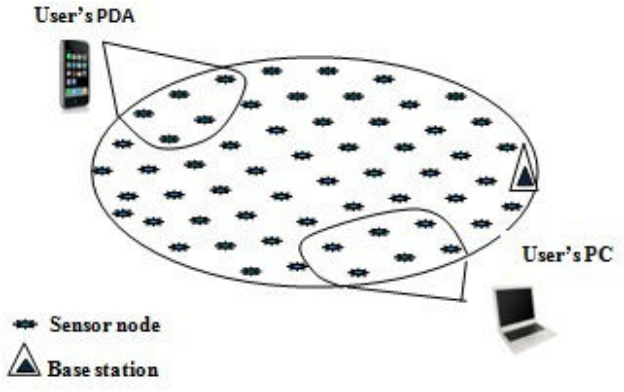


FIGURE 1.1 – Architecture RCSF avec utilisateurs

et envoi périodique de données d'une zone), *événementiel* (collection et envoi de données à la suite d'un évènement), en *mode basé sur des requêtes* (collection et envoi de données à la demande), ou *hybride*. De ces deux architectures, on peut distinguer deux niveaux de contrôle d'accès : le *niveau*

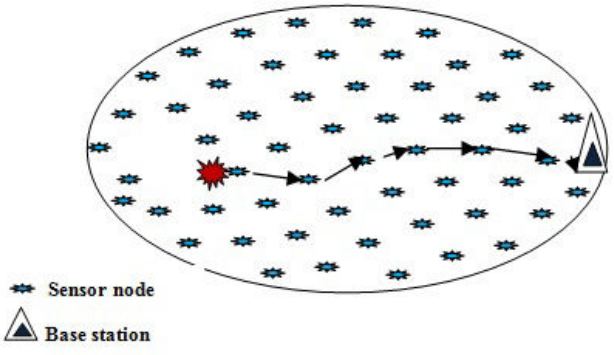


FIGURE 1.2 – Architecture RCSF sans utilisateurs

interne et le *niveau externe* [16].

– Le *contôle d'accès interne :* sécurise les communications entre capteurs et les communications entre capteurs et station de base. Il implique les deux architectures des Figures 1.1 et 1.2 [26, 45, 86, 110].
– Le *contôle d'accès externe :* sécurise les communications entre le réseau de capteurs (capteurs et station de base) et les utilisateurs externes. Ces derniers peuvent, en fonction des services souscrits, envoyer des requêtes aux capteurs voisins, qui vérifient la légitimité de la requête. Cependant, un utilisateur ne doit pas arbitrairement participer à des communications avec le RCSF. Seule l'achitecture de la Figure 1.2 est concernée .

1.2.2/ LES SERVICES DE SÉCURITÉ DU CONTRÔLE D'ACCÈS

Etant lui même un service de sécurité, le contrôle d'accès peut être divisé en deux services de sécurité : *l'authentification* et *l'autorisation*.

– *L'authentification :* elle consiste à établir une relation entre une entité et son identité. Une identité est une propriété individuelle qui ne peut pas être forgée ou copiée. On distingue deux types d'authentification : l'*authentification d'utilisateur* et l'*authentifcation de requêtes*.

  - Dans une *authentification d'utilisateur*, un utilisateur envoie son nom à un capteur et lui prouve son identité, le capteur est en mesure de déterminer si l'identité est valide ou non.
  - *L'authentification de requêtes* permet de vérifier si une requête provient d'un utilisateur autorisé, d'une station de base (SB) ou d'un capteur. Un réseau de capteurs produit l'authentification de requêtes si les propriétés de *sécurité* et de *vivacité* suivantes sont satisfaites (avec une certaine probabilité).
    - *Sécurité* : si dans un réseau de capteurs, un nœud capteur accepte une requête comme légitime, celle-ci provient du RCSF ou d'un utilisateur autorisé.
    - *Vivacité* : toute requête légitime sera reçue par tous les nœuds capteurs capables de la traiter afin de donner une réponse à l'entité légitime (capteur, station de base, utilisateur) l'ayant postée. Ce qui limite la propagation de fausses requêtes.
- *Autorisation* : consiste à établir une relation entre un utilisateur et un ensemble de privilèges (droits d'accès, opérations permises etc.). Dans une autorisation, un utilisateur envoie son nom avec sa requête (exemple : lire, écrire etc.) à un capteur, qui vérifie si l'opération demandée est autorisée ou non.
  Notons qu'en pratique, dans un mécanisme de contrôle d'accès, l'authentification d'utilisateurs ou de requêtes, l'autorisation et l'accès aux données peuvent être combinés dans une seule opération. Quand une requête est envoyée, le mécanisme de contrôle d'accès vérifie sa légitimité (authentification et autorisation), et envoie une réponse (les données demandées ou un message de refus d'accès).

1.2.3/ Les vulnérabilités

Les RCSFs présentent un grand nombre de vulnérabilités qui les exposent à différentes types d'attaques. Nous distinguons les vulnérabilités physiques et les vulnérabilités technologiques.
- *Les vulnérabilités physiques* : elles sont liées à la nature de déploiement notamment les lieux publiques ou hostiles, ce qui expose les liens de communication à des attaques passives comme l'écoute clandestine, le brouillage par interférence. Les nœuds capteurs sont vulnérables à la capture physique et au vandalisme. Le nombre très élevé de nœuds dans les RCSFs sans infrastructure fixe nécessite le développement de protocoles flexibles et scalables. L'addition de nouveaux nœuds ou le disfonctionnement d'un nœud rendent la topologie dynamique, ce qui entraine des solutions plus complexes.
- *Vulnérabilités technologiques* : les services de sécurité doivent prendre en compte les contraintes liées à la technologie des capteurs :
  - *L'énergie* : la consommation énergétique d'un nœud capteur peut être divisée en trois parties : l'énergie de l'unité de captage, l'énergie pour la communication (transmission/réception) et l'énergie du microprocesseur.
  - *Le calcul* : le nœud capteur est doté d'un processeur d'une capacité de calcul très limitée d'où l'exécution impossible d'algorithmes cryptographiques complexes.
  - *La mémoire* : après le chargement du système d'exploitation, le nœud capteur ne dispose pas suffisamment d'espace mémoire pour le stockage de certaines clés et l'exécution de certains algorithmes.
  - *Transmission/réception* : les capacités de transmission/réception du nœud capteur sont limitées pour des besoins de conservation d'énergie. La transmission est l'opération la plus coûteuse dans les RCSFs. Il a été démontré dans plusieurs publications scientifiques que la transmission d'un bit est équivalent en termes d'énergie, de l'exécution d'environ 800 à 1000 instructions[57]. Cette valeur augmente avec la portée de la radio.

1.2.4/ Les défis de sécurité

Les défis de sécurité pour un mécanisme de contrôle d'accès est d'assurer un certain nombre d'exigences de sécurité à travers l'authentification et l'autorisation tout en prenant en compte les ressources limitées des capteurs :

- Faire en sorte qu'un utilisateur ou capteur non autorisé qu'on appellera « adversaire ou attaquant » dans la suite, ne réussisse pas à intégrer ou à utiliser les services du réseau de capteurs.
- Les nœuds capteurs et utilisateurs autorisés ne doivent pas traiter ou recevoir des données corrompues.
- L'accès au réseau ne doit pas être interdit aux capteurs et utilisateurs légitimes.
- Le mécanisme doit également prendre en compte les exigences des capteurs en termes de capacité de stockage, de calcul et de transmission. Ainsi, le mécanisme de contrôle d'accès doit minimiser le stockage des clés et utiliser des algorithmes cryptographiques moins coûteux en calcul. L'énergie étant la ressource la plus précieuse dans un réseau de capteurs avec une durée de vie du capteur qui en dépend, elle doit être économisée surtout lors des transmissions. Il faut des protocoles de contrôle d'accès qui minimisent les tranmissions.
- Des techniques d'agrégation vont imposer des mécanismes de sécurité point à point, difficiles à mettre en œuvre lorsque les données sont chiffrées. Le chiffrement et l'agrégation sont deux concepts qui ne vont pas très bien ensembles.

Une solution de bout en bout serait intéressante pour l'économie d'énergie du RCSF, mais si les capteurs intermédiaires se contentent uniquement de recevoir un message et de le retransmettre sans l'authentifier, une telle solution sera très vulnérable aux attaques de type déni de services (DdS) et de répétition. Pour assurer l'authentification, les capteurs doivent déchiffrer puis chiffrer les messages reçus, cela nécessite beaucoup de calculs et une augmentation du temps de traitement. Ce qui ne répond pas aux exigences de certaines applications comme le cas d'une application temps réel.
Voilà quelques exigences parmi d'autres qu'un mécanisme de contrôle d'accès doit prendre en compte.

1.3/ Mécanismes de sécurité légers en énergie pour le contrôle d'accès des RCSFs

1.3.1/ La cryptographie

Les mécanismes de sécurité du contrôle d'accès permettent de mettre en œuvre des services de sécurité. Dans la plupart des mécanismes de sécurisation actuelle, la cryptographie est sans doute la technique la plus utilisée dans le cadre des réseaux filaires et des réseaux sans fil traditionnels disposant d'une capacité de calcul et de mémoire conséquente. Les solutions de cryptographie sont réputées comme des solutions sûres qui répondent à l'ensemble des problèmes liés à la sécurité des données. Les spécificités des réseaux de capteurs, à savoir une faible puissance de calcul et une mémoire limitée auxquelles se rajoute la problématique de préservation de l'énergie, sont des freins considérables à l'utilisation des systèmes cryptographiques courant réputés sûrs (SSL, RSA, etc.). Les travaux de recherche actuels s'attachent à trouver des solutions dites de cryptographie légères [43]. Ces solutions consistent à adapter les algorithmes de cryptographie classiques pour les réseaux de capteurs, ou à en trouver de nouveaux tout aussi efficaces en termes de sécurité, de temps d'exécution et de consommation énergétique. On distingue deux types de cryptographie : la cryptographie symétrique à clé secrète et la cryptographie asymétrique ou à clé publique.

1.3.1.1/ La cryptographie symétrique

L'algorithme de chiffrement par blocs AES a été proposé comme algorithme de chiffrement dans le standard IEEE 802.15.4 pour sécuriser au niveau de la couche MAC les données transitant sur des réseaux de capteurs sans fil. Le temps d'exécution de cet algorithme de chiffrement donne d'assez bons résultats (de l'ordre de plusieurs microsecondes) et limite le surcoût énergétique du chiffrement de données. Néanmoins ces bons résultats sont obtenus par le fait que le chiffrement est effectué au niveau hardware par le module transceiver (plusieurs millisecondes dans le cas de chiffrement software) [56]. D'autres algorithmes de chiffrement symétriques ont été testés sur les capteurs dont les résultats sont visibles dans [68]. Ces résultats permettent de dire qu'aujourd'hui les solutions de chiffrement à clés symétriques sont exploitables au sein des réseaux de capteurs et apportent une réelle solution pour la sécurité du réseau. Cependant, si le chiffrement à clé symétrique est possible au sein des réseaux de capteurs, la sécurité totale de ce type de solution reste à démontrer. D'une part parce que le chiffrement à clé symétrique ne garantit pas l'authentification des données comme peut le faire la signature numérique des chiffrements à clés publiques, et d'autre part parce que se pose le problème de la distribution des clés de chiffrement. Ainsi, si le protocole IEEE 802.15.4 spécifie la méthode de chiffrement à utiliser, à aucun moment elle ne spécifie comment doivent être gérées les clés et comment permettre l'authentification des données.

Lors de la conception du protocole de sécurité pour un RCSF ou de la programmation des nœuds, il est important de choisir la ou les couches OSI qui seront chargées de chiffrer, déchiffrer et vérifier l'authenticité des messages. On peut distinguer trois niveaux :
– La sécurité gérée au niveau du contrôle d'accès au medium.
– La sécurité gérée au niveau applicatif.
– L'approche hybride ou ces deux solutions sont combinées selon les besoins ou le contexte.
L'avantage de gérer le chiffrement au niveau le plus bas est de pouvoir changer facilement le modèle d'utilisation des clés sans modifier la partie logicielle et les protocoles de plus haut niveau, la sécurité est prise en charge de manière transparente. De plus, un paquet invalide sera détecté avec moins de calcul. La gestion de la sécurité au niveau applicatif permet de fournir un flux de bits à transmettre au module d'émission et ainsi de garder une séparation nette des différents niveaux.

L'approche hybride reste envisageable si l'application utilise un chiffrement de bout en bout entre un nœud et la station de base (SB) à travers une clé partagée par le réseau. Ce modèle permet ainsi de faire transiter de l'information confidentielle à travers le réseau tout en autorisant les entêtes des paquets d'être compréhensibles par les nœuds relais mais toujours confidentiels pour un attaquant passif.

On peut distinguer quatre types de communication pour les RCSFs :
– Unidirectionnelle : entre un capteur et la station de base (SB), ou entre un couple de capteurs ;
– Diffusion globale : entre la SB et l'ensemble des capteurs ;
– Diffusion locale : entre un capteur et ses voisins, ou un ensemble de capteurs formant un cluster ;
– Agrégation : dans ce type de communication, les réponses des capteurs font suite aux requêtes de la SB, ou sont déclenchées par des événements.
Pour chacun de ces types de communications, une clé cryptographique s'impose. On distingue généralement quatre modèles d'utilisation des clés :
– Clé partagée par le réseau : c'est la clé partagée par l'ensemble des nœuds du réseau. C'est une clé globale qui est utilisée par la station de base pour chiffrer des messages diffusés à tout le groupe. C'est une solution simple oú toutes les communications peuvent être chiffrées simplement en utilisant un minimum de mémoire (stockage d'une seule clé). Cette solution permet de résoudre en partie le problème de l'écoute passive, car l'information ne circule plus en

clair. Elle n'est pas satisfaisante : d'une part, l'authentification obtenue est de niveau faible car on ne fait que prouver qu'un message provient du groupe et non d'un capteur précis ; d'autre part la clé du réseau étant unique, lorsqu'un nœud est attaqué, la sécurité du réseau est compromise. En plus, cette solution n'autorise pas l'entrée de nouveaux capteurs dans le RCSF, à moins de faire intervenir un dispositif supplémentaire pour l'établissement dynamique des clés secrètes comme un centre de distribution de clés qui serait un médiateur dans l'établissement dynamique de clés secrètes.

– Clé partagée par paire de noeuds : chaque nœud partage une clé dédiée avec son voisin. Si un nœud possède n voisins, il aura (n-1) clés à stocker pour pouvoir communiquer avec ses voisins. Chacune de ces clés est connue seulement par ce nœud et un des (n-1) autres nœuds. L'attaque d'un nœud et la récupération de ses informations secrètes n'affectent pas la sécurité des autres nœuds. Les dommages sont donc limités à ce nœud et n'affectent que les messages passés et futurs envoyés de ou vers ce nœud. Cependant, cette technique exige une capacité mémoire importante pour stocker les (n-1) clés (n peut être grand). Elle est très coûteuse en temps de calcul car chaque nœud qui reçoit un message doit le décrypter et l'encrypter avant de le renvoyer. En plus cette solution n'est pas dynamique et n'autorise pas l'entrée de nouveaux capteurs dans le réseau. Enfin, la gestion des clés est encore plus difficile. Compte tenu des ressources très restreintes des capteurs où chaque octet utilisé doit être justifié, ce modèle n'est probablement pas envisageable.
– Clé partagée par groupes de nœuds (cluster) : ce modèle d'utilisation des clés est un compromis entre la clé partagée par le réseau et la clé partagée par paire de nœuds. Le principe est d'utiliser une clé identique pour tout un ensemble de nœuds, appartenant à un groupe ou cluster (localisation géographique similaire, fonction dans le réseau identique, etc.) dans une communication intra-cluster, et d'utiliser une clé dédiée à la communication entre chaque paire de groupes (inter-cluster) comme le modèle avec une clé par paire de nœuds. Cette technique permet de limiter la consommation mémoire tout en réduisant la portée d'une attaque réussie contre l'un des nœuds d'un groupe. En termes de scalabilité, un compromis acceptable est possible dans ce modèle, parce que le nombre de clés augmente avec le nombre de groupes et non avec la taille du réseau. Il est difficile à mettre en place, et la formation de clusters est un processus dépendant.
– Clé individuelle : chaque nœud possède une clé qu'il partage avec la station de base. Un message envoyé par un nœud est sécurisé sur le réseau jusqu'à atteindre la SB. Cette solution est une des meilleures techniques pour limiter la consommation du réseau. Cependant, elle ne permet pas de sécuriser les informations transmises entre nœuds car ceux-ci ne possèdent pas de clés communes pour communiquer de manière sécurisée entre eux. Si la SB est compromise tout le réseau est menacé.

1.3.1.2/ La gestion des clés

Avant qu'un réseau de capteurs puisse échanger des données sécurisées, des clés de chiffrement doivent être établies entre les nœuds. La distribution de clés entre les nœuds est typiquement un mécanisme de sécurité non triviale et est aujourd'hui un véritable challenge dans les réseaux de capteurs sans fils. La plupart des solutions de sécurité requièrent l'utilisation de clés de cryptage qui sont partagées par les parties communicantes. La gestion des clés est un terme générique pour définir la distribution des clés qui inclut également les processus d'initialisation ou de génération des clés, la distribution initiale de clés ou l'établissement de clés, la révocation de clés et la mise à jour de clés. Elle reste difficile à réaliser dans les RCSFs du fait de l'absence d'infrastructure centralisée en dehors de la SB, et surtout quand elle doit se faire après le déploiement. La couche liaison de données est la première couche à avoir besoin d'une gestion de clés. Le standard IEEE 802.15.4 le plus utilisé à ce niveau considère l'utilisation de clés pour l'échange sécurisé des

transmissions de données sans préciser comment échanger des clés en toute sécurité. Cela laisse ouverte la problématique de gestion de clés qui est au centre de nombreuses recherches récentes. En plus de la couche liaison, les couches supérieures telles que la couche réseau et application doivent également échanger des clés en toute sécurité.

Il est important de noter que le choix du type de clés à utiliser dépend très souvent du modèle de communication du RCSF, et ce dernier est basé sur l'application. Il existe trois catégories générales de mise en place des clés dans un réseau :

- *Un serveur en qui chaque nœud a confiance (trusted-server scheme)* : dans ce schéma, les nœuds utilisent un serveur pour établir leur clé de session. Ce type de schéma ne convient pas souvent dans les RCSFs à cause du manque d'infrastructure en qui chaque nœud a confiance et peut communiquer directement. Les nœuds éloignés de la SB sont donc contraints d'envoyer leurs messages à d'autres nœuds plus proches de la station de base afin de l'atteindre.
- *Le consensus entre les nœuds uniquement (self-enforcing scheme) :* dépend de la cryptographie asymétrique. La cryptographie asymétrique est plus flexible. Elle ne nécessite pas de pré-distribution de clés, ni de clés partagées par paire de nœuds, ni de fonctions de hachage compliquées. Cependant, ce schéma ne convient pas dans les RCSFs à cause des ressources limitées des capteurs.
- *Pré-distribution des clés :* les informations relatives aux clés sont distribuées aux capteurs avant le déploiement. Ces clés doivent être distribuées entre les capteurs, la station de base et les utilisateurs. On peut distinguer trois approches de pré-distribution de clés : l'approche déterministe, l'approche probabiliste et l'approche hybride.
  1. *L'approche déterministe de pré-distribution de clés* : cette approche garantit que deux nœuds quelconques partagent une ou plusieurs clés communes. La distribution de clés est déterminée par le modèle de communication et d'utilisation des clés du protocole de sécurité. Rappelons qu'il existe différents modèles d'utilisation de clés déterministes : une clé partagée par tout le réseau, par paire de nœuds, par groupe de nœuds [26, 66, 45].
  2. *L'approche probabiliste de pré-distribution de clés* : pour les réseaux avec un grand nombre de nœuds, l'approche probabiliste est plus efficace que la méthode déterministe. Dans ce schéma, l'existance d'une ou de plusieurs clés communes entre des nœuds quelconques est incertaine et est garantie avec une certaine probabilité. L'idée de base est de distribuer aléatoirement avant le déploiement à chaque capteur un sous-ensemble de clés k issus d'un ensemble de clés m. Le nombre de clés du sous-ensemble k est choisi de telle manière que deux sous-ensembles aléatoires de m auront une certaine probabilité p d'avoir au moins une clé commune. Il faut trouver un bon compromis entre la taille de m et la valeur k du nombre de clés par nœud pour obtenir une probabilité maximale de réseau connecté. Une grande valeur de m réduit la probabilité p de connectivité du réseau, tandis qu'une valeur plus faible diminue la sécurité en faisant tendre le modèle d'utilisation des clés vers l'équivalent d'une clé unique partagée par le réseau. Ce mécanisme supporte le passage à l'échelle [45, 26].
  3. *L'approche hybride de pré-distribution de clés :* cette approche est une hybridation entre la classe probabiliste et la classe déterministe. Elle vise à réduire le coût en stockage trop important requis par la classe probabiliste, et à améliorer le niveau de sécurité face à des compromissions de la classe déterministe [75, 41].
- *Sans Pré-distribution de clés :* contrairement aux autres mécanismes qui utilisent des clés pré-chargées initialement, ce mécanisme considère les réalités du RCSF. Si un adversaire ne connait pas quand et où un RCSF sera déployé, il sera très difficile de lancer des attaques durant la phase d'initialisation. Dans l'exemple du protocole de diffusion de clés par contamination (key infection), l'établissement des clés s'effectue dans un un délai relativement court avec peu de

transmissions [4]. Le protocole consiste pour chaque nœud à choisir une clé aléatoire et à la diffuser en clair aux nœuds voisins au moment de son arrivée dans le terrain sachant qu'un adversaire a très peu de probabilité d'intercepter une communication aussi locale (et à faible portée) et ne peut dans certains cas surveiller l'ensemble du site de déploiement qui parfois n'est même pas accessible. Ce type de schéma consomme moins d'énergie et ne nécessite pas le stockage de beaucoup de clés contrairement aux autres schémas. Cependant, il ne permet pas l'ajout de nouveau nœud une fois que les clés sont établies.

1.3.1.3/ La cryptographie asymétrique

Les mécanismes de cryptographie asymétriques sont potentiellement de bonnes solutions de sécurité quand le nombre de nœuds est très élevé comme les réseaux de capteurs. Ils posent moins de problèmes pour la gestion des clés. Cependant, la cryptographie asymétrique utile à l'établissement de clés partagées entre nœuds n'est pas réaliste. Les demandes en temps de calcul et en mémoire sont si fortes dans ce cas en plus de la nécessisté d'une infrastructure centralisée pour la gestion des clés. Des travaux de recherche essaient d'optimiser des algorithmes asymétriques comme RSA, appelé WM-RSA pour les capteurs de type MicaZ [103]. Les résultats sont meilleurs que ceux de RSA, mais le temps d'exécution est toujours de l'ordre de la seconde, temps qui n'est pas envisageable sur certaines applications des réseaux de capteurs. En plus, la taille des clés pose toujours des problèmes de stockage (1024 bits pour RSA). Dans cette optique, des travaux récents tentent d'apporter une réponse avec l'utilisation de cryptographie à clé publique basée sur les courbes elliptiques (elliptic curve cryptography, ECC). Les courbes elliptiques assurent le même niveau de sécurité que RSA avec des clés plus courtes [55].

1.3.2/ Chaînes de clés à sens unique

Les chaînes de clés à sens unique peuvent être utilisées avec la cryptographie. Une chaîne de clés à sens unique est obtenue par application successive d'une fonction de hachage à sens unique sur une valeur. Les clés générées permettent d'authentifier des requêtes ou des utilisateurs à travers des codes d'authentification de message (Message Authentication Code, MAC). Rappelons qu'un MAC peut être considéré comme une signature bipartie en cryptographie asymétrique permettant de garantir d'une part l'intégrité du message envoyé et d'autre part la vérification de l'identité de l'émetteur. Un MAC est un algorithme qui prend en entrée un message m et une clé k puis produit un condensé : MAC_K(m). Pour le récepteur, la vérification consiste à vérifier si le message reçu n'a pas été modifié en cours de route, i.e. pour le message m' reçu, on a bien MAC_K(m)=MAC_K(m').

Le principe des chaînes de clés à sens unique consiste à affecter une clé K_0 initialement (avant le déploiement du RCSF) aux capteurs. Ensuite une séquence de clés est générée. La dernière clé K_n est choisie aléatoirement , puis les clés K_{n-1}, K_{n-2}, ...K_1,K_0 sont générées par application successive de la fonction de hachage à sens unique F : K_j=F(K_{j+1}), pour 0 <= j < n. La clé K_0 générée en dernier est distribuée à tous les nœuds du réseau avant le déploiement. Puisque F est une fonction à sens unique, étant donné K_{j+1}, on peut calculer K_0, K_1,....., K_j, mais on ne peut pas calculer K_{j+1} si K_0, K_1,....., K_j sont données. Ainsi, quand un utilisateur veut accéder au réseau, une sous chaîne de la chaîne de clés lui est attribuée, et en même temps distribuée aux nœuds du réseau. Si la sous chaîne de clés reçue par un utilisateur est K_n,....., K_m avec $n < m$, cela signifie qu'il peut envoyer (m-n+1) requêtes au réseau et pour chaque requête, une clé est utilisée. Un nœud capteur du réseau qui reçoit une requête authentifie la clé K_i en vérifiant que quelque soit $i > j$, K_j=F^{i-j}(K_i) où K_i est la clé d'authentification stockée par le nœud. Si l'authentification passe, cela signifie que la requête est légitime, et le nœud répond à la requête en remplaçant K_j par K_i, sinon la requête est rejetée. Ce mécanisme a l'avantage d'utiliser la cryptographie à clé symétrique, avec des communications simples permettant, via l'authentification, de se protéger

contre des attaques comme le déni de service, la mascarade qui sont souvent fréquentes dans les protocoles de contrôle d'accès. Sa sécurité dépendra de la fonction à sens unique utilisée. Par contre le mécanisme n'est pas *scalable*, et un seul utilisateur peut visiter le réseau. En plus, les nœuds doivent stocker la clé de base K_0.

Les mécanismes basés sur plusieurs chaînes de clés à sens unique utilisent la même approche que celle d'une chaîne de clés à sens unique. Chaque chaîne de clés opère de la même façon que la chaîne unique. Ce mécanisme ne permet pas le passage à l'échelle, mais authentifie des requêtes venant de plusieurs utilisateurs à la fois. Il nécessite plus de calcul et d'espace mémoire car pour chaque chaîne, les nœuds doivent stocker la clé de base K_0.

1.3.3/ Les arbres de hachage

Les arbres de hachage ou arbre de Merkle sont des structures de données très utilisées dans les réseaux pair à pair pour assurer l'intégrité des données. Dans les RCSFs, elles peuvent être également utilisées avec les fonctions de hachage cryptographiques pour l'authentification et la distribution de la chaîne de clés [106]. Le principe est d'assigner à chacune des m chaînes de clés à sens unique un entier unique entre 1 et m. Si H est une fonction de hachage et C_i l'engagement (ou clé de base) de la *i-ème* chaîne de clés. On peut calculer K_i=H(C_i) pour tout i appartenant 1,....,m et construire l'arbre de Merkle en utilisant K_1, K_2, ...,K_m comme nœuds feuilles et que chaque nœud non feuille sera calculé par l'application de H à la concaténation de ces deux fils comme le montre l'exemple de la Figure 1.3. Ainsi, pour chaque chaîne de clés, la SB construit un engagement de certificat constitué pour la *i-ème* chaîne de clés, de C_i, et des valeurs correspondantes aux nœuds voisins frères du chemin du *i-ème* nœud feuille à la racine dans l'arbre de distribution. Dans cet exemple, le certificat d'engagement pour la deuxième chaîne est : Cert2=C_2, K_1, K_{34}, K_{58}. Une chaîne de clés et le certificat correspondant sont envoyés à un utilisateur, et la racine de l'arbre (exemple K_{18}) est envoyée à tous les capteurs. Un utilisateur qui veut accéder au réseau diffuse son certificat. Chaque capteur peut immédiatement l'authentifier à travers la racine : exemple si Cert2=C_2, K_1, K_{34}, K_{58}, un capteur peut l'authentifier en vérifiant si H(H(H(H(C_2)||K_1)||K_{34})||K_{58})=K_{18}. Pour le passage à l'échelle, ce mécanisme nécessite plusieurs

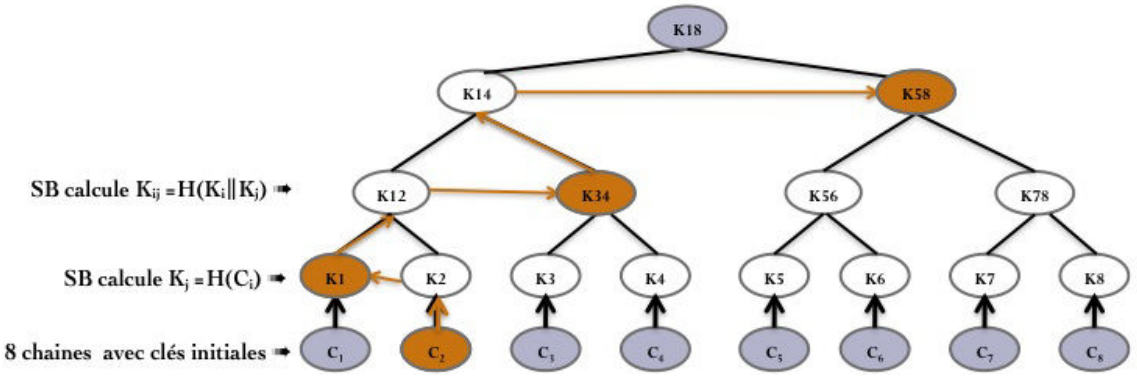


Figure 1.3 – Exemple d'arbre d'engagement et de distribution [106]

couches. L'engagement de toutes les chaînes est représenté par les feuilles des arbres de la couche la plus basse. Les racines respectives des sous arbres des couches basses représentent les feuilles de l'arbre racine de distribution rangée sur la couche haute. La SB utilise la racine de l'arbre de la couche haute pour distribuer les racines de chaque couche basse. Ces racines sont utilisées pour authentifier les certificats des chaînes de clés de la couche haute. Ainsi, à travers les certificats faits à partir de l'arbre racine de distribution, la SB distribue la racine de chaque sous arbre aux

nœuds capteurs, après une phase de pré-distribution de la racine de l'arbre racine de distribution. Ce qui permettra aux capteurs d'authentifier les engagements des chaînes de clés utilisant la racine de chaque sous arbre.

Ce mécanisme présente l'avantage de passer à l'échelle et résout les problèmes (de stockage des clés initiales K_0 et de l'authentification des chaînes de clés) rencontrés dans les mécanismes utilisant les fonctions à sens unique. En effet, le capteur a seulement besoin de stocker la racine de l'arbre utilisée car les engagements sont distribués au besoin. Nous pouvons noter l'absence du mécanisme utilisé pour distribuer la racine.

1.3.4/ Les mécanismes de contrôle d'accès au réseau

Les mécanismes de contrôle d'accès au réseau sont différents des autres. Ils sont basés sur l'architecture du RCSF, les différentes entités (capteurs ou station de base) qui le composent et les modèles de contrôle d'accés. Ainsi, on peut distinguer deux cas, suivant que la station de base intervient ou non lors du contrôle d'accès au réseau.

1.3.4.1/ Avec la station de base

Les SBs sont censées avoir plus de ressources et être mieux protégées contre les attaques que les capteurs. Par conséquent, utiliser la station de base est une approche naturelle pour organiser une telle tâche critique d'authentification de requêtes ou d'utilisateurs. On peut distinguer l'approche directe et l'approche indirecte.

- *Accès direct* : La requête d'accès au réseau est envoyée directement à la station de base via une connexion (filaire ou sans fil) ou par le routage de la requête à travers des réseaux externes (par exemple, Internet) reliés à la SB.
- *Accès distant :* La requête d'accès destinée à la SB est envoyée aux capteurs qui vont servir de relais. La SB effectue le contrôle d'accès et donne ensuite la permission au RCSF (capteurs) de répondre à la requête. Ainsi, la SB contribue à établir la confiance entre le réseau de capteurs et le monde extérieur.
- *Les avantages d'utiliser la station de base :* puisque la SB dispose de plus de ressources qu'un capteur, donc elle peut implémenter des mesures de sécurité plus fortes. Elle peut être placée dans des lieux sûrs pour être protégée contre toute attaque physique et pour détecter rapidement des attaques telles que le DdS et la pénétration.
- *Les inconvénients d'utiliser la station de base :* en tant que serveur dédié, la SB doit être protégée contre tout accès physique et distant non autorisé. Dans la littérature, il est généralement supposé que la gestion d'une SB est plus difficile que celle d'un capteur, mais dans la pratique l'inverse pourrait être le cas. Par exemple, si la station de base est connectée à un serveur web, les plus populaires avec les vulnérabilités connues, l'accès à la SB peut constituer une vulnérabilité. D'ailleurs, il n'est pas toujours possible ou souhaitable de placer une station de base dans un emplacement sécurisé et dédié, surtout si elle est censée communiquer avec les capteurs sans fil. En outre, si l'accès direct est nécesaire, ce pourrait être un inconvénient pour un utilisateur par exemple de se promener dans la zone de déploiement de la SB tout en utilisant des données provenant des capteurs à proximité de l'utilisateur. En cas d'accès distant, plusieurs messages doivent être acheminés entre la SB et l'utilisateur par le réseau de capteurs qui peut ne pas être pratique si la SB est loin. L'utilisateur pourrait également avoir à attendre la réponse de la station de base éloignée tout en ayant besoin des données de capteurs tout près. Et enfin, comme les capteurs à proximité de la station de base sont plus lourdement chargés pour la communication, leur énergie est épuisée plus rapidement, ce qui conduit à une durée de vie plus courte du réseau.

1.3.4.2/ Sans la station de base

Dans les cas où la station de base ne peut pas être utilisée pour contrôler l'accès au réseau, le mécanisme de contrôle d'accès devrait être intégré dans les capteurs. Toutefois, s'appuyant sur n'importe quel capteur quelconque, le contrôle d'accès sera difficile face aux attaques de nœuds capturés ou compromis. Une solution naturelle serait d'utiliser une approche distribuée localement ou à distance impliquant plusieurs capteurs.

- *Approche distribuée localement :* le contôle d'accés est effectué par les capteurs au voisinage de l'entité en question (utilisateurs, capteurs voulant joindre le réseau etc.). Même si les capteurs à proximité vérifient avec succès une requête, ils ont besoin de moyens supplémentaires pour dire au reste du réseau que cette requête vient d'une entité autorisée, ou être en mesure de vérifier au moins sa légitimité.
- *Approche distribuée à distance :* le contôle d'accés est effectué par des capteurs qui peuvent être spécialement choisis à cet effet. L'architecture du réseau pourrait être hétérogène avec des dispositifs de contrôle d'accès dédiés, placés dans certains endroits. Le choix des capteurs peut se faire à travers des algorithmes d'élection.
- *Les avantages (sans l'utilisation de la SB) :* les entités peuvent émettre leur requête depuis n'importe où dans le réseau sans être obligé d'aller au près de la SB. En plus les requêtes peuvent être traitées et répondues localement. Aucun routage vers la SB n'est nécessaire pour répondre à une requête même s'il peut être nécessaire de faire du routage si la requête concerne les données d'un capteur qui n'est pas à proximité de l'émetteur. Si la station de base est surchargée, les données sont encore disponibles et non compromises.
- *Les inconvénients (de ne pas utiliser la station de base) :* L'inconvénient principal est le coût de calcul et des communications si le contrôle d'accès se fait en mode distribué. Les algorithmes distribués doivent être fiables et prendre en compte l'insécurité des nœuds individuels.

1.4/ Les courbes elliptiques

Les courbes elliptiques existent depuis le 3ème siècle pour résoudre des problèmes arithmétiques anciens. Leur étude en algèbre géométrique date du milieu du 19ème siécle. La description en 1984 par Hendrik Lenstra d'un algorithme de factorisation polynomiale sur ces structures a motivé certains chercheurs à s'investir dans l'aspect cryptographique des courbes elliptiques et dans le calcul théorique des nombres [72, 73]. Des recherches ayant abouti à de nouveaux mécanismes cryptographiques basés sur l'asymétrie, notamment le logarithme discret en 1976 sur lequel est basé Diffie-Hellman (Whitfield Diffie and Martin Hellman) et la factorisation des entiers en 1977 sur laquelle est basé RSA (Ron Rivest, Adi Shamir and Len Adleman) ont donné encore plus d'inspiration [36]. C'est ainsi que Neal Koblitz et Victor Miller furent les premiers à adapter indépendamment les courbes elliptiques à la cryptographie en 1985 [64]. Leur intérêt particulier s'explique essentiellement par leur niveau de sécurité très élevé. Comparées au système RSA, on s'aperçoit que pour un même niveau de sécurité, les courbes elliptiques en cryptographie utilisent des clés de plus petites tailles. Il a été démontré qu'une clé de courbe elliptique de 160 bits fournit le même niveau de sécurité qu'une clé de 1024 bits de RSA [55, 54]. Cette qualité rend les ECCs plus attractifs pour les dispositifs aux ressources limitées telles que la capacité de calcul, de transmission/réception et de stockage mémoire qui peuvent impacter fortement sur la consommation énergétique dans le cas des capteurs.

1.4.1/ Préliminaires

La cryptographie asymétrique repose sur deux clés, une clé publique et une clé privée. La dérivation de la clé privée à partir de la clé publique revient à résoudre un problème de calcul qui doit être difficilement résolvable selon le niveau de sécurité. La difficulté de résoudre ce problème

repose sur des procédures de *fonction à sens unique avec trappe* difficilement inversible, mais si on connaît un secret (*trappe*), elle peut alors être inversée facilement. C'est ainsi qu'on peut distinguer :

– *Le problème de la factorisation des entiers sur lequel repose la sécurité du système RSA (1977) :* son principe est fondé sur la difficulté présumée du problème de la factorisation des entiers. Etant donné deux grands nombres premiers p et q, il est facile de faire la multiplication : $N=p.q$. A l'inverse, étant donné N, il est extrêmement difficile de trouver les deux nombres dont il est formé. On peut supposer qu'il s'agit d'un processus à sens unique, dans le fait que factoriser N pour retrouver p et q est très difficile. Ainsi, la sécurité de RSA repose entièrement de la difficulté présumée de la factorisation de grands nombres entiers.
Une approche naïve consisterait à essayer tous les nombres premiers les uns après les autres. Même avec un ordinateur puissant, la procédure reste fastidieuse. Dans RSA, N est la clé publique utilisée pour le chiffrement, tandis que les facteurs p et q constituent la clef secrète pour le déchiffrement.

– *Le problème du logarithme discret sur lequel repose la sécurité du système Diffie-Hellman (1976) et El-Gamal(1984) :* son principe est fondé sur la difficulté présumée du logarithme discret. Pour tout nombre premier p, il existe un entier g compris entre 1, et p-1 qui pour tout entier m compris entre 1, et p-1, il existe un entier x compris entre 0, et p-2 et un seul tel que $m \equiv g^x mod(p)$, autrement dit que m et g^x on le même reste quand on les divise par p. On dit alors que x est le logarithme discret de m dans la base g. Ainsi, étant donnés, m, g et p, il est extrêmement difficile de trouver x, cette difficulté est appelée le problème du logarithme discret.

– *Le problème du logarithme discret elliptique sur lequel repose essentiellement la sécurité de tous les mécanismes basés sur les courbes elliptiques :* son principe est fondé sur la difficulté présumée du logarithme discret elliptique. Soit P un point sur une courbe elliptique, et $Q=dP=P+P+.....+P$ (*dfois, d un entier*) aussi un point de la courbe elliptique, alors d est le logarithme discret de Q en base P. Etant donnés Q et P deux points d'une courbe elliptique, il est extrêmement difficile de trouver d tel que $Q=dP$: c'est le problème du logarithme discret elliptique.

Pour tous ces sytèmes cryptographiques, on peut faire de la génération de clés, du chiffrement avec la clé publique pour assurer la confidentialité, du déchiffrement avec la clé privée, du chiffrement avec la clé privée ou signature pour assurer l'authentification et de la vérification avec la clé publique.

1.4.1.1/ Le système RSA

1. **Génération de clés** : il s'agit de générer les clés publiques et privées. Pour la génération de la clé publique, on choisit deux grand nombres premiers p et q dont le produit est N. Ensuite, on génère les nombres entiers e (un nombre premier) et d tels que $1 < e < \phi$, le plus grand diviseur commun entre e et ϕ est 1, $\phi=(p - 1)(q - 1)$, d est tel que $1 < d < \phi$ et $(ed - 1)$ est un multiple de ϕ. N et e constituent la clé publique, tandis que la clé secrète d est construite à partir de p et q nécessaires pour son calcul.

2. **Chiffrement :** avec la clé publique (N, e), on calcule $c=m^e mod\ N$, où $m \in [0, N-1]$ est le message à chiffrer et c le message chiffré transmis.

Algorithm 1 Génération de clés avec RSA

Entrées : *l* `// taille (en bits) qui détermine le niveau de sécurité`
Sorties : Clé publique *(N,e)* et clé privée *d* générées

begin
1. Sélection au hasard de deux nombres premiers *p* et *q*
2. Calculer le produit N : *pq* et ϕ*=(p - 1)(q - 1)*
3. Sélectionner arbitrairement un entier *e* tel que *1* < *e* < ϕ et le PGDC *(e,* ϕ*)=1*
4. Calculer *d* tel que *1* < *d* < ϕ et $ed \equiv 1 (\mathrm{mod} \phi)$
5. Retourner(*N, e, d*)

Algorithm 2 Chiffrement avec RSA

Entrées : *(N,e) et m* `// la clé publique et le texte claire m ∈[0, N-1]`
Sorties : *c*, `// le texte chiffré`

begin
1. Calculer $c=m^e mod\ N$
2. Retourner(*c*)

3. **Déchiffrement :** à la réception du message chiffré *c*, on calcule $m=c^d mod\ N$.

Algorithm 3 déchiffrement avec RSA

Entrées : *(N,e), d et c* `// la clé publique, la clé privée et le texte chiffré`
Sorties : *m*, `// le texte claire`

begin
1. Calculer $m=c^d mod\ N$
2. Retourner(*m*)

4. **Signature :** le signataire calcule un condensé *h* sur le message *m* avec une fonction de hachage cryptographique H() : *h=H(m)*. Il utilise la clé privée *d* pour calculer la signature *s* : $s=h^d mod\ N$. Le signataire envoie *s* et le message *m* pour vérification.

Algorithm 4 Signature avec RSA

Entrées : *(N,e), d et m* `// la clé publique, la clé privée et le message m`
Sorties : *s*, `// la signature`

begin
1. Calculer *h=H(m)*
2. Calculer $s=h^d mod\ N$
3. Retourner(*s*)

5. **Vérification de la signature :** à la réception de la signature, l'autre partie calcule *h=H(m)*, puis à partir de la clé publique et de la signature *s* reçue, elle calcule $h'=s^e mod\ N$. Enfin elle accepte la signature si *h=h'*.

Algorithm 5 Vérification de la signature

Entrées : *(N,e), m et s* `// la clé publique, le message m, la signature s`
Sorties : Acceptation ou rejet de la signature

begin
1. Calculer *h=H(m)*
2. Calculer *h'=s^e mod N*
3. Accepter si *h=h'* et rejet sinon

1.4.1.2/ Le système du Logarithme Discret

1. **Génération de clés** : une paire de clés est associée à un ensemble de paramètres publiques (*p,q,g*) où *p* est un nombre premier, *q* un diviseur premier de *(p - 1)* et $g \in [1, p\text{-}1]$ est de l'ordre de *q*, c'est à dire que *t=q* est un petit entier positif satisfaisant $g^t \equiv 1 (\text{mod } p)$. Une clé privée *x* est sélectionnée au hasard dans l'intervalle [1, *q*-1] et la clé publique correspondante est $y=g^x mod\ p$. Le *problème du logarithme discret* consiste à déterminer *x* à partir des paramètres publiques *p, q, g* et de *y*

Algorithm 6 Génération de clés avec le Logarithme Discret

Entrées : *(p, q, g)* `// les paramètres publiques générés`
Sorties : Clé publique *(y)* et clé privée *x* générées

begin
1. Sélection au hasard *x* dans l'intervalle [1, *q*-1]
2. Calculer $y=g^x mod\ p$
3. Retourner(*x, y*)

2. **Chiffrement avec le Logarithme Discret :** le message *m* est chiffré en le multipliant par $y^k mod\ p$ où k est choisi aléatoirement par l'émetteur. Ainsi il envoie $C_2=m.y^k mod\ p$ et $C_1=g^k mod\ p$. Le récepteur à partir de sa clé privée *x* calcule $C_1^x \equiv g^{kx} \equiv y^k$ (mod p) et divise C_2 par cette quantité pour retrouver le message *m*. Le *problème de Diffie-Hellman* consiste à calculer $y^k mod\ p$ à partir des paramètres publiques (p, q, g), y, et de $C_1=g^k mod\ p$. Il est aussi difficile que le problème du logarithme discret.

Algorithm 7 Chiffrement avec ElGamal

Entrées : *(p, q, g), y et m∈[0, p-1]* `// les paramètres publiques, la clé publique et le message m ∈[0, p-1]`
Sorties : (C_1,C_2) `// le texte chiffré`

begin
1. Sélectionner k ∈[1, *q*-1]
2. Calculer $C_1=g^k mod\ p$
3. Calculer $C_2=m.y^k mod\ p$
2. Retourner(C_1, C_2)

3. **Déchiffrement avec le Logarithme Discret :** à la réception du message chiffré (C_1, C_2), le récepteur calcule $m=C_2.C_1^{-x} mod\ p$.

Algorithm 8 Déchiffrement avec ElGamal

Entrées : *(p, q, g), x et* C_1, C_2 `// les paramètres publiques, la clé privée et le message chiffré`

Sorties : *m* `// message en claire`

begin

1. Calculer m=$C_2.C_1^{-x}$ mod p
2. Retourner(m)

4. **Signature avec le Logarithme Discret (Digital Signature Algorithm proposé par NIST en 1991) :** le signataire sélectionne un entier $k \in$ [1, q-1] et calcule $T=g^k mod\ p$, $r=T mod\ q$, puis calcule $s=k^{-1}(h+xr) mod\ q$ où x est la clé privée et h est un condensé sur le message m avec une fonction de hachage cryptographique H() : $h=H(m)$. L'émetteur envoie la signature (r, s) et le message m pour vérification.

Algorithm 9 Signature avec DSA

Entrées : *(p, q, g), x et m* `// les paramètres publiques, la clé privée et le message m`

Sorties : (*r, s*) `// la signature`

begin

1. Sélectionner $k \in$ [1, q-1].
2. Calculer $T=g^k mod\ p$
3. Calculer $r=T mod\ q$. Si $r=0$, alors retour à l'étape 1
4. Calculer $h=H(m)$
5. Calculer $s=k^{-1}(h+xr) mod\ q$. Si $s=0$, alors retour à l'étape 1
6. Retourner(*r, s*)

5. **Vérification de la signature DSA :** à la réception de la signature (r, s), l'autre partie calcule $w=s^{-1} mod\ q$, $u_1=hw\ mod\ q$, $u_2=rw\ mod\ q$, $T=g^{u_1}y^{u_2}\ mod\ p$ et $r'=T\ mod\ q$. A partir de la relation $s=k^{-1}(h+xr) mod\ q$, on en déduit $k=s^{-1}(h+xr) mod\ q$, puis on calcule $T=g^{hs^{-1}}y^{rs^{-1}}$. Enfin elle vérifie si $r=r'$.

Algorithm 10 Signature avec DSA

Entrées : *(p, q, g), y, m et r, s* `// les paramètres publiques, la clé publique, le message m et la signature (r,s)`

Sorties : Acceptation ou rejet de la signature

begin

1. Vérifier si les entiers r et s sont dans l'intervalle [1, q-1]
2. Calculer $h=H(m)$
3. Calculer $w=s^{-1} mod\ q$
4. Calculer $u_1=hw\ mod\ q$ et $u_2=rw\ mod\ q$
5. Calculer $T=g^{u_1}y^{u_2}\ mod\ p$
6. Calculer $r'=T\ mod\ q$
6. Accepter si $r=r'$ et rejet sinon

1.4.2/ Les courbes elliptiques pour la cryptographie

Les courbes elliptiques sont les courbes les plus simples après les droites et les coniques. On peut considérer une courbe elliptique sur un corps fini $\mathbb{F}$, elle intervient ainsi dans certains protocoles cryptographiques. On peut définir une courbe elliptique E sur un corps fini $\mathbb{F}$ noté $E(\mathbb{F})$ par son équation à la forme de Weierstrass [4] :

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \ . \tag{1.1}$$

où a_1, a_2, a_3, a_4 *et* a_6 $\mathbb{F}$.

Les corps sont des systèmes de nombres bien connus tels que les nombres rationnels $\mathbb{Q}$ ou les nombres réels $\mathbb{R}$ et les nombres complexes $\mathbb{C}$. Un corps est un ensemble $\mathbb{F}$ possédant deux opérations élémentaires, l'addition (notée +) et la multiplication (notée .), satisfaisant les propriétés arithmétiques suivantes :
– $(\mathbb{F}, +)$ est un groupe abélien (avec l'addition) et l'élément neutre est 0.
– $(\mathbb{F}^*, .)$ est un groupe abélien (avec la multiplication) et l'élément neutre est 1.
– La distributivité est respectée : $(a+b).c = a.c + b.c$ pour tout $a,b,c \in \mathbb{F}$.
Un corps est fini quand il contient un nombre fini d'éléments, autrement dit l'ensemble $\mathbb{F}$ est fini. Il existe trois grands types de corps finis qui sont utilisables pour l'implémentation de la cryptographie pour les courbes elliptiques : les *corps premiers*, les *corps binaires* et les *corps d'extension*. L'ordre d'un corps fini est le nombre d'éléments présents dans le corps. Compter le nombre de points d'une courbe elliptique définie sur un corps fini fait partie des problématiques essentielles dans la recherche des courbes cryptographiquement sûres. La communauté des mathématiciens s'en est notamment intéressé, c'est Hasse en 1922 qui démontra le résultat sur ce nombre également appelé ordre du groupe des points d'une courbe elliptique sur un corps fini :

$$| \#E(\mathbb{F}_p) - p - 1 \leq 2\sqrt{p} | \tag{1.2}$$

Les corps premiers notés par $\mathbb{F}_p$ où $p = q^m$ et q un nombre premier (appelé la caractéristique de $\mathbb{F}_p$) sont généralement utilisés en cryptographie. Si m=1 alors $\mathbb{F}_p$ est appelé un *corps premier*. Si $m \geq 2$ alors $\mathbb{F}_p$ est appelé un *corps d'extension*. Les *corps binaires* sont des corps finis d'ordre 2^m à coefficient dans {0,1}. Dans cette thèse, nous avons travaillé avec les corps premiers $\mathbb{F}_p$, où $p > 3$. Pour ces corps premiers, si la caractéristique est supérieure à 3, l'équation de Weierstrass pour une courbe elliptique sur un corps fini premier noté $E(\mathbb{F}_p)$ peut être représentée par :

$$E : y^2 = x^3 + ax + b \ . \tag{1.3}$$

où a et b $\in \mathbb{F}_p$.

Pour être utilisée en cryptographie, la condition nécessaire est que le discriminant du polynôme soit égal à 0. Cette condition garantie que, pour tout point de la courbe elliptique, passe une et une seule tangente.

$$f(x) = x^3 + ax + b, \triangle = 4a^3 + 27b^2 \neq 0. \tag{1.4}$$

L'ensemble des points (x, y), dont les coordonnées x, y vérifient l'équation 1.1 et le point à l'infini noté ∞ sont sur la courbe et forment un groupe abélien additif $(E(\mathbb{F}_p), +)$:

$$(E(\mathbb{F}_p), +) = \{x, y\} \in \mathbb{F}_p.\mathbb{F}_p : y^2 - x^3 - ax - b = 0 \cup \{\infty\} \tag{1.5}$$

Ce groupe est principalement constitué de deux opérations de base : le doublement de point ($2P$) et l'addition de points $(P+Q)$ où P et Q sont deux points differents de la courbe.

Etant donnés $P=(x_p,y_p)$ et $Q=(x_q,y_q)$ deux points ($\neq \infty$) d'une courbe elliptique sur un corps fini $\mathbb{F}_p$ noté $E(\mathbb{F}_p)$. Géométriquement, l'addition de point (P + Q) avec P≠ Q consiste à prendre le symétrique du troisième point (P*Q) d'intersection de la droite PQ avec la courbe elliptique. Le doublement d'un point est le cas particulier d'addition où P = Q, on prend alors le symétrique du point d'intersection de la tangente en P avec la courbe elliptique. Si P et Q sont symétriques par rapport à l'axe des x, dans ce cas la droite PQ coupe la courbe au point à l'infini (qui est le zéro du groupe) et donc Q = -P.

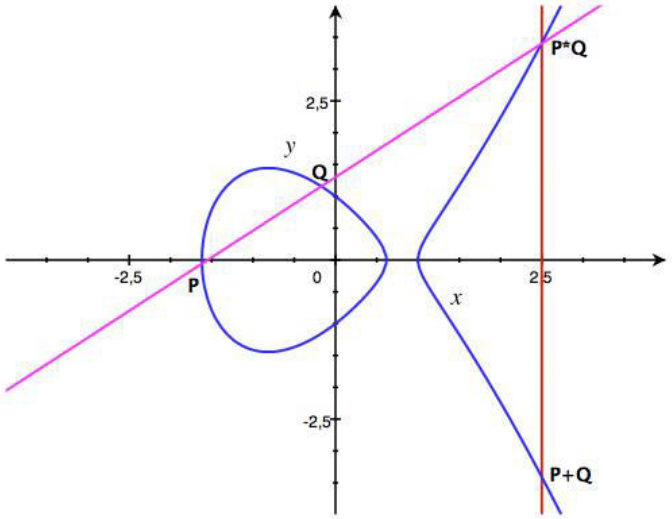


FIGURE 1.4 – Addition de points

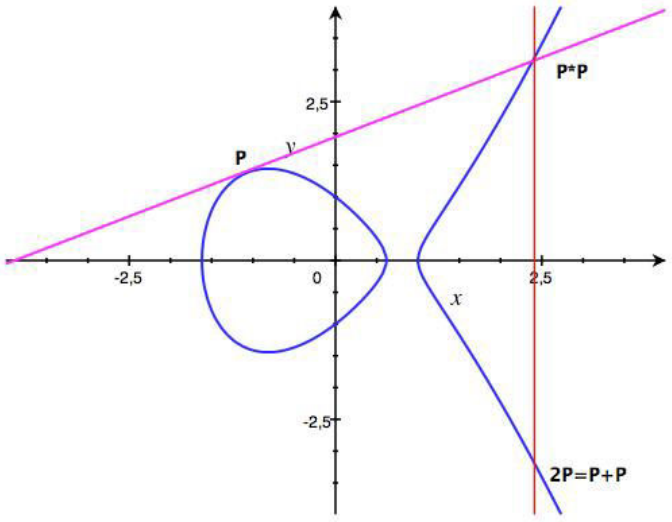


FIGURE 1.5 – Doublement de point

L'addition de points $P+Q=(x_{pq},y_{pq})$ ou le doublement de point $2P=P+Q=(x_{pq},y_{pq})$ si $P=Q$ peuvent être calculés à travers les équations 1.6 et 1.7

$$\begin{cases} x_{pq} = \lambda^2 - x_p - x_q \\ y_{pq} = \lambda(x_p - x_{pq}) - y_p \end{cases} \tag{1.6}$$

$$\begin{cases} \lambda = \dfrac{y_q - y_p}{x_q - x_p}, si\ P \neq Q \\ \lambda = \dfrac{3x_p^2 + a}{2y_p}, si\ P = Q \end{cases} \tag{1.7}$$

1.4.2.1/ Le sytème du Logarithme Discret Elliptique

Si P est un point de la courbe elliptique $E(\mathbb{F}_p)$ d'ordre n, et Q un autre point de la courbe, la difficulté de trouver l'entier $d \in$ [0, n-1] tel que $Q=dP$ est appelé : le problème du logarithme discret elliptique. L'entier d est appelé logarithme discret de Q en base P noté $d=log_pQ$. Soit une courbe elliptique sur un corps fini $E(\mathbb{F}_p)(a,b)$ obtenue avec l'équation 1.8 pour $0<x<p$ et a,b$\geq$0 et $< p$.

$$y^2 = x^3 + ax + b\ mod\ p \tag{1.8}$$

Soit P un point d'ordre n (un nombre premier), le groupe cyclique généré par P et noté $<P>$ = { ∞, P, 2P, 3P, ... ,(n-1)P }. Ainsi le nombre premier p, l'équation de la courbe elliptique E, le point P et son ordre n sont les paramètres publiques des cryptosystèmes. La clé privée est un entier d sélectionné uniformément au hasard dans l'intervalle [1, n-1] et la clé publique correpondante est le point $Q=dP$

1. **Génération de clés avec le Logarithme Discret Elliptique** : une paire de clés est associée à un ensemble de paramètres publiques (p, E, P, n) où p est un nombre premier, E une courbe elliptique, P le point générateur et n son ordre, c'est à dire que $nP=\infty$. Une clé privée d est sélectionnée au hasard dans l'intervalle [1, n-1] et la clé publique correspondante est $Q=dP$. Le *problème du logarithme discret* consiste à déterminer d à partir des paramètres publiques (p, E, P, n).

Algorithm 11 Génération de clés avec le Logarithme Discret Elliptique

Entrées : p, E, P, n `// les paramètres publiques générés`
Sorties : Clé publique (Q) et clé privée d générées

begin
1. Sélectionner au hasard d dans l'intervalle [1, n-1]
2. Calculer $Q=dP$
3. Retourner(Q, d)

2. **Chiffrement avec le Logarithme Discret Elliptique (ElGamal) :** le message m se présente sous forme d'un point M, il est chiffré en l'additionnant au point de la courbe kQ où k est un entier choisi aléatoirement par l'émetteur, et Q est un point représentant la clé publique du récepteur. Ainsi, l'émetteur envoie $C_2=M+kQp$ et $C_1=kP$ au récepteur qui, à partir de sa clé privée d calcule $dC_1=d(k)P=k(dP)=kQ$ et retrouve $m=C_2-kQ$.

Algorithm 12 Chiffrement ElGamal avec les courbes elliptiques

Entrées : *(p, E, P, n)*, *Q* et *m* `// les paramètres publiques, la clé publique et le message clair m`

Sorties : (C_1,C_2) `// le texte chiffré`

begin

1. Représenter le message *m* en tant que point M ∈ $E(\mathbb{F}_p)$
2. Sélectionner k ∈[1, *n*-1]
3. Calculer C_1=*kP*
4. Calculer C_2=*M*+*kQ*
5. Retourner(C_1, C_2)

3. **Déchiffrement avec le Logarithme Discret (ElGamal) :**

Algorithm 13 Déchiffrement ElGamal avec les courbes elliptiques

Entrées : *(p, E, P, n), d et* C_1, C_2 `// les paramètres publiques, la clé privée et le message chiffré`

Sorties : *m* `// message en clair`

begin

1. Calculer *M*=C_2 - *d*C_1, et extraire *m* de M
2. Retourner(*m*)

4. **Signature avec le Logarithme Discret Elliptique :** le signataire sélectionne un entier $k \in [1, n\text{-}1]$ et calcule le point *kP* de coordonnées (x_1, y_1), soit $r=x_1$. Il calcule k^{-1}mod*n* puis $s = k^{-1}(h + dr)$ où *d* est la clé privée et *h* est un condensé sur le message *m* avec une fonction de hachage cryptographique H() : *h=H(m)*. L'émetteur envoie la signature *(r, s)* et le message *m* pour vérification.

Algorithm 14 Signature avec les courbes elliptiques

Entrées : *(p, E, P, n), d et m* `// les paramètres publiques, la clé privée et le message m`

Sorties : (*r, s*) `// la signature`

begin

1. Sélectionner $k \in [1, n\text{-}1]$.
2. Calculer *kP* de coordonnées (x_1, y_1), soit $r=x_1$. si *r modn=0* retour à l'étape1
3. Calculer k^{-1} mod n
4. Calculer *h=H(m)*
5. Calculer $s = k^{-1}(h + dr)$. Si *s=0*, alors retour à l'étape 1
6. Retourner(*r, s*)

5. **Vérification de la signature DSA :** à la réception de la signature *(r, s)*, l'autre partie calcule $w=s^{-1} mod\ n$, $u_1=hw\ mod\ n$, $u_2=rw\ mod\ n$ puis u_1P+u_2Q et obtient comme résultat un point de coordonnées (x_2, y_2). La signature est vérifiée si $r=x_2$.

Algorithm 15 Vérification de signature avec les courbes elliptiques

Entrées : *(p, E, P, n), Q, m et (r, s)* `// les paramètres publiques, la clé publique, le message m et la signature (r,s)`

Sorties : Acceptation ou rejet de la signature

begin

1. Calculer $w=s^{-1} mod\ n$ et $h=H(m)$
2. Calculer $u_1=hw\ mod\ n$ et $u_2=rw\ mod\ n$
3. Calculer u_1P+u_2Q et obtenir le point de coordonnées (x_2, y_2)
4. Accepter si $r=x_2$, et rejet sinon

Le problème du logarithme discret sur une courbe elliptique bien choisie est plus rapide que celui du logarithme discret dans les corps finis. L'algorithme connu comme étant le plus efficace pour résoudre un tel problème est à temps exponentiel, contrairement au système RSA pour lequel il existe des algorithmes à temps sous-exponentiel. Un algorithme est sous-exponentiel si le logarithme du temps d'exécution croît asymptotiquement moins vite que tout polynôme donné. Le niveau de sécurité d'une clé est en directe correspondance avec sa taille. Ainsi, pour une même résistance, les courbes elliptiques requièrent des clés plus courtes que RSA. Or l'usage de clés de petites tailles confère beaucoup d'avantages : les calculs sont plus rapides, la consommation électrique globale est diminuée, et l'espace mémoire est réduit. Contrairement à RSA, les courbes elliptiques sont plus rapides pour les opérations privées que pour les opérations publiques. RSA demande moins de temps en chiffrement et vérification de signature numérique qu'en déchiffrement et la génération de signature qui sont lents et demandent plus de ressources. Au contraire, les courbes elliptiques en cryptographie nécessitent plus de calcul lors du chiffrement et vérification de la signature que le déchiffrement et la génération de la signature. Néanmoins, non seulement le chiffrement et la vérification sont efficaces, mais également le déchiffrement et la génération de signature. Ainsi, elles sont pratiques dans les environnements à fortes contraintes de ressources tels que les réseaux de capteurs, les cartes à puces, les téléphones mobiles, etc.. Les ECCs sont appelées à remplacer progressivement RSA [102]. Le Tableau 1.1 présente une comparaison en termes de taille de clés des cryptosystèmes ECC et RSA pour le même niveau de sécurité.

Clés RSA (bits)	*Clés ECC(bits)*
1024	160
2048	224
3072	256
7680	384
5360	521

TABLE 1.1 – Comparaison entre ECC et RSA (paramètres de NIST)

1.4.3/ ARITHMÉTIQUE DES ECCS POUR LA MULTIPLICATION SCALAIRE

La hiérarchie mathématique des courbes élliptiques implique trois niveaux arithmétiques [55] : sur le corps fini, sur un point de la courbe ou sur le scalaire k quand il est multiplié par un point de la courbe.

1.4.3.1/ LE NIVEAU CORPS FINI

Ce niveau est relatif à l'arithmétique modulaire sur le corps fini, où celle-ci comprend un ensemble d'entiers sur lequel les opérations arithmétiques : l'addition, la soustraction, la multiplication,

l'inversion et le carré. Ces opérations sont calculées modulo un nombre p (premier pour les *corps finis premiers*).

- *Addition* : soient $a, b \in \mathbb{F}_p$, $(a+b) mod\ p = r$, où r est le reste de la division entière de $a+b$ par p, $0 \leq r \leq p-1$.
- *Soutsraction* : soient $a, b \in \mathbb{F}_p$, $(a-b) mod\ p = r$, où r est le reste de la division entière de $a-b$ par p, $0 \leq r \leq p-1$. Cette opération peut être remplacée par une addition de a et de $(-b)$, où b est l'unique élément tel que $b+(-b)=0$. On appelle b la négation de $(-b)$
- *Multiplication* : soient $a, b \in \mathbb{F}_p$, $(a.b) mod\ p = r$, où r est le reste de la division entière de $a.b$ par p, $0 \leq r \leq p-1$.
- *Inversion* : soit $a \neq 0 \in$ et $\mathbb{F}_p$, $(a^{-1}) mod\ p = r$, est l'unique entier $r \in \mathbb{F}_p$ pour lequel $(a.r) mod\ p = 1$.
- *Carré* : soient $a \in \mathbb{F}_p$, $(a^2) mod\ p = r$, où r est le reste de la division entière de a^2 par p, $0 \leq r \leq p-1$. Dans certaines implémentations, cette opération est remplacée par la multiplication $a.a$.

1.4.3.2/ Le niveau point

Ce niveau concerne les opérations sur les points principalement l'addition et le doublement de points dont les constructions géométriques sont représentées dans les Figure 1.4 et 1.5 , et les formules en coordonnées affines par les équations 1.6 et 1.7.

Soient $P=(x_p,y_p)$ et $Q=(x_q,y_q)$ deux points différents sur la courbe elliptique $E(\mathbb{F}_p)$ avec $p > 3$:

- L'addition de points $P+ Q= (Q+ P)$ est un troisième point de la courbe elliptique.

- Le doublement d'un point $P+ P= [2]P$ est un point de la courbe elliptique.

Il existe d'autres opérations comme par exemple le triplement d'un point, le quadruple d'un point, le quintuple d'un point etc. :

- Le triplement d'un point : $P+ P + P=[3]P$.
- Le quadruple d'un point : $P+ P + P+P=[4]P$

L'efficacité en termes de temps de calcul de ces opérations dépend de plusieurs paramètres comme par exemple :

- *La succession des opérations de points* : pour le quadruple d'un point par exemple on peut faire un doublement suivi de deux additions (((2P) + P)+P), ou un doublement et une addition ((2P) +(2P)).
- *La formule* : la formule qui contiendra moins d'opérations arithmétiques sera sans doute plus efficace.
- *Le système de coordonnées* : prenons l'exemple sur les coordonnées affines et jacobiennes.
  En *coordonnées affines*, un point de la courbe elliptique $E(\mathbb{F}_p)$ est représenté par le couple (x,y) ; il n'y a pas de représentation du point à l'infini. Les coordonnées affines sont moins efficaces car elles contiennent des opération d'inversions qui sont plus coûteuses. Le coût d'une addition de points est $1I + 1M + 1S$ avec *I, M et S* étant respectivement l'*inversion*, la *multiplication* et le *carré*. Et celui d'un doublement de point est $1I + 1M + 2S$.
  En *coordonnées jacobiennes* : on évite l'opération d'inversion en ajoutant une troisième coordonnée Z. Un point de la courbe elliptique $E(\mathbb{F}_p)$ en coordonnées projectives est représenté par le triplet (X, Y, Z) à travers la relation d'équivalence :
  (X : Y : Z)={($\lambda^c X$, $\lambda^d Y$, λZ) : λ un élément du corps fini, (c et d) $\in \mathbb{Z}^+$.
  En coordonnées projectives jacobiennes c=2 et d=3, le point à l'infini est représenté par ∞(1 : 1 : 0).
  Soient $P(X_p, Y_p, Z_p)$ et $Q(X_q, Y_q, Z_q)$ deux points de la courbe elliptique en coordonnées jacobiennes, les équations 1.9 et 1.10 représentent respectivement les formules d'addition de points $P + Q=(X_{pq}, Y_{pq}, Z_{pq})$ et de doublement $2P=(X_{2p}, Y_{2p}, Z_{2p})$ de point en coordonnées jacobiennes. Ainsi, le coût d'une addition de point est 12 *multiplications* + 4 *carrés* et celui d'un

doublement de point est 4 *multiplications*+ 6 *carrés*. Comme dans les formules d'addition et de doublement il n'y a donc plus d'inversion modulaire à calculer pour effectuer l'addition et le doublement de point.

$$\begin{cases} X_{pq} = \alpha^2 - \beta^3 - 2Z_q^2 X_p \beta^2 \\ Y_{pq} = \alpha(Z_q^2 X_p \beta^2 - X_{pq}) - Z_q^3 Y_p \beta^3 \\ Z_{pq} = Z_p Z_q \beta \\ \alpha = Z_p^3 Y_q - Z_q^3 Y_p \\ \beta = Z_p^2 X_q - Z_q^2 X_p \end{cases} \tag{1.9}$$

$$\begin{cases} X_{2p} = \alpha^2 - 2\beta \\ Y_{2p} = \alpha(\beta - X_{2p}) - 8Y_p^4 \\ Z_{2p} = 2Y_p Z_p \\ \alpha = 3X_p^2 + aZ_p^4 \\ \beta = 4X_p Y_p^2 \end{cases} \tag{1.10}$$

Il existe d'autres paramètres comme l'équation de la courbe, les paramètres de la courbe etc..

1.4.3.3/ Le niveau scalaire

Le troisième niveau repose sur des développements binaires pour une représentation optimale du scalaire k afin d'accélérer le calcul de la multiplication scalaire qui est l'opération fondamentale à effectuer dans le cadre de l'ECC. Les possibilités de calcul offertes par les courbes elliptiques sont : l'addition de points, le doublement de point, la multiplication scalaire (multiplication d'un point P de la courbe par un entier k noté $[k]P$), le calcul de l'opposé d'un point de la courbe et le calcul du point à l'infini. La multiplication scalaire est incontournable dans l'usage des courbes elliptiques pour la cryptographie. Les algorithmes surtout de signature numérique effectuent cette opération si souvent qu'on peut se demander si la cryptographie avec les ECCs ne repose pas essentiellement sur cette opération. Dans la suite, nous allons décrire quelques méthodes pour effectuer rapidement $Q = [k]P$ (où P est un point de la courbe elliptique, et k un entier appelé la clé).

1.4.4/ Algorithmes d'accélération de la multiplication scalaire efficaces

Beaucoup de méthodes de calcul efficaces de la multiplication scalaire existent dans la littérature [50]. Dans cette section, nous allons d'abord présenter un algorithme de base appelé *Doublement-et-Addition* (ou Double-and-Add) pour calculer $Q = [k]P$ puis des raffinements de celui-ci.

1.4.4.1/ Algorithme du doublement-et-addition (DA)

L'algorithme du doublement-et-addition (DA) est une représentation additive des algorithmes utilisés pour l'exponentiation. Comme on peut le voir sur les algorithmes 16 et 17 où le scalaire k est représenté en binaire sur l bits : $k=\sum_{i=0}^{l-1} k_i 2^i$ où $k_i \in \{0, 1\}$. Cette méthode consiste à parcourir les bits du scalaire $[k]P= \underbrace{(P + P + P + ... + P)}_{kfois}$ en partant du bit de poids fort vers le bit de poids faible ou inversement. Un doublement est effectué pour chaque bit k_i de k, suivi d'une addition pour chaque bit non nul ($k_i \neq 0$).

Algorithm 16 Doublement-et-Addition bit faible/bit fort

input : $k=(k_{l-1},.....,k_1,k_0)_2$, P $\in E(\mathbb{F}_p)$
output : $Q=[k]P$

begin
  $Q \longleftarrow \infty$;
  for $i \leftarrow 0$ **to** *l-1* **do**
    // début du balayage bit par bit de bit faible vers bit fort
    if $k_i=1$ **then**
      $Q \longleftarrow Q + P$ // On effectue une addition
    $P \longleftarrow 2P$ // On effectue un doublement
  return (Q)

Algorithm 17 Doublement-et-Addition bit fort/ bit faible

input : $k=(k_{l-1},.....,k_1,k_0)_2$, P $\in E(\mathbb{F}_p)$
output : $Q=[k]P$

begin
  $Q \longleftarrow \infty$;
  for $i \leftarrow l-1$ **to** *0* **do**
    // début du balayage bit par bit de bit fort/ vers bit faible
    $Q \longleftarrow 2Q$ // On effectue un doublement
    if $k_i=1$ **then**
      $Q \longleftarrow Q + P$ // On effectue une addition
  return (Q)

Pour un scalaire k donné, le nombre d'opérations de doublements est *(l-1)* et celui d'opérations d'additions est égal au nombre de bits non-nuls (noté h) -1. Avec cette méthode, la densité moyenne des bits non-nuls (bits à 1) sur l'ensemble de tous les scalaires k de longueur l bits est approximativement *l/2*. Ainsi, les algorithmes 16 et 17 effectuent en moyenne *(l-1) doublements* et *(l-1)/2 additions*. Par exemple, pour un scalaire $k = 379 = (101111011)_2$, l=9 et le nombre de bits non-nuls h=7. Alors on exécute 8 doublements et 6 additions.

Cette méthode *Doublement-et-Addition* peut être généralisée en utilisant des fenêtres de taille fixe ou variable. Le principe consiste à diviser le scalaire k en m blocs de w bit(s) (w un entier de taille fixe ou variable), à chaque bloc de bits correspond un nombre V_i.

Comme dans *DA* où l'on traite 1 bit par 1 bit, et si le bit traité vaut 0 on effectue Q=2^1Q (doublement), sinon (bit traité égal à 1 > 0) on effectue Q=2^1Q (doublement) puis Q=Q + 1P (addition). Dans la méthode par fenêtres, on traite bloc par bloc et si la valeur du bloc est égal à 0 on effectue Q=2^wQ, sinon (valeurs des w bits traités = V_i) on effectue Q=2^wQ puis Q=Q + V_iP comme le montre l'algorithme 18. Par exemple, pour un scalaire $k = 379 = (101111011)_2$ partionné en blocs $\underbrace{1011}_{w=4}\ \underbrace{110}_{w=3}\ \underbrace{11}_{w=2}$. Ainsi, V_i prend respectivement les valeurs 3, 6 et 11 correspondants respectivement aux points précalculés 3P, 6P et 11P. Ainsi, pour cet exemple, la multiplication scalaire du bloc *(m-1)* vers le bloc 0 peut se dérouler de la manière suivante : [11]P $\rightarrow 2^3$.[11]P(3 doublements répétés) + [6]P(addition) $\rightarrow 2^2$.[94]P(2 doublements répétés) + [3]P(addition) $\rightarrow$ [379]P. Ainsi, on a effectué 5 doublements et 2 additions.

Algorithm 18 Méthode par fenêtres

input : $k=(\underbrace{k_{l-1}k_{l-2}k_{l-3}k_{l-4}}_{bloc\ (m-1)}, ..., \underbrace{k_5k_4k_3}_{bloc1}, \underbrace{k_2k_1k_0}_{bloc0}$, P $\in E(\mathbb{F}_p)$

output : $Q=[k]P$

```
begin
    Q ⟵ ∞ ;
    for i ← m-1 to 0 do
                                                  // début du balayage bloc par bloc
        Q ⟵ 2^w Q
        // On effectue un doublement de point répété w fois
        if V_i>0 then
            Q ⟵ Q + V_i P  // On effectue une addition avec le point précalculé
            V_i P
    return (Q)
```

Cependant, ce mécanisme fait intervenir des points pré-calculés dont le nombre dépend de la taille des blocs. Si les blocs sont de taille fixe de w bits, le nombre de points pré-calculés est (2^w -2) où le -2 représente les blocs pour V_i= 0 ou 1. Si les blocs sont de taille variable, comme par exemple avec 3 blocs de w_1 bits, w_2 bits et w_3 bits, le nombre de points pré-calculés reste le même (2^w -2). On peut remarquer que, utiliser la méthode des fenêtres réduit le temps de calcul et augmente le stockage mémoire et le temps de calcul des points précalculés. Quand la taille des blocs augmente, le nombre de points précalculés croit exponentiellement et le nombre d'opérations à effectuer diminue. Ainsi, la sélection de la fenêtre implique le temps de calcul. Un compromis doit être fait entre la taille des blocs et le temps de calcul des points pré-calculés. Selon les recommandations de NIST, la meilleure longueur de fenêtre est w=4.

Pour réduire le nombre de points précalculés, la méthode des fenêtres glissantes de taille variable avec un maximum de chiffres égal à w peut être utilisée. Pour cette méthode, les valeurs V_i des blocs sont impairs, les 0 consécutifs sont sautés. Par conséquent une fenêtre commence et se termine par un chiffre non-nul. Par exemple pour un scalaire k = 379 = $(101111011)_2$ partionné en blocs $\underbrace{1}_{w=4}\ \underbrace{1111}_{w=3}\ \underbrace{11}_{w=2}$. Ainsi, V_i prend respectivement les valeurs 1, 15 et 3 correspondantes respectivement aux deux points précalculés 15P et 3P. La multiplication scalaire du bloc *(m-1)* vers le bloc 0 peut se dérouler de la manière suivante : P → [2]P(1 doublement) → 2^4[62]P(4 doublements répétés) + [15]P(addition)→ 2.[47]P(1 doublement) → 2^2[94]P(2 doublements répétés)+ [3]P(addition) → [379]P. Ainsi, on a effectue 8 doublements et 2 additions.

Une optimisation possible consiste à trouver une représentation qui contient le plus petit nombre de bits non-nuls afin de réduire le nombre d'opérations d'additions : c'est l'objectif des solutions qu'on va décrire dans la suite de cette section.

1.4.4.2/ Algorithme de la forme non-adjacente (NAF)

Tout comme l'addition, la soustraction des points d'une courbe elliptique est aussi efficace surtout quand il s'agit du calcul facile de l'opposé d'un point où on change seulement une coordonnée : *P (x, y)* en *-P(x, -y).* On peut utiliser une représentation signée de bits de l'entier k. Une des représentations particulièrement intéressante est la forme non-adjacente (Non-Adjacent Form, NAF) qui utilise les chiffres {-1, 0, 1} : $k=\sum_{i=0}^{l-1} k_i 2^i$ où $k_i \in$ {-1, 0, 1}. Le principe de calcul de la multiplication *[k]P* consiste à parcourir les chiffres (du poids fort vers le poids faible) du scalaire représenté sous sa forme NAF, un doublement est effectué pour chaque chiffre de k, et pour

chaque chiffre non nul on effectue une addition lorsque le chiffre vaut 1 ou une soustraction lorsque le chiffre vaut -1. L'avantage de cette représentation est qu'elle possède les propriétés suivantes :

1. Le scalaire k possède une représentation NAF unique notée NAF (k).
2. NAF(k) possède le moins de bits non nuls parmi toutes les représentations signées de bits de l'entier k.
3. Le nombre de chiffres de NAF(k) est au plus égal au nombre de bits de la représentation binaire de k plus un.

Par exemple, si on prend le cas $k = 255_2 = (11111111)_2$ où la densité des bits non-nuls est maximale, le calcul de $255P$ nécessite 7 additions de points. Mais si on le transforme en $256P$ -P qui est égal à (10000000 -1)P, une seule addition sera nécessaire. Ainsi, le NAF(k) = $(10000000\bar{1})_2$ où $\bar{1}$ représente -1. Le NAF(k) d'un scalaire peut être généré en divisant successivement le scalaire k par 2. Si k est impair, le reste $r \in$ {-1, 1} est choisi de telle sorte que le quotient (k-r)/2 soit pair. Ainsi, le prochain bit de la représentation sera à 0.

Algorithm 19 Calcul du NAF d'un scalaire k

input : *Le scalaire k (entier)*
output : *NAF(k)*

```
begin
    i ⟵ 0 ;
    while (k≥1) do
        if (k impair) then
            k_i ⟵ 2 − (k mod 4);
            k ⟵ k_i;
        else
            (k_i ⟵ 0
        k_i = k/2;
        i ⟵ i + 1;
    return (k_{i−1}, k_{i−1}, ....k_1, k_0)
```

A partir de l'algorithme Doublement-et-Addition bit fort/bit faible, l'algorithme 19 calcule la multiplication scalaire en utilisant la représentation NAF(k).

Algorithm 20 Méthode NAF

input : *NAF(k),P* $\in E(\mathbb{F}_p)$
output : *Q = [k]P*

```
begin
    Q ⟵ ∞ ;
    for i ← l-1 to 0 do
            // début du balayage chiffre par chiffre du chiffre fort vers
        chiffre faible
        Q ⟵ 2Q                                  // On effectue un doublement
        if (k_i = 1) then
            Q ⟵ Q + P;                           // On effectue une addition
        if (k_i = −1) then
            Q ⟵ Q − P;                          // On effectue une soustraction
    return (Q)
```

Ainsi, la densité moyenne des chiffres non-nuls (chiffre à -1 ou 1) sur l'ensemble de toutes les NAF (k) de longueur *(l-1)* chiffres est approximativement *(l-1)/3*. L'algorithme 20 effectue en moyenne *(l-1) doublements* et *(l-1)/3 additions*. Cependant, il nécessite un temps de conversion du scalaire k en NAF(k) (voir algorithme 19).

Cette méthode NAF peut être généralisée, en utilisant un ensemble de chiffres ($C_{2^w} = \{-2^{w-1}, ..., 2^{w-1}\}$) pour représenter le scalaire k. Ce qui revient à le découper en fenêtres de longueur fixe w. Par exemple (C_{2^3} = {-4, -3, -2, -1, 0, 1, 2, 3, 4}). On peut ainsi définir le NAF_w(k) suivant la formulation :

$$NAF_w(\mathrm{k}) = \sum_{i=0}^{l} k_i 2^i P, \text{ avec} |k_i| < 2^{w-1}$$

Par exemple : si le scalaire $k = 379 = (101111011)_2$, alors il peut être calculé NAF_2(k), NAF_3(k), NAF_4(k) (avec -1=$\bar{1}$) :

1. NAF_2(k) =(1 0 $\bar{1}$ 0 0 0 0 $\bar{1}$ 0 $\bar{1}$)
2. NAF_3(k) =(3 0 0 0 0 $\bar{1}$ 0 0 3)
3. NAF_4(k) = (3 0 0 0 0 0 0 $\bar{5}$)

L'algorithme 21 présente la méthode du *Doublement-et-Addition* utilisant une forme non- adjacente du scalaire traité par fenêtres de longueur fixe.

Algorithm 21 Méthode binaire NAF avec fenêtres de longueur fixe

input : *NAF(k),P* $\in E(\mathbb{F}_p)$*, les points précalculés [j]P pour j ={1, 3,..., (*2^{w-1}*-1)}*
output : *Q = [k]P*

```
begin
    Q ⟵ ∞ ;
    for i ← l-1 to 0 do
        // début du balayage chiffre par chiffre de poids fort vers poids faible
        Q ⟵ 2Q                                   // On effectue un doublement
        if (k_i ≠ 0) then
            if (k_i > 0) then
                Q ⟵ Q + [k_i]P;                  // On effectue une addition
            else
                Q ⟵ Q − [k_i]P;                  // On effectue une soustraction
    return (Q)
```

La densité moyenne des chiffres non-nuls sur l'ensemble de toutes les NAF (k) de longueur l chiffres est approximativement *l/(w+1)*. Ainsi, l'algorithme 21 effectue en moyenne *(l-1) doublements* et *l/(1+w) additions*. Cependant, cette méthode induit les points précalculés [j]P pour j = 1, 3, . . ., 2^{w-1}-1. Malgré le coût de ces points pré-calculés (1doublement + (2^{w-2} -1)additions), l'utilisation du $NAF_w(k)$ avec fenêtre reste plus intéressante que celle sans fenêtre.

Une dernière généralisation de cette méthode consiste à utiliser $NAF_w(k)$ avec des fenêtres de longueur variable (ou glissante) ayant un nombre maximum de chiffres. Ces fenêtres commencent et se terminent par un chiffre différent de 0.

Si on prend l'exemple de NAF_2(k) =(1 0 $\bar{1}$ 0 0 0 0 $\bar{1}$ 0 $\bar{1}$) avec des fenêtres glissantes ayant une longueur maximale de 3 chiffres, ces fenêtres commencent et se terminent par un chiffre différent de 0. On obtient donc :

$$NAF_2(\mathrm{k}) = (\underline{1\ 0\ \bar{1}}\ 0\ 0\ 0\ 0\ \underline{\bar{1}\ 0\ \bar{1}})$$

Les points précalculés sont [3]P et [5]P, la multiplication scalaire s'effectue comme suit : [3]P → [6]P(doublement) → [12]P(doublement) → [24]P(doublement) → [48]P(doublement) → [96]P(doublement) → [192]P(doublement) → [384]P(doublement) → [379]P(soustracition de -[5]P). Ainsi, on a effectue 8 opérations de points, contre 12 dans le cas où les fenêtres sont fixes.

1.4.4.3/ Algorithme de la forme mutuelle opposée (MOF)

Des mécanismes plus récents comme la forme mutuelle opposée (Mutual opposite form, MOF) [84] et le re-codage par complément [27] utilisent aussi la représentation signée des bits {-1, 0, 1}.

Dans MOF, la représentation du scalaire k est obtenue en faisant une soustraction de chaque bit k_{i-1} avec celui de k_i. Le bit de poids fort est à 1 et celui du poids faible est à -1. Sa sortie est comparable à celle de NAF. Par exemple : si le scalaire $k = 379 = (101111011)_2$, alors il peut être calculé MOF(k)=1 $\bar{1}$ 1 0 0 0 $\bar{1}$ 1 0 $\bar{1}$. La conversion est plus simple que celle de NAF car elle ne nécessite que des opérations de soustraction. En plus MOF peut scanner les bits de gauche à droite ou inversement, ce qui est plus flexible.

1.4.4.4/ Algorithme du re-codage par complément à 1 (RC1)

Dans la méthode du re-codage par complément à 1, la représentation du scalaire k est obtenue à travers son complément $\bar{k}$: $\sum_{i=0}^{l-1} k_i 2^i = 2^l - \bar{k} - 1$. Le complément $\bar{k}$ s'obtient en inversant chaque bit du scalaire k. Par exemple : si le scalaire $k = 379 = (101111011)_2$, alors il peut être calculé : $k = 2^9 - \bar{k} - 1 = (1000000000 - 010000100 - 1)_2 = (10\bar{1}0000\bar{1}00 - 1)_2$. Ainsi, on peut constater que la densité des bits non-nuls est réduite de 7 à 4. Cependant, si le nombre de 1 dans le scalaire k original est supérieur à $l/2$, la méthode n'est plus interessante car l'objectif est d'avoir le moins de 1 dans la représentation finale.

1.4.4.5/ Le système de numération à double base

Dans les méthodes exposées précédemment, le scalaire est représenté dans une seule base, le système de numération à double base propose une représentation dans deux bases (Double-Base Number System, DBNS) [38]. Le scalaire k est représenté comme une somme de puissances combinées de 2 et 3 : $k = \sum_{i=l}^{l'} k_i 2^{a_i} 3^{b_i}$ où $k_i \in \{-1, 1\}$ et $a_i, b_i \geq 0$. L'usage direct de ce système peut induire un coût de calcul élevé : $\sum_i b_i triplements$, $\sum_i a_i doublements$.

Une amélioration significative permet de réduire les coûts en réutilisant tous les calculs intermédiaires [37]. On garde la représentation initiale de k avec la contrainte supplémentaire que les exposants forment deux suites décroissantes : $a_{max} \geq a_1 \geq a_2 \geq .....a_{l'}$ et $b_{max} \geq b_1 \geq b_2 \geq .....a_{l'}$. Cette formulation permet de ne calculer que a_{max} *doublements*, b_{max} *triplements* et (l'-1) *additions*.

Exemple : $752 = 2^3 \mathrm{x} 3^4 + 2^2 \mathrm{x} 3^3 - 2^2$
La multiplication scalaire s'effectue comme suit : $2^2(3^3(2 \text{ x } 3\mathrm{P} + \mathrm{P}) - \mathrm{P})$. Ainsi, le coût de la multiplication scalaire est de 4 *triplements* + 3 *doublements* + 2 *additions*. Cette approche a été généralisée en utilisant un espace de chiffres un peu plus large nécessitant ainsi des points précalculés [39]. Dans ce cas les valeurs de k_i sont des nombres premiers différents de 3 : {±1, ±5, ±7, ±11}.

1.4.4.6/ Comparaison

S'il y a de la mémoire de stockage disponible, on peut utiliser les points précalculés pour diminuer le temps de calcul. La méthode par fenêtre ou bloc peut être utilisée différemment sur les représentations signées telles que NAF, MOF, le codage par complémentation, ou sur les représentations non signées tel que doublement-et-addition. Si on s'intéresse à la représentation par fenêtre glissante, le nombre de points précalculés varie selon les méthodes. Prenons l'exemple des fenêtres de longueur variable (glissante) ayant un nombre maximum de 5 chiffres.

Pour la méthode du doublement-et-addition, on aura toutes les combinaisons impaires d'au maximum de 5 bits, c'est à dire qui commencent et se terminent par un 1. On aura ainsi au maximum 15 points précalculés : [3]P, [5]P, [7]P, [9]P, [11]P, [13]P, [15]P, [17]P, [19]P, [21]P, [23]P, [25]P, [27]P, [29]P, [31]P.

Pour la méthode wNAF, les blocs sont traités par fenêtres de longueur variable (ou glissante) ayant un nombre maximum de 5 chiffres. Ces fenêtres commencent et terminent par un chiffre non-nul. Par conséquent la valeur V_i de chaque bloc du scalaire k est impair et est inférieur à 2^w. Il n'y a pas deux chiffres non-nuls consécutifs, donc le nombre de zéros est au moins égal au nombre de chiffres nuls dans le bloc -1. Le nombre de points précalculés maximal nécessaire est de (2^{w-2})-1. Si la longueur maximale de la fenêtre est de 5 bits, le plus grand point précalculé correspondant est $\underbrace{10101}_{w=5}$ = 21P, et les combinaisons possibles pour les points précalculés sont les suivantes :

w=3bits	w=4bits	w=5bits	w=5bits
1 0 1 = 5P	1 0 0 1 = 9P	1 0 1 0 1 = 21P	$\bar{1}$ 0 $\bar{1}$ 0 $\bar{1}$ = -21P
1 0 $\bar{1}$ = 3P	1 0 0 $\bar{1}$ = 7P	1 0 1 0 $\bar{1}$ = 19P	$\bar{1}$ 0 $\bar{1}$ 0 1 = -19P
$\bar{1}$ 0 1 = -3P	$\bar{1}$ 0 0 1 = -7P	1 0 0 0 1 = 17P	$\bar{1}$ 0 0 0 $\bar{1}$ = -17P
$\bar{1}$ 0 $\bar{1}$ = -5P	$\bar{1}$ 0 0 $\bar{1}$ = -9P	1 0 0 0 $\bar{1}$ = 15P	$\bar{1}$ 0 0 0 1 = -15P
		1 0 $\bar{1}$ 0 1 = 13P	$\bar{1}$ 0 1 0 $\bar{1}$ = -13P
		1 0 $\bar{1}$ 0 $\bar{1}$ = 11P	$\bar{1}$ 0 1 0 1 = -11P

Notons que les points négatifs sont les symétriques des points positifs, ils ne sont ni stockés ni calculés, ils s'obtiennent gratuitement. Pour des fenêtres de longueur maximale 5 bits, le nombre de points précalculés est de 10.

MOF utilise une représentation signée au même titre que NAF, mais il peut y avoir deux chiffres non-nuls consécutifs. Pour des fenêtres de longueur maximale de 5 bits, la dérivation des points caculés se fait en faisant une soustraction de chaque bit k_{i-1} du bloc avec celui de k_i. Par exemple pour quelques valeurs (16 à 31) des blocs de 5 bits on a :

Les combinaisons restantes donnent les mêmes valeurs négatives. Ainsi, le nombre de points précalculés est de 7 : [3]P, [5]P, [7]P, [9]P, [11]P, [13]P, [15]P.

Le recodage par complémentation utilise le même principe de représentation de MOF, mais pour la dérivation des points précalculés, on prend toutes les combinaisons de 5 bits au maximum, commençant et se terminant par un chiffre non nuls, soient (2^{w-1}-1)=15 points précalculés : [3]P, [5]P, [7]P, [9]P, [11]P, [13]P, [15]P, [17]P, [19]P, [21]P, [23]P, [25]P, [27]P, [29]P and [31]P.

10000.	10001.	10010.	10011.
.10000	.10001	.10010	.10011
$1\bar{1}000.\mapsto$P	$1\bar{1}001\bar{1}\mapsto$9P	$1\bar{1}01\bar{1}0\mapsto$9P	$1\bar{1}010\bar{1}\mapsto$5P
10100.	10101.	10110.	10111.
.10100	.10101	.10110	.10111
$1\bar{1}1\bar{1}00.\mapsto$5P	$1\bar{1}1\bar{1}1\bar{1}\mapsto$11P	$1\bar{1}10\bar{1}0\mapsto$11P	$1\bar{1}100\bar{1}\mapsto$3P
11000.	11001.	11010.	11011.
.11000	.11001	.11010	.11011
$10\bar{1}000.\mapsto$3P	$10\bar{1}01\bar{1}\mapsto$13P	$10\bar{1}1\bar{1}0\mapsto$13P	$10\bar{1}10\bar{1}\mapsto$7P
11100.	11101.	11110.	11111.
.11100	.11101	.11110	.11111
$100\bar{1}00.\mapsto$7P	$100\bar{1}1\bar{1}\mapsto$15P	$1000\bar{1}0\mapsto$15P	$10000\bar{1}\mapsto$P

Le Tableau 1.2 présente une comparaison des différentes méthodes.

Méthodes	*Coût*	*Points précalculés*	*w=5*	*Directions*
DA	(l-1) D+$\frac{(l-1)}{2}$A	0	...	$\rightleftarrows$
NAF	(l-1) D+$\frac{(l-1)}{3}$A	0	...	$\rightarrow$
MOF	(l-1) D+$\frac{(l-1)}{2}$A	0	...	$\rightleftarrows$
RC1	< (l-1) D+$\frac{(l-1)}{3}$A	0	...	$\rightleftarrows$
wNAF	(l-1) D+$\frac{l}{w+1}$A	$<2^{w-1}-1$	10	$\rightarrow$
wMOF	(l-1) D+$\frac{l}{w+1}$A	$\leq 2^{w-2}-1$	7	$\rightleftarrows$
wRC1	<(l-1) D+$\frac{l}{w+1}$A	$2^{w-1}-1$	15	$\rightleftarrows$

TABLE 1.2 – Coût d'exécution et de stockage mémoire

1.5/ LES PROTOCOLES DE CONTRÔLE D'ACCÈS AUX RCSFS

Dans cette section, nous allons décrire quelques unes des solutions de contrôle d'accès ainsi que des protocoles proposés pour les réseaux de capteurs sans fil. On distingue trois classes de protocoles : les protocoles d'ajout de nouveaux nœuds au réseau de capteurs, les protocoles d'authentification des requêtes et les protocoles d'authentification des utilisateurs.

1.5.1/ LES PROTOCOLES D'AJOUT DE NOUVEAUX NŒUDS

A cause de leurs ressources limitées et du déploiement dans des endroits souvent hostiles, les nœuds capteurs peuvent être attaqués physiquement ou épuiser leur énergie. Par conséquent, l'ajout d'un nouveau nœud capteur devient une nécessité. Beaucoup de solutions sont basées sur la cryptographie asymétrique utilisant les courbes elliptiques [54, 75]. Une première classe des solutions est statique [26, 29, 45, 34], nous allons voir les protocoles dynamiques (utilisés dans les RCSFs), où, l'information existante dans les nœuds capteurs n'est pas mise à jour après l'ajout d'un nouveau capteur.

Le Nouveau Protocole de Contrôle d'Accès (NPCA) [58] utilise un nouveau mécanisme de contrôle d'accès basé sur les ECCs et les chaînes de hachage. Il produit une procédure d'authen-

tification et un mécanisme de génération de clés simple et efficace pour les capteurs. NPCA est basé sur la méthode [109] qui propose un mécanisme de contrôle d'accès basé sur les ECCs et plus efficace que RSA. [109] permet au nouveau nœud de joindre le réseau dynamiquement et intègre un mécanisme d'établissement de clés permettant au nouveau nœud de partager des clés avec ses futurs voisins afin de protéger les communications entre nœuds capteurs. Il inclut une estampille temporelle pour produire l'authentification. Cependant, ce mécanisme suppose que chaque capteur admet un intervalle de temps avant d'être compromis. Ainsi, il ne sera pas utilisé pour certaines applications pratiques des RCSFs [109, 108].

Un Nouveau Protocole de Contrôle d'Accès Dynamique (NPCAD) [59] basé aussi sur [109] est proposé. Cette solution permet de contrôler l'accès d'un nouveau nœud au réseau en utilisant les fonctions de hachage, qui, de par leur faible demande de ressources sont très adaptées dans les RCSFs . Dans ce protocole, chaque nœud exécute 5 opérations de hachage et 4 opérations Ou exclusif pour accomplir une authentification mutuelle et établir une clé partagée afin de sécuriser les communications. Ce qui le rend très adapté pour les capteurs.

NPCA utilise la même approche que dans [59]. Il ne nécessite aucune estampille temporelle et n'admet pas d'intervalle de temps avant qu'un capteur ne puisse être compromis. Il réduit largement le nombre de communications et de calculs entre les nœuds et peut être implémenté comme mécanisme de contrôle d'accès dynamique pour des applications de RCSFs. Ce protocole effectue deux tâches : *une authentification de nœud* et *un établissement de clés*.

– *Authentification de nœud* : un nœud déployé établit son identité avec celle de ses voisins et a le droit d'accéder au WSN à travers une authentification.
– *Etablissement de clé* : à travers l'authentification, des clés sont créées entre le nœud déployé et ses voisins pour sécuriser les communications. Ceci garantit que deux capteurs peuvent trouver une clé commune entre eux (une clé de paire partagée entre un nœud et ses voisins directs).

NPCA est composé de trois phases : une *phase d'initialisation*, une *phase d'authentification et d'établissement des clés*, et une *phase d'addition de nouveau nœud* .

– *Phase d'initialisation :* supposons qu'on ait N_1, N_2,....,N_r nœuds voisins dans une zone. La station de base (SB) choisit r clés secrètes k_1, k_2,....,k_r pour les r nœuds voisins N_1, N_2,....,N_r du nœud à ajouter et chaque clé k_i est chargée dans le nœud N_i correspondant avec la fonction de hachage à sens unique h(). La station de base calcule pour chaque nœud $h^z(k_i)$=h($h^{z-1}(k_i)$) (avec z un grand nombre constant) puis diffuse l'engagement $h^z(k_i)$ et le nombre z au groupe de nœuds N_1, N_2,....,N_r. Dans ce cas, z représente la taille du réseau et limite le nombre maximum de nœuds. L'expression $h^m(k_i)$ représente une application de m fois de la fonction h() sur k_i. La station de base choisit un grand nombre premier p ($p \approx 2^{160}$), une courbe elliptique $E(\mathbb{F}_p)$, un groupe cyclique $G = <P>$ de point générateur P avec $nP=\infty$, où n est l'ordre du point P.
– *Phase d'authentification et d'établissement de clé :* après l'authentification entre nœuds, une clé de paire est partagée. Pour un nœud N_i, la chaîne de hachage est mise à jour après chaque authentification réussie et est remplacée par $h^{z-m}(k_i)$ après avoir procédé à m authentifications. Supposons que les nœuds N_i et N_j aient effectué respectivement u et v fois authentifications, donc les chaînes de hachage diffusées pour N_i et N_j sont respectivement $h^{z-u}(k_i)$ et $h^{z-v}(k_j)$. Le processus d'authentification pour les nœuds pour N_i et N_j se passe comme suit :

    1. Le nœud N_i génère un nombre aléatoire t_i et calcule le point $A_i = t_i$P=(Ax_i, Ay_i) de la courbe elliptique $E(\mathbb{F}_p)$ et s_i=h(Ax_i|| $h^{z-u-1}(k_i)$) puis diffuse { A_i, t_i, N_i}. De même, N_j génère un nombre aléatoire t_j, calcule le point $A_j=t_j$P=(Ax_j, Ay_j) sur la courbe elliptique $\mathbb{F}_p$ et s_j=h(Ax_j|| $h^{z-v-1}(k_j)$), puis diffuse $\{A_j, s_j, N_j\}$. Pour plus de sécurité, les nombres t_i et t_j ne sont pas réutilisés.

    2. Après avoir reçu $\{A_j, s_j, N_j\}$, le nœud N_i calcule une clé de session partagée $k_{ij}=t_iA_j=(kx_{ij},ky_{ij})$ et z_i=h(kx_{ij} || $h^{z-u-1}(k_i)$),

  - il envoie $\{z_i, h^{z-u-1}(k_i)\}$ au nœud N_j.
  - N_j vérifie $\mathrm{h}(h^{z-u-1}(k_i)) = h^{z-u}(k_i)$.
  - Si c'est valide N_j calcule $k_{ij} = t_j A_i = (kx_{ij}, ky_{ij})$ puis vérifie si $\mathrm{h}(Ax_i \| h^{z-u-1}(k_i)) = s_i$ et $\mathrm{h}(kx_{ij} \| h^{z-u-1}(k_i)) = z_i$.
  - Si c'est valide, N_i est authentifié par N_j.
3. N_j calcule $z_j = \mathrm{h}(kx_{ij} \parallel h^{z-v-1}(k_j))$ et envoie $\{z_j, h^{z-v-1}(k_j)\}$
4. N_i vérifie si $\mathrm{h}(h^{z-v-1}(k_j)) = h^{z-v}(k_j)$, $\mathrm{h}(Ax_j \| h^{z-v-1}(k_j)) = s_j$ et $\mathrm{h}(kx_{ij} \| h^{z-v-1}(k_j)) = z_j$.
   - Si c'est valide, N_j est aussi authentifié par N_i.
5. N_i et N_j mettent à jour leur fonction de hachage respectivement à $h^{z-u-1}(k_i)$ et $h^{z-v-1}(k_j)$ et informent les autres nœuds du groupe via la station de base.

Selon l'algorithme de Diffie-Hellman sur les courbes elliptiques [36], $k_{ij}=t_jA_i=t_iA_j=t_it_j\mathrm{P}$. Par la diffusion de la chaîne de hachage, deux capteurs quelconques N_i et N_j sont en mesure de s'authentifier et d'établir une clé de session partagée pour sécuriser leur communication.

– *Phase d'ajout de nouveau nœud :* quand un nouveau nœud avec une identité N_{r+1} veut joindre le réseau, la station de base génère et déploie la clé k_{r+1} et la chaîne de hachage $h^z(k_{r+1})$ dans celui-ci. La SB informe ensuite les autres nœuds du réseau de $h^z(k_{r+1})$ et z. L'authentification et l'établissement de clé est la même que la phase précédente. Ainsi, l'addition de nœud est possible jusqu'à ce que toutes les valeurs de la chaîne de hachage soient consommées.

La sécurité de ce protocole repose sur des mécanismes peu gourmands en ressources tels que les ECCs et les fonctions de hachage à sens unique. Avec la clé secrète k_i, seul le nœud N_i peut dériver la chaîne de hachage valide $h^{z-u-1}(k_i)$ puisque $h^{z-u}(k_i)=\mathrm{h}(h^{z-u-1}(k_i))$. Même si un attaquant peut obtenir : $A_i=t_i\mathrm{P}= (Ax_i, Ay_j)$, $A_j=t_j\mathrm{P}=(Ax_j, Ay_j)$, $s_i=\mathrm{h}(Ax_i \| h^{z-u-1}(k_i))$, $s_j=\mathrm{h}(Ax_j \| h^{z-v-1}(k_j))$ dans l'étape 1, il est très difficile pour lui de dériver $h^{z-u-1}(k_i)$ et $h^{z-v-1}(k_i)$ de s_i et s_j. Ils sont protégés par la fonction de hachage h(). Sans connaître $h^{z-u-1}(k_i)$ et $h^{z-v-1}(k_j)$ de l'étape 1, l'attaquant ne peut pas facilement usurper l'identité des nœuds N_i ou N_j (mascarade) pour calculer s_i et s_j et influencer les autres nœuds.

Les nombres aléatoires t_i et t_j sont utilisés une seule fois et permettent de résister contre la répétition.

Puisque la chaîne de hachage est mise à jour après chaque authentification, l'attaquant ne peut pas utiliser l'ancienne chaîne de hachage pour se faire passer pour un nœud légal.

Si l'attaquant obtient $A_i=t_i\mathrm{P}$ et $A_j=t_j\mathrm{P}$, il est trés difficile pour lui de dériver les nombres t_i et t_j de A_i et A_j respectivement. Sans connaître t_i et t_j, un attaquant ne peut pas obtenir d'information de la clé $k_{ij}=t_jA_i=t_iA_j=t_it_j\mathrm{P}$.

Cependant, dans son extension [63], il a été démontré que NPCA est vulnérable à l'attaque par répétition et par mascarade dûes à une absence d'authentification de la SB. Il effectue seulement une authentification unilatérale et ne permet pas de renouveler la chaîne de hachage. Pour ce faire, [63] propose une solution qui supporte l'authentification mutuelle et le renouvellement de la fonction de hachage. Par ailleurs, dans [71] d'autres insuffisances de NPCA ont été montrées : la durée de vie du réseau et sa disponibilité. La durée de vie du réseau est limitée par la taille de la fonction de hachage (paramètre z). Un nœud N_i ne peut plus être authentifié par ses voisins après avoir utilisé $h^{z-z+1}(k_i)$, en plus l'authentification par la chaîne de hachage nécessite beaucoup de communications et un surplus de mémoire.

Selon NACP, la SB se charge d'annoncer la chaîne de hachage pour sa mise à jour, ce qui signifie que certains nœuds informent la SB de l'état de leurs voisins à travers une communication multi-sauts. En plus, après la diffusion de la chaîne de hachage à mettre à jour, chaque nœud aura à stocker l'état des autres nœuds car il ne sait pas à priori lequel enverra une requête d'authentifica-

tion. [71] reprend NPCA tout en gardant ses avantages et n'a pas besoin d'informer de l'état de la chaîne de hachage. Cependant, il a été démontré dans [107] que [63] présente des vulnérabilités liées à l'attaque par mascarade du nouveau nœud dans sa *phase d'addition* et aucune solution n'est proposée. Toutes ces méthodes sont comparées en termes de temps de calcul et de nombre de communications nécessaires pour réaliser l'authentification et l'établissement de clés (voir Tableau 1.3).

– T_{EM} : multiplication scalaire sur la courbe elliptique
– T_H :le temps pour exécuter la fonction de hachage.
– Le temps de calcul de l'addition modulaire et de l'opération XOR sont ignorés car ils sont négligeables devant T_{EM}, et T_H.

Protocoles	***Temps pour chaque nœud (authentification et établissement de clé)***	***Nombre de transmission pour établir une clé***
Zhou et al.[109]	$3T_{EM}+T_i+T_H$	21
Huang et al. [58]	$2T_{EM}+ 5T_H$	10
Kim et al.[63]	$2T_{EM}+8T_H$	14
Lee et al. [71]	$2T_{EM} + 5T_H$	8
Huang et al.[59]	$5T_H$	7

TABLE 1.3 – Comparaison des temps de calcul et de transmission

Un autre protocole de contrôle d'accès utilise une approche d'addition de nouveau nœud tout à fait différente de celle de NPCA [97]. Il produit, l' authentification de nœud, l'intégrité et la confidentialité des données. Cette solution utilise Self-Certified Elliptic Curve Diffie-Hellman (ECDH) cryptosystem [6] afin d'établir une clé de paire entre un nouveau capteur à déployer et un nœud contrôleur c. Ainsi, le réseau a besoin d'une autorité de certificat (AC), des nœuds de contrôle qui peuvent être des nœuds réguliers avec plus de ressources et des nœuds réguliers qui ne sont que de simples relais. A travers l'authentification, les nœuds contrôleurs établissent des clés de paire avec le nouveau nœud capteur.

Afin d'éviter des attaques de type Déni de Service (DdS) contre le Self-Certified ECDH, une authentification faible basée sur une fonction polynomiale est utilisée à priori [75]. L'AC génère un polynôme bivarié f de degré t sur un corps fini $GF(p)$ où p est un grand nombre premier et la fonction f satisfait les propriétés suivantes : $f(x,y) = f(y,x)$. Pour un nœud contrôleur c, l'AC calcule $f(c,y)$ en remplaçant x par c dans le polynôme bivarié $f(x, y)$ et le déploie dans le contrôleur c. Similairement, pour un nœud simple i, l'AC calcule $f(i,y)$ en remplaçant x par i dans le polynôme bivarié $f(x, y)$ et le déploie dans le nœud i. Pour une communication entre un nœud i et un contrôleur c, une clé secrète est générée sur la base de leur identité : $f(c, i)=f(i, c)$. Ainsi, la clé $f(c, i)$ est utilisée pour authentifier les messages échangés. Avant qu'un nouveau nœud ne joigne le réseau, l'AC génère des clés privées liées aux paramètres des nœuds du réseau (nœuds contrôleurs et nœuds réguliers). Les nœuds contrôleurs diffusent périodiquement leur identité (IDc). Un nouveau nœud i qui veut joindre le réseau sélectionne le contrôleur avec le plus fort RSSI (strongest signal strength indicator) et lui envoie une requête qui contient la clé publique U_i basée sur son Self- Certified ECDH, un nonce (number once) et son identité (IDi). Le nouveau nœud i appose un code d'authentification de message HMAC en utilisant la clé de paire k_{ic} générée par la méthode polynômiale. Quand le nœud contrôleur reçoit la requête, il évalue le polynôme en utilisant i, dérive la clé k_{ci}, puis vérifie le HMAC. Si le HMAC est non valide, la requête est supprimée, sinon, le contrôleur exécute le protocole Self-Certified ECDH pour établir une clé de paire sk_{ci} avec le nœud i. Puis en retour, il envoie une réponse contenant la clé publique U_c générée par Self-Certified ECDH avec un HMAC

utilisant k_{ci} et un nouveau HMAC utilisant sk_{ci}. Après un délai transitoire, le contrôleur c envoie la nouvelle clé de groupe chiffrée avec la clé sk_{ci} pendant que le nœud i a fini l'authentification avec le protocole Self-Certified ECDH et obtient la clé k_{ic}. Ainsi, le nœud i déchiffre la nouvelle clé de groupe. Pour le contrôle d'accès, tous les nœuds du groupe utilsent la même clé de groupe pour protéger les paquets transmis dans le réseau. Pour tout paquet envoyé, l'expéditeur génère un code d'intégrité de message (Message integrity code, MIC) avec la clé du groupe. Pour tout paquet reçu, le récepteur utilise la clé du groupe pour vérifier le MIC. Si le MIC est valide, le paquet est retransmis, sinon il est supprimé.

Cependant, on peut noter que la sécurité de ce protocole dépend de celle du mécanisme de distribution de clé utilisé. Tous les nœuds éligibles se partagent une clé de réseau qui doit être mise à jour quand un nœud est compromis. Ainsi, des mécanismes sont utilisés pour mettre à jour cette clé [97, 83, 26]. La sécurité de la fonction polynômiale est faible, elle est divulguée si le nombre de nœuds compromis est supérieur au degré du polynôme. Cette fonction produit juste une authentification faible et ne peut pas remplacer le Self-Certified ECDH pour l'établissement de clés.

1.5.2/ Les protocoles d'authentification de requêtes

L'authentification de requêtes signifie l'authentification de l'origine des données. Pour séparer les concepts, nous distinguons deux types de requêtes : les requêtes des utilisateurs que nous appelons *requêtes externes* et celles du système c'est à dire de la station de base ou d'un nœud capteur que nous appelons *requêtes internes*. Beaucoup de capteurs ne reçoivent pas directement la requête de la station de base, mais via un autre capteur. Par conséquent l'authentification devient une nécessité afin de s'assurer que les données envoyées par un capteur correspondent à la requête originale. Dans ce cas, pour éviter l'envoie de requêtes par un attaquant, un mécanisme doit être mis en place dans chaque nœud capteur.

1.5.2.1/ Authentification de requêtes externes

Le RCSF satisfait l'authentification des requêtes à travers les propriétés de sureté et de vivacité définies comme suit :
- *Sureté* : si un nœud capteur s traite une requête q, celle-ci est postée par un utilisateur légitime.
- *Vivacité :* toute requête q postée par un utilisateur légitime, sera traitée par un nœud (ou chaque nœud) d'un groupe de nœuds capteurs susceptibles de la traiter afin de fournir une réponse à l'utilisateur.

Dans la suite, nous allons passer en revue quelques mécanismes existants d'authentification de requêtes d'utilisateur.

a) Authentification robuste de requêtes d'utilisateur [17]

Cette solution permet de résister à la capture d'un certain nombre de nœuds capteurs à partir d'un seuil donné [17]. Cela signifie que si le nombre de capteurs au voisinage de l'utilisateur est n, t ($t<n$) capteurs peuvent être compromis. Ainsi, l'utilisateur pourra communiquer avec n-t capteurs de son voisinage. La solution produit l'authentification d'une requête d'utilisateur si celle-ci implique un seul capteur : exemple la température captée par un nœud capteur s.

Lorsque le nombre d'utilisateurs d'un RCSF est très élevé, la méthode naturelle utilisée pour l'authentification est la cryptographie asymétrique à cause de sa scalabilité. L'idée de base du protocole d'authentification robuste de requêtes est de laisser les capteurs au voisinage de l'utilisateur servir d'interprètes (ou passerelles) entre la cryptographie asymétrique de l'utilisateur et celle symétrique utilisée dans le RCSF durant les communications entre nœuds capteurs. Ainsi, l'utilisateur s'authentifie aux nœuds capteurs de son voisinage en utilisant la cryptographie asymétrique, et ensuite ces derniers communiquent avec le reste du réseau en utilisant la cryptographie symétrique. Cette approche utilise une infrastructure de clé publique PKI (Public Key Infrastructure) avec

différentes stratégies de gestion des certificats. La station de base sert d'autorité de certificat (AC) avec respectivement ses clés publique et privée (*cle_privAC, cle_pubAC*). Le certificat d'un utilisateur (U) légitime est signé par l'AC en utilisant la clé publique i.e. *certU = signAC(cle_pubU).* Chaque nœud capteur est préchargé avec la clé publique de l'AC, ce qui leur permet de vérifier indépendamment les certificats des utilisateurs. Puisque la cryptographie asymétrique nécessite plus de calculs en déchiffrement et génération de signature qu'en chiffrement et vérification de signature[48, 54], pour une authentification unilatérale d'un utilisateur, les nœuds capteurs du voisinage effectuent seulement la vérification de signature. Contrairement aux ECCs, le chiffrement et la signature nécessitent plus de calculs que le déchiffrement et génération de signature. Cependant, il peut être très coûteux pour un capteur de vérifier des signatures si le nombre de requêtes est très élevé. On peut noter que les capteurs ne sont pas authentifiés par l'utilisateur, ainsi un nœud compromis peut se faire passer pour plusieurs capteurs et authentifier l'adversaire. Dans leur travaux futurs, les auteurs envisagent d'utiliser les ECCs pour une solution authentification robuste intégrant une authentification mutuelle et un mécanisme d'établissement de clés de session.

Ce protocole, par l'envoi de faux certificats, peut être vulnérable à l'attaque par brouillage de la couche MAC. Il ne permet pas aussi de traiter les requêtes impliquant plusieurs nœuds capteurs comme par exemple, le calcul de la température dans une région donnée du réseau de capteurs.

b) Authentification de requêtes basée sur la clé symétrique [12]

Les mécanismes de contrôle d'accès basés sur la cryptographie asymétrique dans les RCSFs sont peu nombreux. [17] est la première technique qui produit l'authentification d'une requête d'utilisateur si celle-ci implique un seul capteur. De ce fait cette solution ne permet pas de gérer l'authentification des requêtes impliquant plusieurs capteurs. [12] improvise la technique de pré-distribution de clé de [23] et y ajoute un support additionnel d'authentification de requête. Il traite les requêtes d'un utilisateur impliquant plusieurs capteurs et utilise la cryptographie symétrique. Il admet un seuil de nœuds compromis et utilise un polynôme aléatoire bivarié et symétrique comme secret global. Comme [17], seule la propriété de sureté est traitée, celle de vivacité est laissée à d'autres protocoles (exemple : protocole de routage sécurisé).

Dans le protocole, un utilisateur diffuse son identité ID_u et sa requête q. Le réseau de capteurs identifie un ensemble de capteurs S_q capables de la traiter. Ces capteurs élisent un leader en utilisant la méthode [40]. Ce dernier s'engage de générer un *nonce* (number once) qu'il va envoyer aux autres capteurs de S_q et l'utilisateur sera notifié du *nonce* et de S_q. Pour chaque capteur de S_q, l'utilisateur calcule un code d'authentification de message (Message Authentication Code, MAC), et rassemble tous les MACs puis les envoie à chaque capteur de S_q. Chaque capteur de S_q, après avoir reçu la collection de MACs, calcule un MAC sur le nonce et vérifie s'il y'a une correspondance dans la collection de MACs reçue. Si tel est le cas, la requête est acceptée.

Comparée à [17], cette solution présente plusieurs avantages. Elle est basée entièrement sur la cryptographie symétrique et considère les requêtes impliquant plusieurs capteurs. Puisque le protocole est basé sur la méthode [23], il est sécurisé à *(t-1)* (t est le degré du polynôme utilisé) nombre de nœuds capturés. Cependant, il n'est pas tout à fait résistant à l'attaque déni de service, car un adversaire peut occuper un nœud avec de faux messages même s'il ne peut réussir à s'authentifier. En conséquence, cette technique doit inclure un mécanisme de rejet automatique des requêtes illégitimes.

Le Tableau 1.4 présente une comparaison des protocoles et des mécanismes (fonction de hachage à sens unique et arbre de hachage) d'authentification de requêtes d'utilisateurs selon deux paramètres : la complexité de communication qui dépend de la densité de nœuds et la complexité du stockage mémoire.
– u représente le nombre de chaînes de clés utilisées ;

– n le nombre de capteurs pouvant authentifier avec succès une requête légitime ;
– t le degré du polynôme utilisé pour négocier la clé ;
– N le nombre de capteurs dans le RCSF.

Solutions	*Complexité communication*	*Complexité stockage*
Une seule chaîne de clés	O(1)	2
Arbre de hachage	O(1)	2+2.u
Authentification robuste [17]	O(n)	2
Authentification (clé symétrique) [12]	O($\sqrt{N}$)	O($t$log(N))

TABLE 1.4 – Comparaison des solutions d'authentification de requêtes d'utilisateurs

1.5.2.2/ Authentification de requêtes internes

Pour certaines applications des RCSFs, les données doivent être protégées contre tout accès non autorisé en particulier quand la station de base est la seule entité autorisée à envoyer des requêtes. Les propriétés de sureté et de vivacité sont définies comme suit :
– *Sureté* : si un nœud capteur traite une requête q, celle-ci provient de la station de base ou d'un nœud capteur du réseau.
– *Vivacité :* toute requête légitime q sera reçue par tous les capteurs du réseau concernés.
Une première solution propose d'utiliser les signatures numériques pour l'authentification des requêtes propagées dans un réseau de capteurs où chaque capteur sera préchargé avec la clé publique d'une autorité de certificat [93]. Cependant, la cryptographie asymétrique est coûteuse pour les ressources limitées des nœuds capteurs.

Des approches d'authentification de requêtes internes proposent une solution d'authentification des requêtes diffusées par la station de base dans le réseaux de capteurs [15]. Ce protocole, montre comment la station de base authentifie ses requêtes auprès des nœuds pour que seules les requêtes légitimes soient répondues par les capteurs tandis que la propagation de fausses requêtes est limitée dans une partie logarithmique du réseau. Par exemple dans un réseau de capteurs où le modèle de livraison est basé sur des requêtes commandées par la station de base, les capteurs doivent vérifier avec une certaine probabilité si les requêtes sont envoyées par la station de base ou non. Dans ce cas, deux solutions sont possibles :
– Si une requête est légitime, c'est-à-dire venant de la station de base, les capteurs doivent la propager rapidement.
– Si elle est ilégitime, c'est-à-dire non envoyée par la station de base, elle doit être freinée le plus rapidement possible pour ne pas se propager dans le réseau, au pire des cas, la propagation se fait dans une partie logarithmique du réseau.
Ce protocole utilise la cryptographie symétrique et est basé sur la méthode décrite dans [25], mais il est plus performant et est lié à une coopération implicite entre les nœuds quand une requête est diffusée dans le réseau. Il utilise des identités (IDs) basées sur des mécanismes de prédistribution de clés décrits dans [45, 78]. Le protocole utilise la *stratégie de passe*, c'est-à-dire si un capteur n'est pas en mesure de déterminer si une requête est légitime ou non, il la passe à ses voisins. L'authentification des requêtes est probabiliste, le protocole utilise l'idée dans [25] avec un MAC de 1-bit après application d'une fonction de hachage $h()$ sur une requête q. Chaque capteur est préchargé avec un groupe de clés choisies aléatoirement dans un ensemble de clés, et pour chaque requête q, un nombre de MACs de 1-bit est calculé en utilisant les clés du groupe. Quand une requête est reçue, le capteur peut, avec une certaine probabilité, avoir une de ses clés utilisée pour calculer les MACs et pourra ensuite vérifier l'authenticité de la requête. Plus le nombre de MACs augmente, plus le probabilité de détecter une fausse requête augmente. Ainsi, une requête

interne sera propagée sans obstacle, cependant la propagation de celle venant d'un adversaire sera trés limitée et sera finalement supprimée. La fonction de hachage h() étant irréversible, il sera impossible pour un adversaire de forger des requêtes en choisissant un nombre quelconque x et de trouver la requête q associée telle que $h(q)=x$.

D'autres approches comme SPINS (Security Protocol for Sensors Network) [86] à travers μTESLA réalisent l'authentification des requêtes diffusées en utilisant les fonctions de hachage, une clé symétrique partagée par chaque nœud avec la station de base et un temps de synchronisation. Ce protocole réalise l'asymétrie par une révélation retardée des clés symétriques et utilise un MAC pour authentifier les messages diffusés. Sa sécurité dépend de celle du mécanisme de gestion du temps de synchronisation qui est un problème difficile dans les réseaux de capteurs [47].

1.5.3/ Les protocoles d'authentification d'utilisateurs

Dans les réseaux de capteurs sans fil, les solutions d'authentification d'utilisateurs sont moins courantes. Ces solutions utilisent la technique d'authentification par mot de passe qu'on peut trouver un peu partout dans differents exemples et applications des réseaux même traditionnels (connexion à un réseau local sans fil, transaction par carte bancaire etc.). Certaines de ces solutions utilisent la technique d'authentification par mot de passe faible [60]. Elles présentent l'avantage d'une mémorisation facile du mot de passe. Cependant, la cryptographie à clé publique utilisée reste le principal inconvénient pour une application dans l'environnement des capteurs. Par contre, d'autres solutions centrées sur l'environnement des cartes à puce utilisent la technique d'authentification par mot de passe fort et sont basées uniquement sur les fonctions de hachage et l'opérateur OU exclusif, ce qui facilite leur implémentation dans les RCSFs [9, 69, 94, 96]. Leur inconvenient réside dans la difficulté de mémorisation du mot de passe. Une premiére adaptation de ces dernières plus récentes, basées sur la technique d'authentification par mots de passe fort a été proposée pour les RCSFs [104]. Cette adaptation basée sur la méthode [70] permet aux utilisateurs légitimes d'accéder au réseau de capteurs selon le principe de fonctionnement suivant. Avant d'envoyer des requêtes au réseau de capteurs, un utilisateur doit s'enregistrer avec un nom d'utilisateur et un mot de passe auprès d'un nœud passerelle ou Gateway (GW), qui peut être la station de base du réseau. Ensuite, l'utilisateur doit se loguer auprès d'un des nœuds (login Nodes-LN) doté de plus de ressources et chargé d'authentifier les utilisateurs déjà enregistrés auprès du GW. Et enfin, l'utilisateur est authentifié par la GW via les LNs du réseau. Une fois que cela est effectué avec succès, l'utilisateur peut ainsi accéder aux services sollicités du réseau de capteurs. Le protocole est composé de trois phases : une *phase d'enregistrement*, une *phase de login*, et une *phase d'authentification*.

– *Phase d'enregistrement* : l'utilisateur envoie son identité ID_u et son mot de passe PW à la passerelle du réseau GW. Le GW calcule des valeurs A et B puis enregistre ID_u et PW. Il distribue ID_u, A, et une estampille T_s, aux nœuds login (LNs) qui sont en mesure de produire une interface login aux utilisateurs.
– *Phase login* : l'utilisateur envoie sont ID_u et son PW au LN. Si l'ID_u est valide, le LN récupère A et calcule non seulement B, mais aussi C_2 et C_1 puis envoie l'identité ID_u, C_2, C_1, l'estampille temporelle T_s à la GW pour une authentification finale.
– *Phase d'authentification* : le GW vérifie la validité de l'utilisateur et de l'estampille, récupère A et B, puis calcule C_1 et C_2 et les vérifie avant de valider l'authentification.

Cependant, des travaux plus récents ont montré beaucoup d'insuffisances et de vulnérabilités dans ce protocole. Nous allons les voir un peu plus en détail dans le chapitre 2.

1.5.4/ Comparaison des différentes solutions

Dans cette partie, il sera très difficile de trouver des paramètres de comparaison car les protocoles n'utilisent pas les mêmes approches et/ou mécanismes de sécurité. Pour cela, nous avons sélectionné les paramètres qui sont souvent pris en considération dans différents protocoles. Parmi ces paramètres on a :
La complexité de communication : liée à la densité du réseau.
– T_H : le temps nécessaire pour calculer une fonction de hachage à sens unique,
– T_{XOR} : le temps pour effectuer une opération XOR,
– T_{EXP} : le temps pour calculer une exponentiation modulaire,
– T_{EM} : temps pour exécuter une multiplication scalaire sur la courbe elliptique,
– C_{MH} : le délai de communication multi-sauts entre le LN et la GW.
– n : représente le nombre de nœuds capteurs au voisinage de l'utilisateur
– N : la taille du réseau
– *O(1)* : signifie la communication du protocole ne dépend pas de la densité des nœuds.
– *La topologie :* dans certains protocoles comme ceux d'authentification d'utilisateurs, on a deux types de nœuds : les nœuds capteurs simples, et les nœuds capteurs contôleurs ou login avec plus de puissance et destinés à effectuer plus d'opérations. Ces protocoles peuvent être utilisés ou adaptés à la topologie hiérarchique où les contôleurs ou LNs vont jouer le rôle de chef de cluster . Par contre dans d'autres protocoles, tous les nœuds capteurs sont identiques et ont les mêmes fonctions. Ce qui permettra leur utilisation ou adaptation à la topologie plate.
– *La scalabilité :* qui montre le passage à l'échelle du protocole.
– *Le type de clé :* qui peut être clé symétrique(Sym) ou clé asymétrique (Asym).
A ces paramètres on peut ajouter la résistance contre les attaques les plus fréquentes sur les protocoles de contrôle d'accès telles que :
– RD(Résistance au DdS) : résistance à l'attaque par déni de service.
– RR(Résistance à la Répétition) : résistance à l'attaque par répétition.

Solutions	*Clés*	*Scalabilité*	*Complexité communication*	*Topologies*	*RD*	*RR*
Wong et al [104]	Sym	Oui	$7T_H$+$4T_{XOR}$+ $3C_{MH}$= $O(1)$	hierarchique	Oui	Non
Vaidya et al.'s scheme [101]	Sym	Yes	$11T_H$+$4T_{XOR}$+ $3C_{MH}$ =$O(1)$	hierarchique	Oui	Oui
Authentification robust [17]	Asym	Oui	$2nT_H$+$3T_{EXP}$ =$O(n)$	plate	Non	Oui
Authentification (clé symétrique) [12]	Sym	Non	$O(\sqrt{N})$	plate	Non	Oui
Chaîne de clés [106]	Sym	Non	$O(1)$	plate	Oui	Oui
Chaînes de clés[106]	Sym	Oui	$O(1)$	plate	Oui	Oui
Arbre de hacahge[106]	Sym	Non	$O(1)$	plate	Oui	Oui
Arbres de hacahge [106]	Sym	Oui	$O(1)$	plate	Oui	Oui
NPCAD [59]	Asym	Oui	$2T_{EM}$+$5T_H$= $O(1)$	plate	Oui	Non

TABLE 1.5 – Comparaison de différentes solutions

1.6/ CONCLUSION

Beaucoup de protocoles de contrôle d'accès sont basés sur la cryptographie à clé publique [54, 17, 53]. Leur usage dans les RCSFs pose problème à cause des ressources limitées des capteurs. Même si des solutions utilisent l'approche asymétrique basée sur les courbes elliptiques beaucoup moins coûteuses en termes de calcul et de stockage mémoire, la cryptographie symétrique reste la meilleure solution pour les réseaux de capteurs. Pour qu'un nœud puisse intégrer le module de contrôle d'accès du réseau quand il le rejoint, un secret (clé, identité etc.) doit être déployé à priori. Ce qui impose souvent des mécanismes de prédistribution de clés au protocole d'addition de nouveaux nœuds lors de leur phase d'initialisation. Les lieux de déploiement souvent hostiles et peu sûrs, font que les solutions de contrôle d'accès doivent utiliser des approches distribuées, avec l'addition d'un nouveau nœud. L'authentification d'une requête ou d'un utilisateur doit être effectuée par plusieurs capteurs afin d'éviter le problème des nœuds compromis.

On peut constater que tous ces mécanismes de contrôle d'accès étudiés dans ce document s'appuient sur un mécanisme de gestion de clés (non décrit) et dont la phase de pré-distribution comporte un secret à priori partagé. Par conséquent, le mécanisme de contrôle d'accès hérite de toutes les vulnérabilités du mécanisme de gestion de clés sur lequel il est basé. On peut envisager, des mécanismes de contrôle d'accès intégrant la gestion de clés dans un seul et unique protocole.

II

CONTRIBUTIONS

2

Authentification et vérification de protocoles à base de hachage

Sommaire

2.1/ Le hachage : généralités et contraintes cryptographiques

Une fonction de hachage H() applique une série de transformations sur des données en entrée et produit une sortie H(x), plus petite, appelée selon le contexte *somme de contrôle*, *empreinte*, *hash*, *résumé de message*, ou *condensé*.

$$\begin{cases} H : \{0,1\}^* \longrightarrow \{0,1\}^n \\ x :\longrightarrow H(x) \end{cases} \tag{2.1}$$

La sortie de la fonction de hachage, en général un nombre, ou une chaîne de caractères de taille limitée ou fixe, peut généralement varier entre 128 et 512 bits selon les algorithmes de hachage. Le but principal du hachage est de permettre une optimisation quand la taille des données de départ nuit aux performances, et de traiter certains services de sécurité. On distingue deux types de fonctions de hachage : les fonctions de hachage cryptographiques et les autres fonctions de hachage.

Les fonctions de hachage non cryptographiques sont moins contraignantes. Leur objectif principal reste la minimisation du stockage mémoire par la construction de structures de données, comme les tables de hachage. On sait que plus la taille des empreintes est petite (opérateur modulo permet

de réduire la taille des empreintes), plus la probabilité des collisions augmente. Il y a collision quand deux ou plusieurs entrées distinctes ont une empreinte identique avec la même fonction de hachage. Il faut donc faire le compromis entre la taille des empreintes et le nombre de collisions.

En cryptographie, les fonctions de hachage sont surtout utilisées pour optimiser et renforcer les performances des algorithmes de chiffrement et ainsi assurer des services de sécurité tels que le contrôle d'accès (authentification et autorisation) et l'intégrité à travers des relations d'égalité, d'égalité probable, non-égalité etc. Ainsi, on peut noter :

- La réduction de la taille des données à chiffrer (par exemple en signature numérique) permettant d'accélèrer les calculs.
- La génération de nombres pseudo-aléatoires utilisés dans diverses applications cryptographiques. Les fonction de hachage aident aussi à la génération de clés pour le chiffrement et à la distribution de clés en cryptographie symétrique.
- Le contrôle d'accès, par le stockage des empreintes des mots de passe afin de les garder secrets. Lors de l'identification d'un utilisateur, le systéme compare l'empreinte du mot de passe d'origine (stockée) avec celle du mot de passe demandé. Toutefois, cette manière de faire n'est pas complètement satisfaisante. Si deux utilisateurs décident d'utiliser le même mot de passe, alors les condensés obtenus seront identiques. Le *salage*, procédure qui consiste à concaténer une chaine de caractères souvent pseudo-aléatoire (nommée le sel) à l'entrée avant le hachage, permet de remédier à cela.
- L'authentification et l'intégrité, par exemple lors des téléchargements de fichiers. En effet, on rencontre des empreintes calculées permettant notamment de vérifier la validité et l'intégrité des archives récupérées. En cryptographie asymétrique, les fonctions de hachage sont largement utilisées dans les signatures électroniques pour l'authentification des objets.

Par ailleurs, en cryptographie, la priorité est de sécuriser l'*empreinte* générée par le hachage de toute attaque. Le temps de calcul et la taille de l'empreinte (généralement plus longue que celle des données) passe au second plan. Ce qui nécessite de faire le compromis entre optimisation des ressources et robustesse en termes de sécurité. Ainsi, une fonction de hachage cryptographique est plus exigeante et doit satisfaire un certain nombre de contraintes :

- Il doit être impossible en pratique de trouver une donnée d'entrée à partir de son *empreinte* : c'est la *résistance aux préimages*, elle est réalisée grâce au caractère d'irréversibilité, de sens unique de la fonction de hachage.
- Etant donnée l'empreinte d'une entrée x, il doit être difficile, voir même impossible, de trouver une autre entrée différente de x et qui donne la même empreinte : c'est la *résistance aux secondes préimages.*
- Il doit être difficile, de trouver deux ou plusieurs entrées aléatoires générant la même empreinte : c'est la *résistance aux collisions*. Pour traiter les collisions, la fonction de hachage doit être parfaite, autrement dit, elle doit être mathématiquement une injection de l'ensemble des entrées (ensemble de départ) dans celui des sorties (ensemble d'arrivée). Ceci peut se faire à travers une distribution uniforme des empreintes dans l'espace disposé. De façon probable, une fonction de hachage *H()* uniforme obéit à la relation suivante :

$$P(H(x) = y) = \frac{1}{card(E)} \tag{2.2}$$

P étant la probabilité de l'empreinte, y un élément de l'ensemble d'arrivée E.

D'autres propriétés sont attendues des fonctions de hachage cryptographiques :

- Comportement imprévisible : un infime changement de l'entrée comme par exemple l'inversion d'un bit entraine une perturbation importante de l'empreinte : c'est l'*effet d'avalanche*. Cet effet contribue au caractère d'irréversibilité de la fonction de hachage.
- Rapide à calculer : cela permettra de gagner du temps surtout quand on doit chiffrer l'empreinte.

Pour une optimisation, normalement le temps de hachage et de chiffrement de l'empreinte doit être inférieur au temps de chiffrement de l'entrée.
– Elle doit être publique pour plus de flexibilité dans son utilisation : même si elle appartient à la famille des algorithmes cryptographiques.

Les trois types d'attaque que l'on trouve généralement sur les fonctions de hachage sont :
– rechercher une entrée possible à partir de l'empreinte. Il est possible de renforcer la sécurité de la fonction de hachage contre cette attaque en utilisant le *salage* qui consiste à concaténer une chaine de caractères souvent pseudo-aléatoire le sel, à l'entrée avant le hachage.
– forger des données pour obtenir la même empreinte qu'un autre message, comme dans le cas d'une attaque par dictionnaire, qui consiste à tester une série d'entrées potentielles, les unes à la suite des autres en espérant obtenir la même empreinte stockée. Ou encore une attaque par force brute qui consiste à tester de manière exhaustive toutes les entrées possibles.
– rechercher des collisions quelconques.

Aujourd'hui, le standard SHA-1 tout comme les fonctions de hachage de la famille des MD sont cassés, ce qui nécessite de faire appel à d'autres fonctions de hachage. Keccak [21] a été choisi en 2008 par l'Institut National des Standards et de la Technologie américain (NIST), comme le nouveau standard des fonctions de hachage.

2.2/ Authentification par hachage pour sécuriser les réseaux de capteurs sans fil

Le mécanisme de sécurité à utiliser par un réseau de capteurs doit tenir compte de l'environnement et des contraintes de celui-ci. Il doit permettre une optimisation des ressources, par la transmission, le stockage, les calculs. Il doit supporter le passage à l'échelle pour un grand nombre de capteurs. Le type d'application, l'architecture du réseau et le modèle de l'attaquant associés aux vulnérabilités du système peuvent aussi influencer sur le choix du mécanisme à utiliser.

Le réseau de capteurs sans fil est constitué généralement d'un grand nombre de capteurs, et très souvent destinés au contrôle d'espaces géographiquement limités. En général, les données récoltées par les capteurs comme par exemple la température ne sont pas confidentielles. Dans certaines applications, la plupart des requêtes sont envoyées à une station de base ou à la passerelle du réseau. Cependant, pour les applications temps réel ou critiques, comme par exemple les applications militaires, les données critiques, doivent être protégées contre toute utilisation frauduleuse, et accessibles en temps réel non seulement depuis la station de base ou la passerelle du réseau, mais parfois aussi de n'importe où dans le réseau à travers les capteurs en mode ad hoc. Dans ce contexte, il est essentiel de limiter l'accès au réseau seulement aux entités éligibles (capteurs ou utilisateurs), tandis que les requêtes provenant des entités non autorisées ne doivent ni être traitées ni transmises par les capteurs du réseau [46]. Le contrôle d'accès a toujours été un problème classique dans beaucoup d'applications et de systèmes informatiques existants. L'authentification à distance des utilisateurs a été depuis longtemps la solution de base la plus utilisée dans les réseaux traditionnels. Cependant, dans les réseaux de capteurs, elle reste moins utilisée à cause des contraintes énergétiques,@ de mémorisation, de calcul et de transmission. Initialement, les solutions d'authentification d'utilisateurs [9, 60, 69, 94, 96] proposées dans l'environnement des cartes à puce étaient inspirées de Lamport(1981) [67], à la différence qu'aucune table de vérification n'était stockée dans les systèmes distants pour vérifier la validité du login de l'utilisateur. Ces solutions utilisent une approche par mot de passe avec un login statique. Certaines d'entre elles [60] utilisent la technique d'un mot de passe faible. Elles présentent l'avantage d'une mémorisation facile du mot de passe. Cependant, de par sa forte demande en ressources, la cryptographie à clé publique basée sur le cryptosystème El Gamal [10] qui utilise une signature basée sur le logarithme discret reste le principal inconvénient pour une application dans l'environnement des

capteurs. Par contre, d'autres solutions [9, 69, 94, 96] utilisent la technique d'un mot de passe fort et sont basées uniquement sur les fonctions de hachage et l'opérateur OU exclusif qui nécessitent moins de calculs, ce qui facilite leur implémentation dans les RCSFs. Leur inconvénient réside dans la difficulté de mémorisation du mot de passe. Des solutions comme[35] introduisent la technique du mot de passe fort avec un login dynamique afin de se protéger contre l'usurpation de login, ce qui permet le libre changement de login et de mot de passe. Cependant, Lee et al.[70] ont montré que ces protocoles basés sur un login dynamique sont toujours vulnérables aux attaques telles que la répétition, la contrefaçon de login et la fabrication. Leur solution proposée pour les cartes à puce utilise la même technique des fonctions de hachage et du OU exclusif. Pour une adaptation dans les RCSFs. Wong et al. [104] proposent une solution basée sur Lee et al.[70] mais moins coûteuse en calcul. La solution de Tseng et al.[99] montre les insuffisances de Wong et al.[104] et essaie d'améliorer sa sécurité. Récemment, Vaidya et al., dans [101], améliorent leur version antérieure [11] basée sur Wong et al. [104]. Leur solution est sécurisée pour contrer plusieurs attaques comme l'attaque sur le login, par répétition et par fabrication, mais reste toujours vulnérable à d'autre types d'attaques que nous détaillerons plus loin.

2.3/ Contribution 1 : Protocoles d'authenfication

Dans ce chapitre, nous présentons un protocole d'authentification robuste par mot de passe, une extension d'une famille de protocoles, à savoir les méthodes de Wong et al, Tseng et al, Vaidya et al.. Contrairement à l'authentification par mot de passe faible, moins coûteuse en stockage mémoire, mais plus coûteuse en calcul (cryptographie asymétrique), ces solutions utilisent l'authentification par mot de passe fort plus coûteuse en stockage mémoire mais moins coûteuse en calcul (les fonctions de hachage). L'objectif de la solution proposée est de se prémunir contre des attaques de types Déni de Service (DdD) en améliorant la sécurité et les performances des protocoles existants. Cette présente contribution s'applique aux domaines des réseaux de capteurs sans fil permettant à des utilisateurs de s'authentifier avant d'accéder aux différents services. La solution proposée utilise les mêmes mécanismes des fonctions de hachage, et bénéficie en plus d'une validation via une implémentation et une vérification de quelques propriétés de sécurité.

Nous présentons en premier lieu quelques hypothèses, et le modèle du réseau sur lequel repose notre contribution. Ensuite, nous passons en revue le protocole de Vaidya et al.[101] avec quelques vulnérabilités. Enfin, nous présentons notre solution avec une analyse de sa sécurité.

2.3.1/ Hypothèses et Modèle du réseau

Un protocole de sécurité peut être efficace de par le mécanisme de sécurité utilisé, mais l'architecture du réseau et le scénario de communication peuvent nous renseigner sur la sécurité du protocole. Le réseau sur lequel va fonctionner notre solution est composé de trois types de nœuds :

- le nœud passerelle ou gateway (GW) : qui assure l'enregistrement et l'authentification finale des utilisateurs, ce nœud est physiquement déployé dans un endroit sûr.
- les nœuds login (LNs) : ils effectuent une partie de l'authentification et se chargent de l'authentification finale des utilisateurs à travers la gateway. Ces nœuds login sont physiquement sécurisés et déployés dans divers endroits du réseau afin de faciliter l'accès des utilisateurs au réseau.
- le nœud simple : qui sert uniquement de relai entre les LNs et la GW.

Les solutions existantes comme celle que nous proposons sont composées de quatre phases : une *phase d'enregistrement*, une *phase de login*, une *phase d'authentification* et une *phase de changement de mot de passe*. Ces phases seront décrites plus en détail dans les prochaines sections.

Durant la phase d'enregistrement, l'utilisateur (UD) est au voisinage de la passerelle GW, cette phase s'effectue une seule fois dans un endroit sécurisé. Durant la phase de login, l'utilisateur doit être au voisinage d'un nœud login. Les Figures 2.1 et 2.2 présentent respectivement l'architecture physique du réseau en *phase d'enregistrement* et en *phase de login-authentification*.

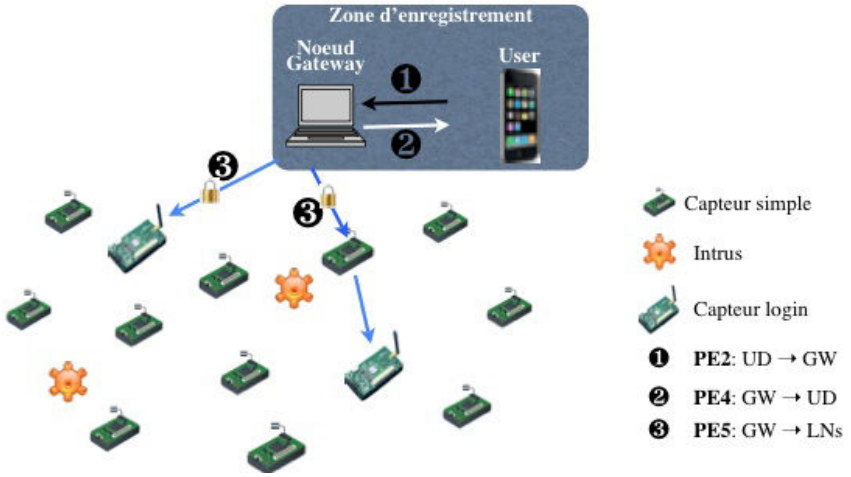


FIGURE 2.1 – Phase d'Enregistrement

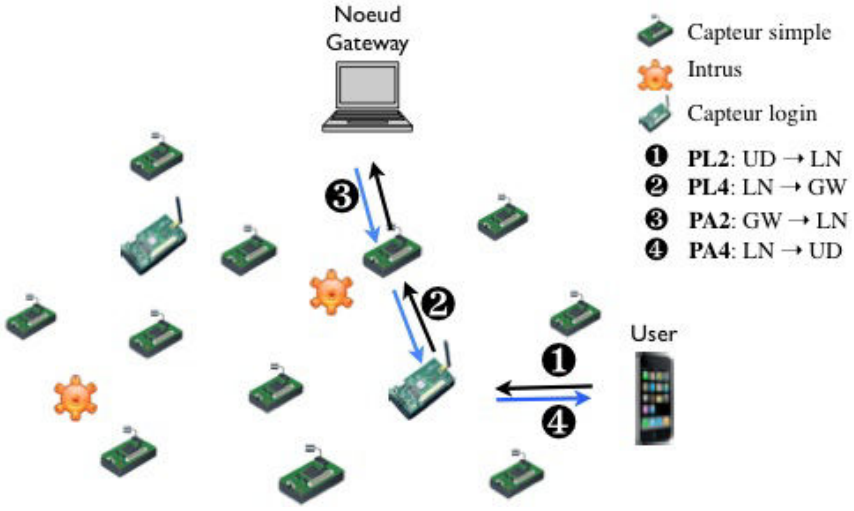


FIGURE 2.2 – Phase Login-Authentification

2.3.2/ DESCRIPTION DU PROTOCOLE D'AUTHENTIFICATION DE VAIDYA ET AL.

La solution de Vaidya et al. [101] est composée de quatre phases : une phase d'enregistrement, une phase de login, une phase d'authentification et une phase de changement de mot de passe. Nous utiliserons pour la suite les notations du Tableau 2.1.

Symboles	*Description*
UD	dispositif de l'utilisateur (PDA, PC etc..)
GW	le nœud passerelle
LN	un nœud capteur Login
H( )	une fonction de hachage à sens unique
$\oplus$	l'opérateur OU exclusif (XOR)
$\|\|$	l'opérateur de concatenation
Succ_Reg	message d'enregistrement avec succès
Acc_login	message d'accepation du login
Succ_Change	message de changement avec succès
x	la clé de la passerelle
UID	l'identité de l'utilisateur
PW	mot de passe choisi par l'utilisateur
TS	temps d'enregistrement d'un nœud
t, T , T_0	temps actuels enregistrés par un des nœuds
$\triangle$T	délai de transmission
$\longrightarrow$	transmission de message

TABLE 2.1 – Notations

1. *Phase d'enregistrement :* durant cette phase, l'utilisateur (UD) est au voisinage de la passerelle GW. Il choisit librement un mot de passe PW et calcule le haché vpw = H(PW). Puis, au temps TS, il envoie son identité UID et le hachage vpw à la passerelle GW en mode sécurisé. Après réception du message, la passerelle GW calcule X = H(UID||x) puis répond à l'utilisateur le message Succ_Reg (X) avec X pour lui notifier que l'enregistrement est effectué avec succès après avoir stocké (UID, vpw, X, TS). L'utilisateur stocke X pour une utilisation future. La GW distribue ensuite les paramètres (UID, X, TS) aux nœuds capteurs de login (LNs) capables de fournir des interfaces aux utilisateurs pour se loguer.

Algorithm 22 *: Etapes de la Phase d'Enregistrement (PE)*

PE1 - - UD : Calcule vpw = H(PW) ;

PE2 - - UD $\longrightarrow$ GW : UID, vpw ;

PE3 - - GW : Calcule X = H(UID || x) ;

- Stocke UID, vpw, X, TS ;

PE4 - - GW $\longrightarrow$ UD : Succ_Reg(X) ;

PE5 - - UD : Stocke X ;

PE5 - - GW $\longrightarrow$LNs : UID, X, TS ;

PE6 - - LN : Stocke UID, X, TS ;

2. *Phase de login :* durant la phase de login, l'utilisateur est au voisinage du LN, il calcule A = H(vpw||t), puis soumet (UID, A, t) au LN le plus proche. Après réception de la requête à T_0, le LN vérifie si le UID soumis est non valide et T_0 - t$\geq\triangle$T. Si au moins une condition est vérifiée, le message de login est rejeté, sinon le LN récupère le paramètre A correspondant et calcule C_K = (X $\oplus$A $\oplus$ T_0), puis il envoie (UID, C_K, T_0, t) à la passerelle GW.

Algorithm 23 *: Etapes de la Phase de Login (PL)*

PL1 - - UD : Calcule A = H(vpw||t) ;

PL2 - - UD ⟶ LN : UID, A, t ;

PL3 - - LN : **Si** UID est valide

- **Alors** le X correspondant est connu ;
- **Sinon** rejette le message de login ;
- **Si** T_0 - t≥△T
- **Alors** rejette le message de login ;
- **Sinon** récupère A, calcule C_K = (X ⊕A ⊕ T_0) ;

PL4 - - LN ⟶ GW :UID,C_K,T_0,t ;

3. *Phase d' authentification :* durant cette phase, la passerelle GW vérifie la validité de UID et t, la requête de login est rejetée s'ils ne sont pas valides. S'ils sont valides, alors la passerelle GW vérifie si T_1-T_0 ≥ △T et T_0-t ≥ △T. Si au moins une des conditions est satisfaite, la requête de login est considérée comme un message répété et est rejetée. Sinon la passerelle récupère le vpw et le paramètre A correspondants, puis calcule A'= H(vpw|| t) et C_K' = (X ⊕ A'⊕ T_0). Si C_K≠ C_K', le message de login est rejeté, sinon, la GW calcule V_M = H(X||A'|| T_1) et envoie un message d'acceptation (Acc_login, V_M, T_1) au LN. Le LN calcule V'_M, et si V_M =V'_M , il calcule également Y_K = H(V'_M|| T_2) et envoie (Acc_login, Y_K, T_1, T_2) à l'utilisteur UD. Après réception du message au temps T_3, l'utilisateur UD vérifie si T_1-T_0 ≥ △T et T_0-t ≥ △T. Si au moins une des conditions est vraie, alors le message Acc_login est rejeté, sinon, l'utilisateur calcule V"$_M$ =H(X||A||T_1) et Y'_K = H(V"$_M$||T_2), puis vérifie si Y_K =Y'_K. Si cette condition est vraie, l'utilisateur UD commence à obtenir les données, sinon le message d'acceptation de login Acc_login est rejeté.

4. *Phase de changement de mot de passe :* durant la phase de changement de mot de passe, l'utilisateur UD change son mot de passe PW en PW1. Ainsi, il calcule vpw1 = H(PW1) et envoie le triplet (UID, vpw, vpw1) à la passerelle GW en mode sécurisé. La passerelle GW vérifie son UID et vpw, alors met à jour sa base de données. La passerelle envoie ensuite un message de changement effectif avec succès (Succ_Change) à UD en même temps elle distribue les informations de mise à jour à tous les LNs. Après réception, les LNs vérifient la validité du UID avant de mettre à jour leurs données.

Algorithm 25 *: Etapes de la Phase de changement de mot de Passe (PP)*

PP1 - - UD : Calcule vpw1 = H(PW1) ;

PP2 - - UD ⟶ GW : UID, vpw, vpw1 ;

PP3 - - GW : **Si** UID et vpw existe sur sa liste

Alors mets à jour vpw avec vpw1, TS avec TS1 ;

PP4 - - GW ⟶ UD : envoie Succ_Change ;

PP5 - - GW ⟶LNs : envoie UID, TS1 ;

- LN : **Si** UID existe dans sa liste ;
- **Alors** mettre à jour TS avec TS1 ;

Algorithm 24 *: Etapes de la Phase d'Authentification (PA)*

PA1 - - GW : **Si** UID et t sont valides

- **Alors** récupère les paramètres (UID, X, TS) ;
- **Sinon** rejette le message de login ;
- **Si** T_1 - $T_0 \geq \triangle$T et T_0 - t$\geq \triangle$T
- **Alors** message de login supposé répété et rejeté ;
- **Sinon** calcule A' = H(vpw|| t) ;
- calcule C'_K = (X$\oplus$A'$\oplus$ T_0) ;
- **Si** $C'_K \neq C_K$
- **Alors** rejette le message de login ;
- **Sinon** calcule V_M = H(X||A'||T_1) ;
- Stocke t ;

PA2 - - GW $\longrightarrow$ LN : Acc_login,V_M,T_1 ;

PA3 - - LN : **Si** T_2 - $T_1 \geq \triangle$T

- **Alors** rejette le message de login ;
- **Sinon** Calcule V'_M = H(X||A||T_1) ;
- **Si** $V_M \neq V'_M$
- **Alors** LN rejette le message de login ;
- **Sinon** Calcule Y_K=H(V'_M||T_2) ;

PA4 - - LN $\longrightarrow$UD : Acc_login,Y_K,T_1,T_2 ;

PA5 - - UD : **Si** T_1 - $T_0 \geq \triangle$T et T_0 - t$\geq \triangle$T

- **Alors** rejette le message Acc_login ;
- **Sinon** calcule $V"_M$ = H(X||A||T_1) ;
- calcule Y'_K = H($V"_M$||T_2) ;
- **Si** $Y_K \neq Y'_K$
- **Alors** rejette le message Acc_login ;
- **Sinon** commence à obtenir les données ;

2.3.3/ Vulnérabilités du protocole d'authentification

Dans cette section, nous nous intéressons à la sécurité du protocole de Vaidya et al. [101]. Ainsi, nous allons montrer que ce protocole est vulnérable à l'attaque de DdS durant sa phase de login, et à l'attaque par falsification d'estampilles temporaires durant sa phase d'authentification.

Notons que, durant la phase de login, l'utilisateur est dans le voisinage du LN, le scénario de communication entre le LN et la passerelle GW est de plusieurs sauts. Puisque le UID a circulé plusieurs fois en clair dans le réseau, et que durant la phase de login, seuls le UID et le temps t sont vérifiés par le LN, le DdS peut survenir de deux façons différentes. Premièrement, l'intrus peut intercepter ou écouter un UID valide puis le soumet avec un faux mot de passe. Puisque le LN vérifie seulement le UID, il va transmettre ce message à la passerelle située à plusieurs sauts. Ainsi, tous les capteurs intermédiaires entre le LN et la GW vont assurer la propagation d'une fausse requête à travers le réseau. Ce qui va entrainer une perte d'énergie considérable au niveau

de tous ces capteurs assurant le relai. Cette transmission d'un saut de l'intrus entrainera plusieurs retransmissions dans le réseau, ce qui à la longue épuise l'énergie des capteurs. Deuxièmement, ce scénario peut arriver d'une autre façon. Puisque les mots de passe n'apparaissent jamais en clair lors de leur saisie, si un utilisateur se trompe de saisie de mot de passe, le même effet se reproduit. La transmission étant généralement l'opération la plus coûteuse en énergie, la propagation de fausses requêtes doit être évitée.

Le protocole est aussi vunérable lors d'attaque par falsification d'estampilles temporaires échangées durant sa phase d'authentification. Vaidya et al.[101] ont supposé qu'un attaquant ayant capturé un LN, obtient UID, X, TS, et qu'il lui est possible d'écouter UID, A, t afin de monter une attaque par falsification sur Wong et al. [104]. Partant de ces suppositions, leur protocole, de façon différente, sera vulnérable à l'attaque par falsification des temps transmis entre les différentes entitées impliquées. Puisque seul le délai de transmission est vérifié, l'attaquant peut créer deux faux temps T'_0 et t' dont la différence respecte le délai de propagation en y ajoutant un petit nombre ξ partout sur les deux temps T_0 et t déjà transmis en clair dans le réseau. Ainsi, il effectue T'_0=T_0+ξ ,t'=t+ξ, ensuite il calcule C'_K = H(X⊕A⊕T'_0) puis envoie le message (UID, $C'_K T'_0$,t'. Puisque (T_1 - T'_0)≤△T et (T'_0-t')≤△T le message passe.

2.3.4/ Solution proposée

Dans cette partie, nous proposons une nouvelle solution afin de résoudre les faiblesses notées dans Vaidya et al. [101]. Cette nouvelle solution comporte les mêmes phases : une *phase d'enregistrement,* une *phase de login*, une *phase d'authentification*, et une *phase de changement de mot de passe* qui est la seule à ne pas avoir subi de changements dans cette nouvelle solution proposée.

1. *Phase d'enregistrement :* dans Vaidya et al., l'utilisateur choisit un mot de passe PW, puis calcule vpw=H(PW) et envoie vpw avec son identité pour se loguer. Et pour le reste du protocole, le mot de passe PW n'est plus utilisé. Il n'est pas nécessaire de calculer vpw, qui est autant vulnérable que PW. L'utilisateur peut choisir tout simplement un mot de passe puis le soumettre, et c'est le H(PW) qui sera stocké par la passerelle GW et les nœuds de login LNs.

   Ainsi, dans cette phase d'enregistrement de la solution proposée, l'utlisateur UD choisit librement son mot de passe PW qu'il va soumettre avec son identité à la passerelle GW en mode sécurisé. La passerelle GW calcule X = H(UID||x) puis répond à l'utilisateur le message Succ_Reg(X) avec X pour lui notifier que l'enregistrement est effectué avec succès. Elle stocke les paramètres (UID, H(PW), X, TS), et distribue (UID, X, H(PW),TS) aux nœuds login LNs capables de fournir des interfaces aux utilisateurs pour se loguer.

Algorithm 26 *: Etapes de la Phase d'Enregistrement(PE)*

PE1 - - UD : choisit son mot de passe PW ;

PE2 - - UD ⟶ GW : UID, PW ;

PE3 - - GW : calcule X = H(UID || x)

Store UID, H(PW), X, TS ;

PE4 - - GW ⟶ UD : Succ_Reg (X) ;

PE5 - - UD : Stocke X ;

PE6 - - GW ⟶LNs : UID, X, H(PW), TS

PE7 - - LN : Stocke UID, X, H(PW),TS ;

2. *Phase de login :* l'utilisateur calcule A= H(H(PW)||t) et soumet (UID, A, t) au LN. Après réception de la requête au temps T_0, le LN vérifie : si l'identité UID est non valide ou A ≠ H(H(PW)||t), ou T_0 - t ≥△, le message de login est rejeté, sinon, le LN calcule C_K = (X⊕A⊕T_0) et t'=H(H(PW)||key⊕t) ; puis envoie (UID,C_K,T_0,t') à la passerelle GW.

Algorithm 27 *: Etapes de la Phase de Login (PL)*

PL1 - - UD : calcule A = H(H(PW)||t) ;

PL2 - - UD : ⟶ LN : UID, A, t ;

PL3 - - LN : **Si** UID est non valide

- **Alors** rejette le message de login ;
- **Sinon** calcule A' = H(H(PW)||t) ;
- **Si** A≠A'
- **Alors** rejette le message de login ;
- **SinonSi** T_0 - t≥△T
- **Alors** rejette le message de login ;
- **Sinon** récupère le paramtre A correspondant ;
- calcule C_K = (X ⊕A ⊕ T_0) ;
- calcule t'=H(H(PW)||key)⊕t ;

PL4 - - LN ⟶ GW :UID,C_K,T_0,t'

3. *Phase d'authentification :* durant cette phase, la passerelle vérifie si le UID et le temps t sont valides. Le message de login est rejeté s'ils ne le sont pas. Ensuite il calcule t=t'⊕H(H(PW)||key), puis vérifie si T_1 - T_0≥△T et T_0 - t≥△T. Si au moins une condition est satisfaite, alors le message de login est considéré comme répété et est rejeté. Sinon la passerelle GW récupère les paramètres H(PW) et A correspondants puis calcule A' =H(H(PW)||t) et C'_K = (X⊕A'⊕ T_0). Le message de login est rejeté si C_K≠C'_K, sinon la GW calcule V_M = H(X||A'||T_1) puis envoie le message (Acc_login, V_M, T_1) au LN et stocke t. Le LN calcule V'_M, et après vérification que V_M = V'_M, il calcule Y_K = H(V'_M||T_2). Ensuite il envoie le message (Acc_login, Y_K, T_1, T_2) à l'utilisateur UD. Après réception du message au temps T_3, l'utilisateur UD vérifie si T_1 - T_0≥△T et T_0 - t≥△T. Si au moins une condition est vérifiée, le message d'acceptation de login est rejeté. Sinon, l'utilisateur calcule, $V"_M$ = H(X||A||T_1) et Y'_K = H($V"_M$||T_2), puis vérifie si Y_K =Y'_K. Si la condition est vraie, l'utilisateur commence à obtenir les données, sinon il rejette le message d'acceptation de login.
4. *Phase de changement de mot de passe :* cette phase n'a pas changé, elle est la même que celle de Vaidya et al..

2.3.5/ Analyse de sécurité de la solution proposée

Dans cette section, nous passons à l'analyse de la solution proposée afin de montrer qu'elle est résistante à plusieurs types d'attaques. Nous terminons la section par une étude comparative avec d'autres solutions existantes.

– *Securité*
  – *L'attaque par Deni de Service :* en donnant aux LNs la possibilité de vérifier le mot de passe lors de la phase de login, notre solution se protège contre le DdS. Puisque le LN stocke H(PW), après avoir reçu le message de login (UID, A,t), il peut calculer

Algorithm 28 *: Etapes de la Phase d'Authentification (PA)*

PA1 - - GW : **Si** UID et t sont valides
- **Alors** récupère (UID, X, TS, H(PW))
- calcule t=t'⊕H(H(PW)||key) ;
- **Sinon** rejette le message de login ;
- **Si** T_1 - T_0≥△T et T_0 - t≥△T
- **Alors** message de login supposé répété et rejeté ;
- **Sinon** calcule A' = H(H(PW)|| t ;
- calcule C'_K = (X⊕A'⊕ T_0) ;
- **Si** $C'_K \neq C_K$
- **Alors** rejette le message de login ;
- **Sinon** calcule V_M = H(X||A'||T_1) ;
- Stocke t ;

PA2 - - GW ⟶ LN : Access_login,V_M,T_1 ;

PA3 - - LN : **Si** T_2 - T_1≥△T
- **Alors** rejette le message de login supposé répété ;
- **Sinon** calcule V'_M = H(X||A||T_1) ;
- **Si** $V_M \neq V'_M$
- **Alors** rejette le message de login ;
- **Sinon** calcule Y_K=H(V'_M||T_2) ;

PA4 - - LN ⟶UD : Acces_login,Y_K,T_1,T_2 ;

PA5 - - UD : **Si** T_1 - T_0t≥△T et T_0 - t≥△T
- **Alors** rejette le message (Acces_login) ;
- **Sinon** calcule V''_M = H(X||A||T_1) ;
- calculeY'_K = H(V''_M||T_2) ;
- **Si** $Y_K \neq Y'_K$
- **Alors** rejette le message (Acces_login) ;
- **Sinon** UD commence à obtenir les données ;

A'=H(H(PW)||t). Si A'=A, alors le mot de passe est correct, sinon le message de login est rejeté.

– *L'attaque par falsifcation :* la solution proposée protège également contre la falsification des estampilles temporaires envoyées en clair entre le LN et la GW dans Vaidya et al. Nous proposons de les envoyer en mode sécurisé avec une simple utilisation du OU exclusif. C'est ainsi que le LN, après avoir reçu le message de login, calcule une fausse estampille *t'*= *t*⊕H(H(PW)||key) puis envoie le message (UID,C_K,T_0,*t'*) au lieu de (UID,C_K,T_0,*t*) à la passerelle GW. Ce qui rend l'estampille t confidentiel. Puisque H(H(PW)||key) ⊕t⊕H(H(PW)||key)=*t*, la passerelle GW calcule *t*=*t'*⊕H(H(PW)||key) pour retrouver l'estampille *t*.

– *Comparaison avec d'autres solutions*

– Le Tableau 2.2, donne une comparaison en termes d'opérations de hachage, de OU exclusif ainsi que le nombre de communications multi-sauts de quelques solutions utilisant la même approche.

Protocoles	*Nombre total d'opérations*
Wong et al.[104]	$7T_H$+$4T_{XOR}$+$3C_{MH}$
Tseng et al.[99]	$5T_H$+$4T_{XOR}$+$3C_{MH}$
Vaidya et al. [11]	$8T_H$+$4T_{XOR}$+$3C_{MH}$
Vaidya et al. [101]	$11T_H$+$4T_{XOR}$+$3C_{MH}$
Solution Proposée	$15T_H$+$7T_{XOR}$+$3C_{MH}$

TABLE 2.2 – Comparaison du nombre d'opérations effectuées
T_H : temps pour exécuter la fonction de hachage H().
T_{XOR} : temps pour exécuter l'opération XOR .
C_{MH} : Délai de communication multi-sauts entre le LN et la GW.
NB : Ces opérations peuvent aussi être traduites en énergie.

D'après le Tableau 2.2, on peut remarquer que la solution proposée a un coût supérieur de quatre opérations de hachage (T_H) et de trois opérations OU exclusif (T_{XOR}) à celle de Vaidya and al.[8]. Notons que, le coût énergétique de l'opération OU exclusif est largement inférieur à celui du hachage. Ainsi, notre solution se retrouve avec un léger surplus de consommation d'énergie tout en produisant une meilleure sécurité.

– Considérons une requête avec une UID valide mais avec un mot de passe erroné qui proviendrait de l'intrus ou d'une erreur de saisie de la part d'un utilisateur légitime. Pour notre solution, cette requête de login sera freinée au niveau du LN, et pour les autres solutions, elle sera propagée jusqu'à la passerelle GW. Pour ce cas, le Tableau 2.3 montre une comparaison entre notre solution et celles de Vaidya et al..

Protocoles	*Nombre total d'opérations*
Vaidya et al. [11]	$4T_H$+$2T_{XOR}$+$2C_{MH}$
Vaidya et al. [101]	$4T_H$+$2T_{XOR}$+$2C_{MH}$
Solution Proposée	$3T_H$

TABLE 2.3 – Comparaison du nombre d'opérations effectuées

On voit ici que notre solution présente moins d'opérations, par conséquent sa consommation d'énergie sera meilleure. Et en plus, notons que, cet écart d'énergie va augmenter considérablement dans les protocoles de Vaidya et al. en fonction du nombre de sauts entre le LN et la GW, alors que pour notre solution qui ne fait pas intervenir une communication multi-saut (C_{MH}) pour les fausses requêtes, l'energie sera constante.

2.3.6/ SIMULATION ET IMPLÉMENTATION

2.3.6.1/ SIMULATION

Nous rapellons qu'il y a deux protocoles de Vaidya et al.[101, 11]. Le but de notre implémentation est d'estimer la consommation énergétique en fonction du nombre de sauts entre le LN et la GW en s'appuyant sur les comparaisons effectuées dans les deux tableaux de la précédente section. Nous avons fait une implémentation de notre solution et celle de Vaidya et al. [101] avec TinyOS. Le programme est testé sur la plateforme MicaZ, et le simulateur Avrora est utilisé pour mesurer la consommation d'énergie.

A la première étape de la simulation, nous avons évalué la consommation d'énergie en fonction du nombre de sauts entre le LN et la GW en se basant sur le Tableau 2.2 où les deux protocoles sont considérés sans attaques. Pour chaque paramètre de données comme (UID,A,PW, C_k etc.) on a utilisé des données de 16 bits. Pour le hachage, nous avons utilisé une implémentation de la fonction de hachage universelle PolyR décrite dans le papier de Ted and al.[65], comme une interface TinyOS. La Figure 2.3, montre la consommation d'énergie de chaque solution.

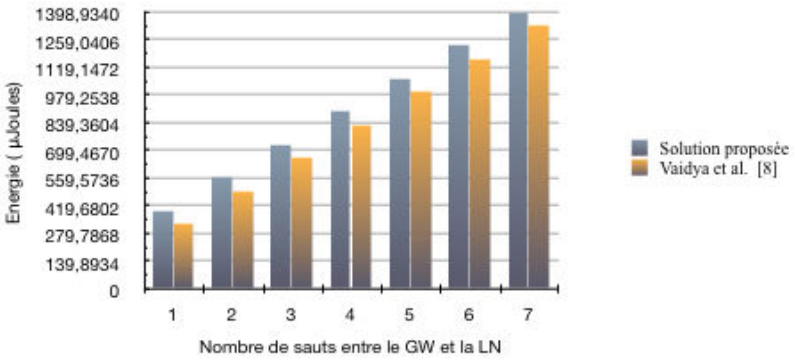


FIGURE 2.3 – Consommation énergétique basée sur le Tableau 2.2

A la deuxième étape, nous nous intéressons à l'effet de propagation d'une fausse requête sur la consommation énergétique. Ainsi, nous avons utilisé le Tableau 2.3 pour évaluer l'énergie consommée en fonction du nombre de sauts entre le LN et la GW. La Figure 2.4 montre la consommation d'énergie de chacune des solutions. Nous pouvons voir que l'énergie reste constante pour notre solution car la fausse requête ne se propage pas. Elle augmente en fonction du nombre de sauts dû aux transmissions des capteurs assurant la propagation de la fausse requête.

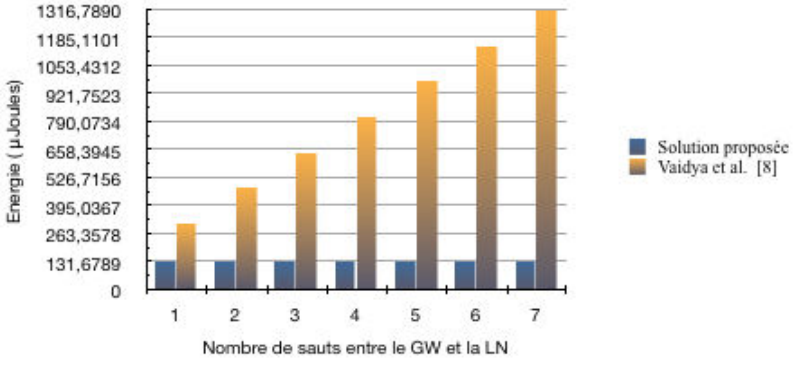


FIGURE 2.4 – Consommation énergétique basée sur le Tableau 2.3.

2.3.6.2/ IMPLÉMENTATION

Dans la suite de nos tests, nous implémentons les algorithmes d'authentification pour un usage dans un cas réel. L'utilisation d'un réseau de capteurs dans le domaine de la médecine pourrait permettre une surveillance permanente des patients, par exemple les personnes âgées, et une possibilité de collecter des informations physiologiques de meilleure qualité, facilitant ainsi le diagnostic. Ainsi, on pourra suivre plus facilement l'évolution de la pathologie d'une personne, historiser l'état de santé sur le dossier médical personnel, transmettre des alertes au médecin ou de l'ima-

gerie médicale pour un soin plus rapide. Dans ce sas, le déploiement d'un RCSF pour un hôpital, consiste à utiliser des LNs déployés pour l'accès des utilisateurs et une passerelle vers l'internet qui fait en même temps office de serveur d'authentification. Dans nos tests, il a été question de faire le compromis en termes de consommation énergétique entre les opérations de sécurité et celles de transmission des données. Dans des cas d'utilisation comme celui du pacemaker où le capteur est inséré dans le corps du patient, le besoin d'économiser l'énergie devient important afin d'éviter des opérations d'insertion multiples. Notre implémentation a été testée sur deux sites différents de Senslab. Senslab est une plateforme de réseaux de capteurs composée de 1024 nœuds, répartis sur 4 sites, à raison de 256 nœuds par site. L'expérience a été menée sur deux sites différents :
– Un site doté d'une plate forme à deux dimensions avec des capteurs wsn430 : cc1101 [61]
– Un autre doté d'une plate forme à trois dimensions avec des capteurs wsn430 :cc2420 [62]
La documentation technique indique que la consommation en courant électrique des wsn430 avec le composant radio CC1101 est repartie dans les états suivants :
– En réception (RX) : 16,9mA
– En transmission (TX) : entre 17,0mA et 34,2mA selon la puissance d'émission
– En état de veille (SLEEP) : 0,2μA
Les caractéristiques principales de quelques nœuds capteurs sans fil sont représentées dans le Tableau 2.4.

Nœud capteur	***MicaZ***	***TelosB***	***WSN430***
Processeur	Atmel AT-Mega 128L	TI MSP430	TI MSP430
Vitesse processeur	16 MHz	*8 MHz*	8 MHz
Taille RAM	4 Ko	10 Ko	10 Ko
Espace programme	128 Ko	48 Ko	48 Ko
Radio	TI CC2420 IEEE 802.15.4		TI CC1100
Fréquence	2400-2483		315/433/868/915
Voltage	2,7 V	1,8-3,6 V	3,7 V

TABLE 2.4 – Caractéristiques de quelques nœuds capteurs sans fil

Les résultats récupérés à partir de la plateforme sont : la tension en *volt*, l'intensité en *ampère*, la puissance en *watt*. La consommation de l'énergie électrique a été calculée selon le modèle énergétique linéaire défini par l'équation suivante :

$$E = P_1 t_1 + P_2 t_2 + P_3 t_3 + \ldots\ldots + P_{n-1} t_{n-1} + P_n t_n \tag{2.3}$$

- Avec P_i : la puissance moyenne du nœud capteur i à l'instant t_i.
- Energie moyenne $E_m = E/n$ où n est le nombre de capteurs.

L'échantillonnage des mesures s'est fait périodiquement toutes les 5000ms pendant une minute, avec des capteurs configurés en mode « alimentation par batterie ».

Dans un premier temps, le chemin des paquets est prédéfini dans un modèle de communication avec une distance de 5 sauts entre la GW et l'utilisateur. 6 capteurs sont nécessairement utilisés : une passerelle , 3 nœuds relais, un nœud login et un utilisateur. La Figure 2.5 montrent l'évolution de l'énergie moyenne des 6 nœuds capteurs en fonction du temps. Dans un deuxième temps, nous avons lancé la même expérience sur une plateforme à trois dimensions. La Figure 2.6 montre l'évolution de l'énergie moyenne consommée par les 6 nœuds en fonction du temps.

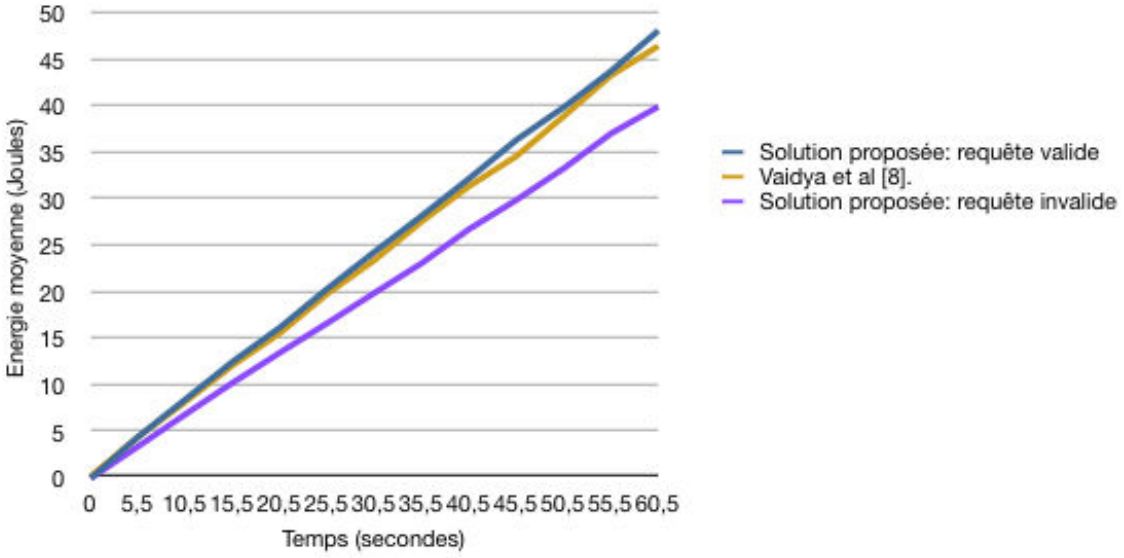


FIGURE 2.5 – Consommation énergétique en 2D

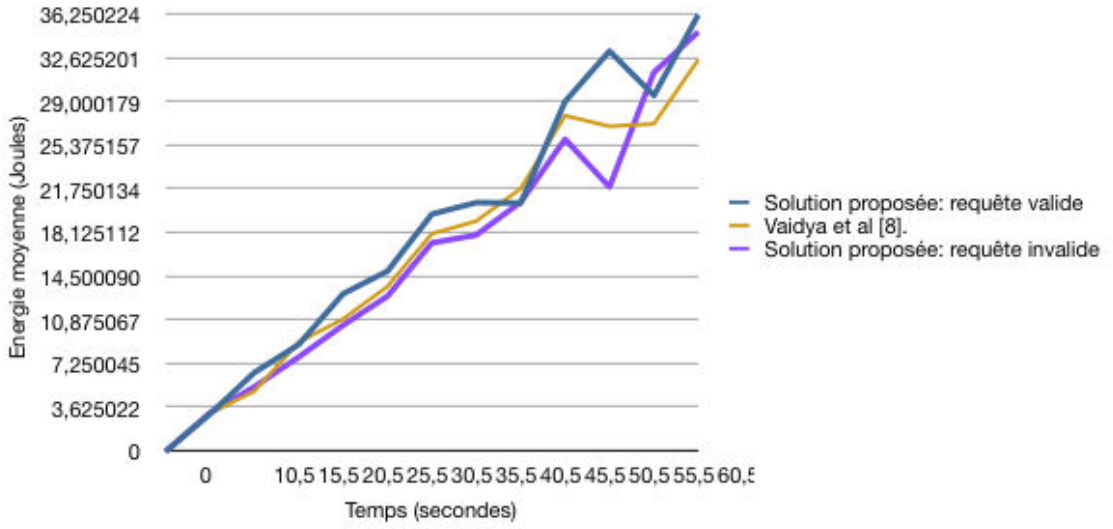


FIGURE 2.6 – Consommation énergétique en 3D

Comme nous pouvons le constater, les tendances sont les mêmes que dans la partie simulation. La solution proposée est relativement plus économe en énergie que celle de Vaiya et al.[101] lorsqu'une requête invalide est envoyée.

2.3.7/ CONCLUSION

Dans ce chapitre, nous avons proposé d'optimiser le protocole de Vaiya et al.[101] afin d'avoir un protocole beaucoup plus sûr dans l'environnement des capteurs. La solution proposée conserve tous les avantages de celle de Vaidya et al. et améliore sa sécurité par la protection contre le DdS et la falsification. Dans certains cas, elle permet une meilleure sécurité avec seulement un coût énergétique additionnel de quatre opérations de hachage (T_H) et de trois opérations OU exclusif

(T_{XOR}). Et dans d'autres cas, nous avons aussi montré dans l'implémentation, qu'elle est bien meilleure en consommation énergétique car elle permet d'effectuer au moins une opération de hachage (T_H), de deux opérations OU exclusif (T_{XOR}) et deux opérations (C_{MH}) de moins que celle de Vaidya et al.. Cette différence augmente avec le nombre de sauts entre le LN et la GW. Dans la suite de nos travaux, nous introduisons la probabilité de risque à partir de laquelle le comportement de chaque solution sera étudié. Ce qui permettra de déterminer énergétiquement la meilleure solution pour une architecture donnée d'un RCSF.

2.4/ Contribution 2 : Authentification basée sur une analyse probabiliste de risque d'attaques par DdD

Cette contribution est une suite logique de la première. Dans la première partie, après avoir introduit le concept de probabilié de risque, nous utilisons le modèle de l'attaquant pour étudier le comportement des solutions basées sur l'architecture de déploiement (distance entre les LNs et la GW) face aux risques du milieu. Ce qui permettra de déterminer pour une architecture physique de RCSF dans un environnement à degré de risque ou de vulnérabilité donné, laquelle des solutions est énergétiquement meilleure. Dans la deuxième partie, nous procédons à une vérification automatique de quelques propriétés de base de sécurité relatives à des services de sécurité à l'aide de l'outils de validation AVISPA.

2.4.1/ Probabilité de risque

Nous assimilons ici la probabilité de risque à la probabilité qu'un événement à caractère malveillant survienne. Dans le cas d'une attaque par DdS montrée dans le protocole de Vaidya et al., la probabilité de risque est la probabilité qu'une fausse requête arrive. Notons que cet événement peut provenir d'un utilisateur légitime par manque d'attention ou d'un attaquant. Schématiquement, la probabilité de risque peut être représentée par la présence de l'attaquant. Plus il y a d'intrus, plus le risque est élevé. Si on sait qu'un intrus plus actif peut lancer plus de requêtes, que plusieurs intrus peu ou moyen actifs, on peut dans ce cas quantifier la probabilité de risque par le rapport entre le nombre de fausses requêtes envoyées sur le nombre total de requêtes (fausses et légitimes).

Après avoir introduit cette notion de probabilité de risque, relative à la probabilité d'une fausse requête, nous allons dans la suite déterminer analytiquement la consommation énergétique de chaque solution basée sur cette probabilité. Les variables utilisées dans notre analyse sont consignées dans la Tableau 2.5.

Signification des variables	*Variables*
Energie consommée sans faux mot de passe (Figure 2.3)	E_v
Energie consommée avec faux mot de passe (Figure 2.4)	E_w
Energie consommée avec une probabilité P_i	E_i
Probabilité de faux mot de passe	P_i

Table 2.5 – Variables utilisées dans l'analyse

Selon une probabilité de distribution uniforme P_i d'une fausse requête, E_i désigne l'énergie consommée dans la formule suivante :

$$E_i = P_i * E_w + (1 - P_i) * E_v \tag{2.4}$$

A partir de cette formule et du nombre de sauts entre le LN et la GW, nous pouvons déterminer la probabilité à partir de laquelle chaque solution permet une meilleure consommation énergétique. Les résultats sont présentés sur la Figure 2.7 pour un saut, Figure 2.8 pour deux sauts et Figure 2.9 pour quatre sauts. On peut voir que notre solution présente des avantages significatifs. Comparée à la solution de Vaidya et al., plus la distance entre le LN et la GW augmente, plus la solution proposée est efficace dans des milieux même de moins en moins vulnérables. Les meilleurs résultats sont atteints à partir d'une probabilité $P_i \geq 0{,}3$ pour un saut, $P_i \geq 0{,}13$ pour deux sauts $P_i \geq 0{,}09$ pour quatre sauts.

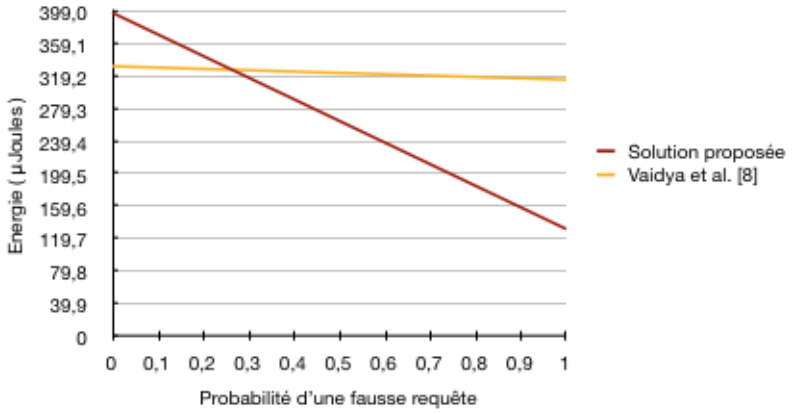


FIGURE 2.7 – Energie à 1 saut entre le LN et la GW

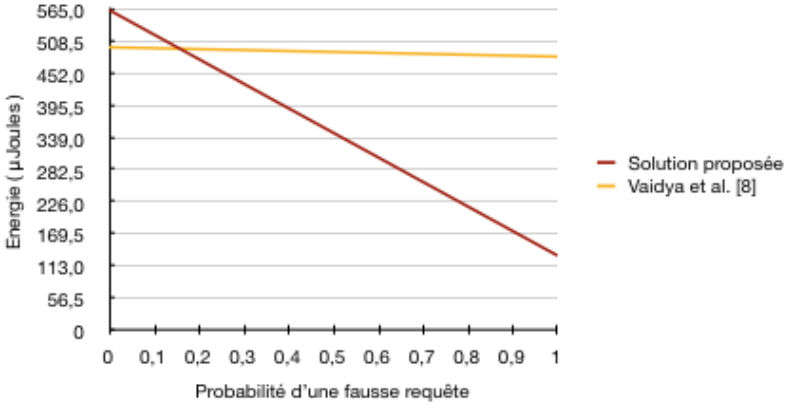


FIGURE 2.8 – Energie à 2 sauts entre le LN et la GW

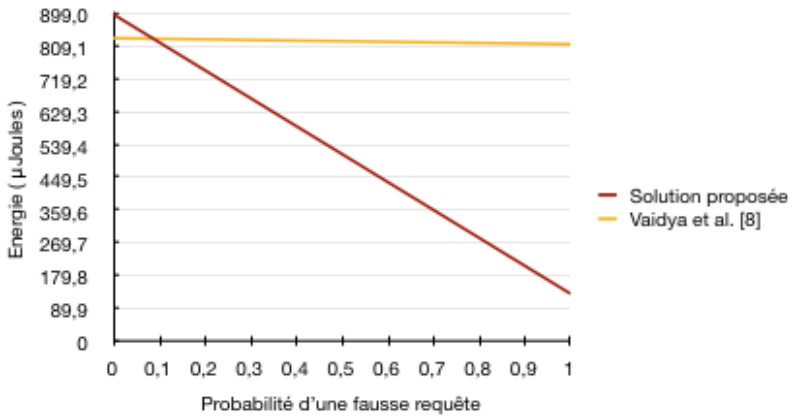


FIGURE 2.9 – Energie à 4 sauts entre le LN et la GW

2.4.2/ VALIDATION PAR ANALYSE ET VÉRIFICATION DES PROPRIÉTÉS DE SÉCURITÉ

Dans cette section, nous utilisons le standard AVISPA (Automated Validation of Internet Security Protocols and Applications), un outil de validation automatique des protocoles de sécurité d'Internet et d'applications afin de valider notre solution [7, 105]. AVISPA permet l'analyse automatique et la vérification de protocoles de sécurité. La capacité de l'outil AVISPA à traiter des protocoles a été démontrée à travers une librairie de protocoles originaires de IETF (Internet Engeeniring Task Force) spécifiés en HLPSL (High Level Protocol Specification) [28], puis chacun des protocoles a fait l'objet d'une vérification avec l'outil AVISPA. Notre solution a été validée via une vérification de quelques propriétés de sécurité relatives à l'authentification et à la confidentialité.

2.4.2.1/ ARCHITECTURE DE AVISPA

Comme on peut le voir sur la Figure 2.10 , on peut distinguer cinq parties dans l'architecture d'AVISPA.

HLPSL (High Level Protocol Specification Language) : Langage de haut niveau créé au cours du projet AVISPA pour spécifier, décrire les protocoles à analyser.

Translator HLPSL2IF (traducteur) : permet de traduire la spécification HLPSL en une spécification IF à partir de laquelle AVISPA effectue la vérification

IF (Intermediate Format) : le format intermédiaire exprimé dans un langage de spécification proche des langages d'entrée des outils de vérification

Les back-end (TA4SP, CL-AtSe, OFMC, SATMC) : des outils qui permettent d'analyser et de vérifier le format intermédiaire.

OF (Output Format) : format de sortie qui contient les résultats.

Les back-end, à savoir les outils CL-AtSe [89, 91], OFMC[14] et SATMC [8] détectent des attaques pour un nombre fini de sessions, l'outils TA4SP (Tree Automata based on Automatic Approximations for the Analysis of Security Protocols) [105] quant à lui se distingue par le fait qu'une propriété de secret vérifiée l'est pour un nombre non-borné de sessions.

– OFMC : effectue une vérification bornée en explorant le système de transitions décrit par une spécification IF. Il supporte la spécification des opérateurs à propriétés algébriques tels que le OU exclusif ou encore l'exponentielle.
– CL-AtSe : c'est un outil basé sur des techniques de résolution de contraintes et implémentant une procédure de décision pour traiter le problème d'insécurité pour un nombre borné de sessions en

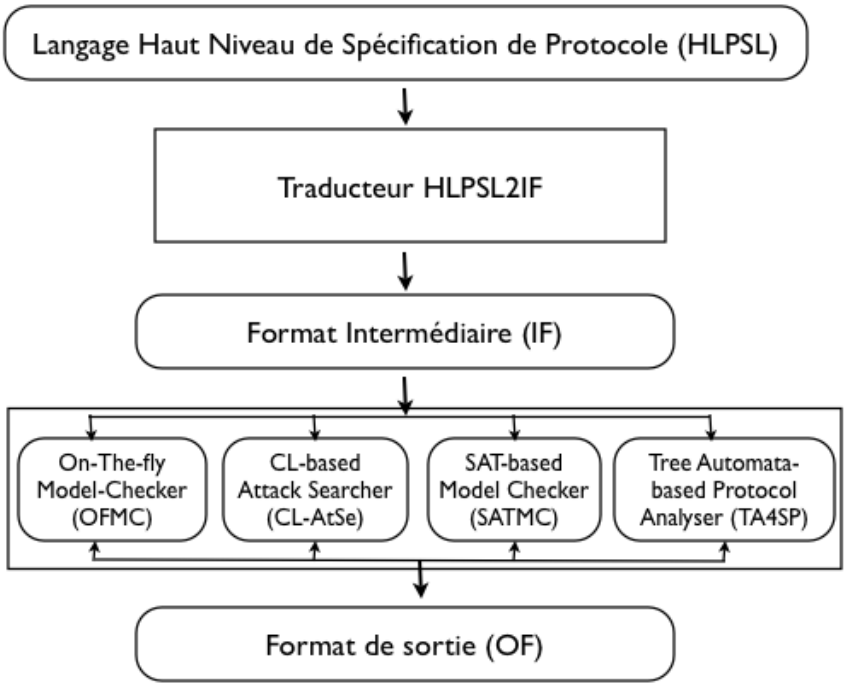


FIGURE 2.10 – Architecture de AVISPA

implantant un algorithme d'unification spécifique et en considérant les opérateurs à propriétés algébriques comme le OU exclusif.
– SATMC : il permet de construire une formule propositionnelle codant un déploiement borné du système de transition IF, l'état initial et l'ensemble des états représentant la violation des propriétés de sûreté spécifiées en IF.
– TA4SP : son but est de permettre la vérification de protocoles avec un nombre non-borné de sessions. C'est une approche pour la validation de protocoles complémentaires aux autres approches, plutôt destinées à la détection d'attaque(s).

2.4.2.2/ LES PROPRIÉTÉS DE SÉCURITÉ

On distingue les propriétés de base que sont le *secret* et l'*authentification*, et les propriétés secondaires à savoir l'*anonyma*, la *non-répudiation*, la *fraîcheur* etc..

– **Secret :** le secret d'une donnée est sa confidentialité. Pour une session, on dit qu'une donnée s est secrète si la session qui contient la donnée s est indistinguable de toute session contenant une donnée s' en lieu et place de s [33]. Une donnée peut être secrète pendant tout le temps ou seulement pendant une session donnée. La première notion est plus facile à modéliser puisqu'il suffit d'exprimer que l'intrus ne peut jamais déduire le secret. La seconde propriété demande un modèle plus précis qui permet d'exprimer les débuts et les fins de sessions. En HLPSL, la définition de la propriété de secret est relative à un événement :

$$\textbf{secret}(X, id, \{A_1, \ldots, A_n\})$$

Indique que la donnée X est secrète et partagée entre les agents A_1,..., A_n.

– **L'authentification :** l'authentification est le fait de s'assurer que l'entité communicante est celle qu'elle prétend être. Pour la plupart des outils de vérification, la vérification de la propriété d'authentification est liée à celle du secret. Ainsi, au lieu de prouver qu'un protocole de sécurité satisfait la propriété d'authentification pour laquelle il a été conçu, on vérifie souvent une propriété de secret dont on pense qu'elle est équivalente [33]. En HLPSL, on distingue deux degrés

d'authentification : *l'authentification faible* que nous comparons ici à une authentification unilatérale et *l'authentification forte* que nous comparons à une authentification mutuelle. La propriété d'authentification faible permet d'exprimer le fait qu'une entité *X* doit pouvoir identifier une entité *Y* avec un critère donné. La propriété d'authentification forte permet d'exprimer le fait qu'une entité *X* doit pouvoir identifier une entité *Y* avec un critère donné et inversement. Dans HLPSL, la définition de la propriétés d'authentification est relatifve aux événements :

witness(*Y,X,id,Z*)

L'agent *Y* déclare qu'il veut communiquer avec *X* et que la valeur *Z* permettra d'authentifier l'agent *X*.

request(*X,Y,id,Z,SID*)

L'agent *X* accepte la valeur *Z* et souhaite que

1. l'agent *Y* existe réellement
2. Et que *Z* a été déterminé par *Y* pour l'authentification de *X*.

La donnée *id* permet d'identifier chacune des propriétés. La donnée *SID* représente un identifiant de session. Cet identifiant est important car il permet également de détecter des attaques de rejeu. Soit Evs l'ensemble des des évenements déclenchés lors de la construction du système.

***Authentification forte* :** la propriété d'authentification forte ayant pour identifiant *id* est vérifiée si :

1. Pour tout *witness*(*x, y, id, m*) ∈ à Evs, il existe *request*(*y, x, id, m, SID*) ∈ à Evs
2. Et pour tout *request*(*y, x, id, m, SID*), request(*y, x, id, m, SID'*) ∈ à Evs, *SID*=*SID'*.

En ce qui concerne la notion d'authentification forte, nous avons donc deux points à vérifier. D'abord, qu'il y ait bien une notion d'authentification exprimée par le point 1), mais aussi, qu'il n'y ait pas d'attaque de rejeu exprimée par le point 2).

***Authentification faible* :** la propriété d'authentification faible consiste en la vérification du point 1) ci-dessus.

2.4.2.3/ Validation de la solution proposée

Afin de valider notre solution avec AVISPA, nous avons procédé à la vérification de la propriété de *secret* du mot de passe stocké et de l'*authentification forte* des LNs par la GW. Nous avons choisi le back-end Cl-atse, qui est un outils permettant d'analyser et de vérifier le format intermédiaire. Ce choix se traduit par le fait qu'il supporte les opérateurs et fonctions utilisés dans la description de notre protocole.

– *Secret du mot de passe stocké* : notre première simulation doit garantir que le mot de passe stocké doit rester confidentiel entre l'utilisateur, les LNs et la GW. Cette partie implique dans deux sessions différentes l'utilisateur, la GW, le LN et l'intrus . La première session que nous appellerons ici *session normale* est composée de l'utilisateur légitime, d'un LN et de la GW. La deuxième session implique l'intrus qui va jouer le rôle d'un l'utilisateur en vue d'attaquer le protocole . Puisque la GW est censée être une entité sûre, donc l'intrus ne peut pas se faire passer pour la GW auprès de l'utilisateur et/ou du LN. Ainsi, dans cette deuxième session, l'intrus va se faire passer pour un utilisateur auprès des LNs et de la GW.

   1. Session normale : Utilisateur (u) - LN (l) - GW (g)
   2. Session intrus : Intrus (i) - LN (l) - GW (g)

   La Figure 2.11 représente le résultat obtenu avec AVISPA. On voit dans le résultat que le protocole est *SAFE*, c'est à dire sûr, pas d'attaques retournées.

– *Secret et authentification forte :* cette deuxième partie de la simulation doit garantir que le mot de passe stocké est confidentiel et à la fois une authentification forte entre les LNs et la GW.

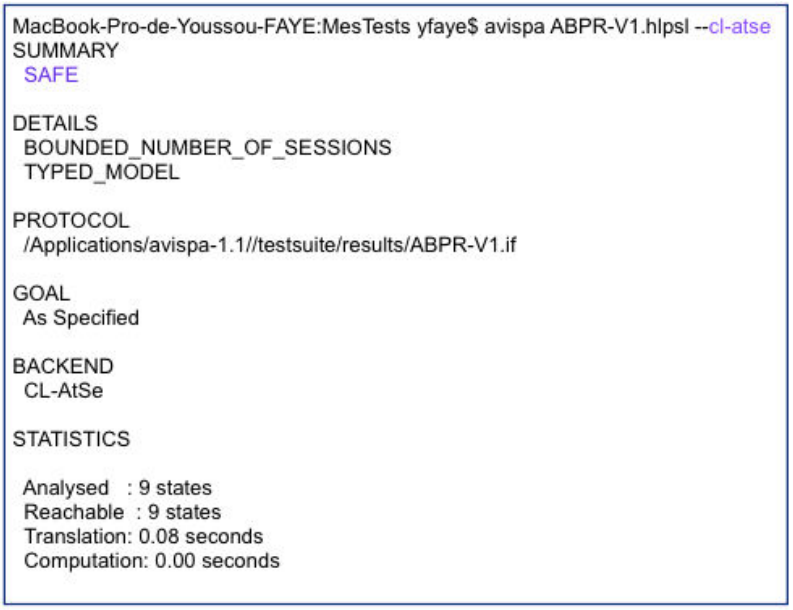


FIGURE 2.11 – Résultat sur la propriété de *secret* du mot de passe

Cette authentification permet aux LNs de s'assurer qu'ils communiquent bien avec la GW car cette dernière assure l'authentification finale des utilisateurs. Ainsi, on distingue trois sessions différentes qui impliquent l'utilisateur, la GW, le LN et l'intrus. On a une *session normale*, une deuxième session dans laquelle l'intrus va jouer le rôle de l'utilisateur, et la troisième session dans laquelle l'intrus va jouer le rôle du LN. Puisque la GW est censée être une entité sûre, donc l'intrus ne peut jouer son rôle auprès de l'utilisateur et/ou du LN.

1. Session normale : Utilisateur (u) - LN (l) - GW (g)
2. Session 1 intrus : Intrus (i) - LN (l) - GW (g)
3. Session 2 intrus : Utilisateur (u) - Intrus (i) - GW (g)

La Figure 2.11 montre le résultat obtenu avec AVISPA.

– ***Discussion et analyse des résultats :***

Dans cette deuxième partie, l'intrus a plus de privilèges. Il peut se faire passer à la fois pour un utilisateur auprès des LNs et pour un LN auprès des utilisateurs. On peut assimiler l'intrus au réseau car tous les messages passent par lui. On peut voir dans le résultat que le protocole est déclaré *UNSAFE*, et une attaque est retournée. Cela ne veut toujours pas dire que quand il y a une attaque, la sécurité est compromise. Si on analyse les traces de l'attaque, on peut constater que l'intrus s'est fait passer pour un LN auprès de l'utilisateur, ce qui se décrit à travers les évenements suivants :

– i → (u, 11) : start

L'intrus *i* initialise une session avec l'identifiant (11) en envoyant un signal « *start* » (composé de l'identité de l'utilisateur *u*, et de la session (*11*) à l'utilisateur pour que ce dernier lui envoie en retour un message crypté avec k_i (clé fabriquée par l'intrus et envoyée à l'utilisateur).

– (u, 11) → {u.n25(PW)}_ki & Secret(u.n25(PW), set_127), Add u to set_127, Add g to set_127 ;

Dans la même session (11), l'évenement (u, 11) → {u.n25(PW)}_k_i indique que l'utilisateur (u) envoie en retour à l'intrus, son identité concaténée avec le mot de passe (d'identité n25) cryptés avec la clé k_i. Le frangment & *Secret(u.n25(PW), set_127)* indique que le *PW* identifié par *n25* est secret et est stocké dans la variable *set_127* générée automatiquement. Enfin les deux événements *Add u to set_127et Add g to set_127* indiquent que seules les entitées u (utilisateur)

```
MacBook-Pro-de-Youssou-FAYE:MesTests yfaye$ avispa ABPR-V1.hlpsl --cl-atse
SUMMARY
 UNSAFE

DETAILS
 ATTACK_FOUND
 TYPED_MODEL

PROTOCOL
 /Applications/avispa-1.1//testsuite/results/ABPR-V1.if

GOAL
 Secrecy attack on (n25(PW))

BACKEND
 CL-AtSe

STATISTICS

 Analysed  : 0 states
 Reachable  : 0 states
 Translation: 0.09 seconds
 Computation: 0.00 seconds

ATTACK TRACE
 i -> (u,11):  start
 (u,11) -> i:  {u.n25(PW)}_ki
          & Secret(n25(PW),set_127);  Add u to set_127;  Add g to set_127;
```

FIGURE 2.12 – Résultat sur la propriété de **secret** et d'authentification forte

et g (gateway) sont autorisées à connaître la variable. On peut conclure donc que l'attaque est menée mais l'intrus *i*, ne connaît pas le secret après cette tentative.

2.4.3/ CONCLUSION

Dans cette deuxième partie, nous avons procédé à une analyse basée sur l'architecture et les vunérabilités du réseau. Par ailleurs, nous avons effectué une vérification automatique des propriétés de secret et d'authentification par AVISPA ; ce qui a permis de déterminer la meilleure solution en termes d'énergie basée sur l'architecture et les risques du milieu de déploiement. Après une analyse que nous jugeons pertinente de la sécurité de la solution proposée dans la première contribution, la vérification automatique des propriétés de sécurité permet en plus, de nous rassurer sur sa sécurité. Nous pensons dans le futur pouvoir faire une analyse plus pertinente des solutions avec des automates afin de faire intervenir d'autres paramètres indicatifs aussi importants que la consommation énergétique, comme par exemple le temps, le stockage mémoire etc. toujours en rapport avec les ressources limitées des capteurs.

3

Accélération de la multiplication scalaire sur les courbes elliptiques pour les réseaux de capteurs sans fil

Sommaire

Dans les Réseaux de Capteurs Sans fil (RCSFs), produire un mécanisme de sécurité robuste avec une faible consommation de ressources est un vrai défi à cause des ressources limitées des capteurs en termes de capacités de calcul, de transmission, de stockage mémoire et d'énergie. Lorsque le nombre de nœuds est très élevé, la cryptographie asymétrique est naturellement la meilleure méthode à cause de sa scalabilité. Même si la cryptographie asymétrique n'est pas conseillée à cause de sa forte demande de ressources, son utilisation avec les courbes elliptiques est plus flexible et donc favorable dans l'environnement des capteurs. Pour un même niveau de sécurité, les courbes elliptiques utilisent des clés plus courtes comparées aux autres systèmes asymétriques comme RSA [88].

La multiplication scalaire notée kP où k est un scalaire (clé privée) et P un point de courbe, est l'opération dominante au sein des courbes elliptiques. On la trouve dans les mécanismes de génération de clés, de chiffrement/déchiffrement et de signature/vérification. C'est aussi l'opération qui consomme le plus de temps et d'énergie. La multiplication scalaire est définie comme suit :

$$E(\mathbb{F}_p) \times \mathbb{Z} \rightarrow E(\mathbb{F}_p), (P,k) \mapsto Q = \underbrace{(P+P+P+.....+P)}_{k fois}. \tag{3.1}$$

Où $\mathbb{F}_p$ est un corps premier de caractéristique p (un nombre premier > 3).
$E(\mathbb{F}_p)$ une courbe elliptique sur le corps premier $\mathbb{F}_p$, et le scalaire $k \in \mathbb{Z}$.

Elle doit être une opération irréversible, étant donnés k un scalaire et P un point de la courbe

elliptique. Il doit être facile de faire la multiplication scalaire $[k]P = Q$, où Q est aussi un point sur la courbe. Par contre connaissant P et Q, il est difficile de retrouver le scalaire k, qui, multiplié par le point P donne le point Q. Ainsi, toute la sécurité des courbes elliptiques repose sur cette difficulté de retrouver l'entier k à partir des points P et Q : on l'appelle le problème du logarithme discret elliptique (Elliptic Curve Discrete Logarithm Problem, ECDLP).

La hiérarchie mathématique de la multiplication scalaire dans les courbes élliptiques implique trois niveaux arithmétiques[55] : sur le corps fini, sur un point de courbe ou sur le scalaire k. Le premier niveau est relatif à l'arithmétique modulaire sur le corps fini, où celle-ci comprend un ensemble d'entiers sur lequel les opérations arithmétiques (additions/soustractions, multiplications et inversions) sont calculées modulo un nombre p (premier pour les *corps finis premiers*). Le deuxième concerne les opérations sur les points comme l'addition de points, le doublement, le triplement, le quadruple, le quintuple etc. d'un point. Et le troisième niveau repose sur des développements binaires pour une représentation optimale du scalaire k .

La représentation binaire du scalaire étant au minimum de l'ordre de 160 bits pour un niveau de sécurité acceptable, il faut des algorithmes optimisés pour une exécution efficace de la multiplication scalaire. Des solutions existent pour accélérer la multiplication scalaire afin de permettre une optimisation des ressources par la réduction du temps de calcul et de la consommation énergétique. Plusieurs recherches se sont développées au niveau de l'arithmétique sur le point (par l'optimisation des formules) à travers les opérations d'addition et de doublement. D'autres reposent sur une représentation optimale souvent en binaire du scalaire afin de réduire le nombre d'opérations de points (addition, doublement etc..) comme dans les algorithmes traditionnels Double-and-Add (DA), Non Adjacent Form (NAF), wNAF etc.. Un choix pertinent du système de coordonnées, les caractéristiques de la courbe (courbes aux propriétés intrinsèques intéressantes) ou des paramètres (a, b et p) peuvent aussi impacter sur la rapidité des calculs. Dans les systèmes distribués comme les RCSFs, des solutions privilégient le partage des tâches à travers le calcul parallèle de la multiplication scalaire (voir chapitre suivant).

Dans ce chapitre, après avoir énoncé brièvement quelques solutions existantes pour accélérer la multiplication scalaire, nous présentons un mécanisme d'accélération de la multiplication scalaire sur les courbes elliptiques définies dans des corps finis. Notre mécanisme est basé sur l'*opposé* et l'*ordre* d'un point, il réduit le nombre d'opérations de points et peut être combiné avec toutes les techniques existantes.

3.1/ Arithmétique des courbes elliptiques pour l'accélération de la multiplication scalaire

Au niveau des corps finis (surtout premiers) des solutions existent pour accélérer les calculs à travers des modes de repésentation des nombres, des façons de calculer efficacement les opérations arithmétiques (addition, soustraction, multiplication, inversion) modulo un nombre premier [81, 30, 13, 92]. Dans cette section, nous allons énoncer les directions qui exploitent les formules des opérations de points et le développement binaire du scalaire.

3.1.1/ Opérations sur les points

Le principe des méthodes d'accélération de la multiplication scalaire au niveau de l'arithmétique sur les points est de rendre plus efficace l'exécution globale des opérations de doublement et d'addition de points à travers leur formule. Cela se traduit par des substitutions algébriques, des optimisations au niveau des opérations arithmétiques élémentaires telles que l'addition (différente de l'addition de points), la soustraction, la multiplication et l'inversion sur les éléments du corps fini.

Dans la suite, nous adopterons les notations suivantes : l'addition de points est notée par A_p, le doublement de point par D_p, le triplement de point par T_p, l'addition unifiée de points par A_u, la mutiplication par M, le carré par S l'inversion par I, la multiplication par une constante par C , le système de coordonnées affines par $\mathcal{A}$, projectives par $\mathcal{P}$ et jacobiennes par $\mathcal{J}$. Ainsi, les opérations coûteuses sont remplacées ou substituées, comme par exemple la substitution de la multiplication par le carré jugé moins coûteux. Il a été généralement accepté dans plusieurs implémentations, qu'un carré est l'équivalent de *0,6* à *0,8* multiplication[24, 51, 49] . Cette technique de substitution d'une opération par une autre a été utilisée par le développement de certaines opérations comme la multiplication à travers d'autres opérations moins coûteuses [76]. Par exemple soit *a, b* des éléments d'un corps fini.

$$ab = \frac{1}{2}[(a+b)^2 - a^2 - b^2] \tag{3.2}$$

On peut voir ici que la multiplication peut être remplacée par trois carrés, trois additions/soustractions et une division. Par conséquent, le remplacement direct n'est pas efficace si *1S =(0,6* ou *0,8)M*. Mais la redondance dans les formules sur les opérations de points dans les courbes elliptiques sur les corps premiers permet de calculer un carré au lieu de trois. Le coût des opérations d'addition, de soustraction, et de division par une constante dans les corps premiers larges comme ceux des courbes elliptiques pour la cryptographie sont négligeables devant la multiplication et le carré. Une addition est égale à 0,05 multiplication[19, 18]. Par ailleurs, la représentation d'un point de la courbe avec deux coordonnées x et y (système de coordonnées affines) implique des opérations d'inversion (plus coûteuses) dans l'addition et le doublement de points. Pour éviter cette inversion qui est supérieure à 30 multiplications [24, 55], une représentation en coordonnées projectives (jacobiennes) de classe équivalente insère un multiple de deux dans la formule et élimine les dénominateurs dans l'équation de Weierstrass. On remarque dans l'équation 3.2 que le développement génère une division par une constante. Pour l'éviter, la formule peut être utilisée de sa façon suivante [76] :

$$2ab = (a+b)^2 - a^2 - b^2 \tag{3.3}$$

L'usage d'un système de coordonnées le plus approprié suivant l'opération à effectuer peut jouer sur l'efficacité. Toutefois, on doit faire le compromis entre l'efficacité vis-à-vis de l'opération concernée et le coût du passage d'un système de coordonnées au suivant. Des courbes aux propriétés intrinsèques intéressantes comme les courbes d'Edwards dans le système de coordonnées affines sont particulièrement efficaces sur les opérations de doublement, de triplement de point [42].

En fixant certains paramètres de la courbe, comme par exemple *a=-3* [2], cela peut améliorer le coût des calculs surtout dans le cas d'un doublement, d'un triplement etc, mais cela impose une restriction sur le choix du paramètre a.

3.1.2/ Opérations sur le scalaire

Le principe des techniques d'accélération de la multiplication scalaire est de réduire au minimum possible le nombre d'opérations de points (généralement l'addition et le doublement), de jouer sur l'ordre de ces opérations pour une exécution efficace, d'utiliser des opérations plus complexes, ou encore d'unifier ces opérations comme doublement-addition, triplement-addition etc. via des formules.

L'algorithme Double-and-Add, est la première technique traditionnelle basée sur l'expansion binaire du scalaire k. Le scalaire est représenté en binaire, il est ensuite scanné bit par bit, et suivant la valeur du bit scanné, une opération élémentaire simple d'addition ou de doublement de points est effectuée. Les algorithmes Non-Adjacent Form (NAF), windows NAF et slinding windows, sont

aussi des techniques plus efficaces que Double-and-Add basées sur le même principe [50, 87]. D'autres algorithmes comme Double-base chains basés sur une représentation binaire sont encore plus performants que les précédents [37, 31, 80]. Ainsi, des algorithmes basés sur ceux précités permettent de les optimiser [98, 100, 20].

Des solutions développent des opérations plus complexes comme le triplement d'un point, le quadruple d'un point mais aussi l'unification des opérations à travers des formules plus complexes. Dans le système de coordonnées affines, l'addition étant moins coûteuse que le doublement, des techniques basées sur l'ordre des opérations développent des expressions composées comme doublement-addition, triplement-addition dans lesquelles le doublement est remplacé par l'addition[44].
Par exemple, pour calculer *2P+Q* (un doublement suivi d'une addition), on calcule *(P+Q)+P* (deux additions). Cette technique est utilisée dans certaines méthodes par l'échange entre les inversions et les multiplications moins coûteuses, et par une extension à travers des formules quadruple, quadruple-addition etc.[31]. L'inconvénient de ces opérations composées est qu'elles incluent des inversions dans leur formule.

Dans des cas spécifiques, la représentation (appelée coordonnées mixtes) dans une addition de points (addition mixte A_m), d'un des points en cordonnées affines, et l'autre en coordonnées jacobiennes donne des formules plus efficaces [32].
Même si l'addition est généralement plus coûteuse que le doublement dans le système de coordonnées jacobiennes, on constate que le coût peut être réduit via une addition spéciale que nous notons ici par A_z(en coordonnées jacobiennes), c'est à dire avec la même coordonnée z identique pour les deux points [79]. Cette formule s'applique dans le contexte où le calcul de la multplication scalaire démarre par une addition. Ainsi, dans *2P+Q*, cette technique est utilisée en remplaçant dans *(P+Q)+P, (P+Q)* par une addition spéciale avec la coordonnée z identique et la deuxième addition par une addition traditionnelle (A_p)[76]. Cela nécessite la même coordonnée z entre le résultat de la première addition et le point P. La méthode est efficace pour des courbes dont le coefficient a est fixé à *-3* comme c'est recommandé par les standards de clés publiques [2]. On peut encore réduire le coût du doublement-addition en unifiant dans une seule formule les deux additions (l'addition mixte et l'addition spéciale)[76].

Dans le Tableau 3.1, nous comparons quelques résultats connus, par le coût de leurs opérations en prenant comme paramètres, le système de coordonnées, le type et les paramètres des courbes afin de constater leur impact sur quelques opérations de points.

Opérations	*Coordonnées*	*Courbes*	*Coût*	*Coût (a=-3)*
A_p traditionnelle	$\mathcal{A}$	Weierstrass	I+2M+S	-
A_p traditionnelle	$\mathcal{J}$	Weierstrass	12M+4S	-
A_p traditionnelle	$\mathcal{P}$	Weierstrass	12M+2S	-
A_p mixte	$\mathcal{J}+\mathcal{A}$	Weierstrass	8M + 3S	-
A_p	$\mathcal{P}$	Edwards	10M+S+C	-
A_p	$\mathcal{P}$	Hessian	12M	-
A_p	$\mathcal{P}$	Huff	12M	-
D_p traditionnel	$\mathcal{A}$	Weierstrass	I+2M+2S	-
D_p traditionnel	$\mathcal{J}$	Weierstrass	4M + 6S	4M + 4S
D_p traditionnel	$\mathcal{P}$	Weierstrass	7M + 6S	-
D_p traditionnel	$\mathcal{P}$	Edwards	3M+4S	-
D_p traditionnel	$\mathcal{P}$	Hessian	12M	-
D_p traditionnel	$\mathcal{P}$	Huff	6M+5S	-
T_p traditionnel [37]	$\mathcal{J}$	Weierstrass	10M + 6S	10M + 4S
T_p traditionnel [76]	$\mathcal{J}$	Weierstrass	9M + 6S	9M + 5S
A_z [76]	$\mathcal{J}$	Weierstrass	5M+2S	-
A_u	$\mathcal{P}$	Weierstrass	11M+5S+C	-
A_u	$\mathcal{P}$	Edwards	10M+S+C	-
A_u	$\mathcal{P}$	Hessian	12M	-
A_u	$\mathcal{P}$	Huff	11M	-

TABLE 3.1 – Coût de quelques opérations sur les points

3.2/ CONTRIBUTION 3 : ACCÉLÉRATION DE LA MULTIPLICATION SCALAIRE À INTERVALLE SÉLECTIF

Dans cette section, nous présentons un nouveau mécanisme appelé Réduction Scalaire (RS) permettant d'accélérer la multiplication scalaire au niveau de l'arithmétique sur le scalaire. Notre mécanisme est basé sur *l'opposé* et l'*ordre* d'un point afin de réaliser une réduction de la taille en binaire du scalaire dans des intervalles bien déterminés. Soit $\mathbb{F}_p$ un corps fini premier de caractéristique supérieur à 3 ($\mathbb{F}_p$ avec p $\neq$ 2 ou 3). Si $E(\mathbb{F}_p)$ est une courbe elliptique sur un corps fini premier et $\#E(\mathbb{F}_p)$ représente le nombre de points de la courbe $E(\mathbb{F}_p)$, $\#E(\mathbb{F}_p)$ est aussi appelé ordre du groupe de points. Le comptage du nombre de points d'une courbe elliptique est une étape indispensable dans la recherche de courbes cryptographiquement sûres. En 1922, Hasse démontra un théorème sur l'ordre du groupe de points d'une courbe elliptique sur un corps fini premier [55] :

$$| \#E(\mathbb{F}_p) - p - 1 \leq 2\sqrt{p} | \tag{3.4}$$

Si $\mathbb{G}$ est un groupe cyclique de $E(\mathbb{F}_p)$ d'ordre n généré par un point de base P (appelé point générateur), les points de $\mathbb{G}$ sont des multiples de P : $\mathbb{G}=\langle P\rangle=\{\infty, P, 2P, ..., (n\text{-}2)P, (n\text{-}1)P\}\subseteq E(\mathbb{F}_p)$ avec $[n]P=\infty$. L'ordre du point P (noté par $\#P$) est n.

3.2.1/ DESCRIPTION DE LA MÉTHODE DE RÉDUCTION DU SCALAIRE (RS)

Notre idée est de réduire la taille du scalaire dans la multiplication scalaire. Ainsi, pour une multiplication scalaire $[k]P$, nous allons trouver une représentation équivalente d'un point $[d]P$ (k et d deux scalaires et $k > d$) dans l'intervalle $[\lfloor\frac{n}{2}\rfloor+1, n\text{-}1]$, où $\lfloor\frac{n}{2}\rfloor$ représente la fonction partie entière

sur $\frac{n}{2}$. Nous rappelons que le calcul de l'opposé d'un point P à savoir le point $-P$ est quasiment gratuit. Nous l'utilisons pour accélérer les calculs. Etant donné un point $P=(x_p, y_p)$ en système de coordonnées affines, pour calculer la négation du point $[k]P=(x_{kp}, y_{kp})$, on calcule $[k]P=(x_{kp}, y_{kp})$ et on change le signe de la coordonnée y (y_{kp}). Notons que pour tout point P de la courbe elliptique, le point $-P$ est aussi sur la courbe. A partir de l'ordre du point P, nous pouvons remplacer $[k]P$ par une représentation équivalente d'un point $[d]P$ en utilisant l'opposé du point $[d]P$. D'une manière générale, pour un scalaire k (entier) secret, nous obtenons à partir de $[k]P$, des représentations de points $[d]P$ à travers les équations suivantes :

$$\begin{cases} a. & \text{If } k > \text{n} \text{ , } kP = dP \quad \text{où } d = (k - \lfloor \frac{|k|}{n} \rfloor .n) \\ b. & \text{If k} \in]\lfloor \frac{n}{2} \rfloor, \text{n-1}], kP = dP \quad \text{où } d = (k - n) \\ c. & \text{If k} \in]0, \lfloor \frac{n}{2} \rfloor], kP = dP \quad \text{où } d = k \\ d. & \text{If } k{=}n \text{ or 0 or } -n, kP = \infty \\ e. & \text{If k} \in [-(\text{n-1}), -\lfloor \frac{n}{2} \rfloor[, kP = dP \quad \text{où } d = (n + k) \\ f. & \text{If k} \in [-\lfloor \frac{n}{2} \rfloor, 0[, kP = dP \quad \text{où } d = k \\ g. & \text{If k <- n, } k\text{P}{=}dP \quad \text{où } d{=} (k{+}n.\lfloor \frac{|k|}{n} \rfloor) \end{cases} \tag{3.5}$$

Pour le cas spécifique des courbes elliptiques pour la cryptographie, $k \in]0, (\text{n-1})]$, nous obtenons la représentation équivalente $[d]P$ du point $[k]P$ à travers les équations (3.5.b) et (3.5.c) comme suit :

$$\begin{cases} b. & \text{If k} \in]\lfloor \frac{n}{2} \rfloor, \text{n-1}], kP = dP \quad \text{où } d = (k - n) \\ c. & \text{If k} \in]0, \lfloor \frac{n}{2} \rfloor], kP = dP \quad \text{où } d = k \end{cases} \tag{3.6}$$

Afin de mieux décrire notre méthode, nous utilisons l'exemple ci-dessous :
Exemple :
Soit le nombre premier $p = 23$, que nous avons choisi très petit, mais dans la réalité p est largement supérieur à 23. Si nous considérons une courbe elliptique E sur $\mathbb{F}_{23}$ définie par $E(\mathbb{F}_{23})$: $y^2 = x^3+\text{x}+1$, alors $\#\, E(\mathbb{F}_{23}) = 28$, $E(\mathbb{F}_{23})$ est un groupe cyclique. Soit $P(0,1)$ le point générateur. Les points de $E(\mathbb{F}_{23})$ sont les suivants :

P=(0, 1)	$2P$=(6, -4)	$3P$=(3, -10)	$4P$=(-10, -7)
$5P$=(-5, 3)	$6P$=(7, 11)	$7P$=(11, -3)	$8P$=(5, -4)
$9P$=(-4, -5)	$10P$=(12, 4)	$11P$=(1, -7)	$12P$=(-6, 3)
$13P$=(9, -7)	$14P$=(4, 0)	$15P$=(9, 7)	$16P$=(-6, 3)
$17P$=(1, 7)	$18P$=(12, -4)	$19P$=(-4, 5)	$20P$=(5, 4)
$21P$=(11, -3)	$22P$=(7, -11)	$23P$=(-5, -3)	$24P$=(-10, -7)
$25P$=(3, 10)	$26P$=(6, 4)	$27P$=(0, -1)	$28P$=∞

Partant de cet exemple et de la notion de groupe cyclique, nous faisons une représention circulaire des points comme on peut le voir respectivement sur les Figures 3.1 et 3.2 pour le cas général et celui spécial des courbes elliptiques pour la cryptographie. Pour le cas général, on peut voir sur la Figure 3.1 que les points [-34]P, [-6]P, [22]P, [50]P ont les mêmes coordonnées. Pour calculer [-6]P (l'opposé de [6]P), nous calculons le point [6]P puis nous apposons le signe moins sur la coordonnée y. De ce fait calculer [-6]P est équivalent à calculer [6]P et est gratuit. Ainsi, on peut calculer quelques multiplications scalaires de la façon suivante :

Pour calculer [50]P, nous calculons [-6]P en appliquant la formule 3.5-a.

Pour calculer [22]P, nous calculons [-6]P en appliquant la formule 3.5-b.

Pour calculer [-34]P, nous calculons [-6]P en appliquant la formule 3.5-g. Pour le cas spécifique des courbes elliptiques pour la cryptographie, on peut voir sur la Figure 3.2 que le

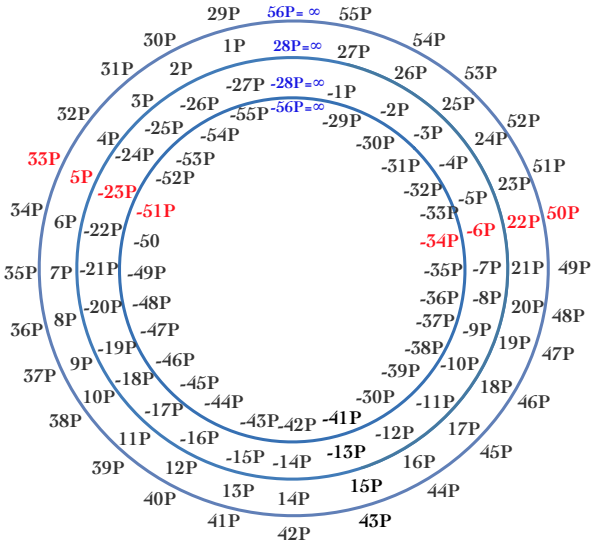


FIGURE 3.1 – Cas général

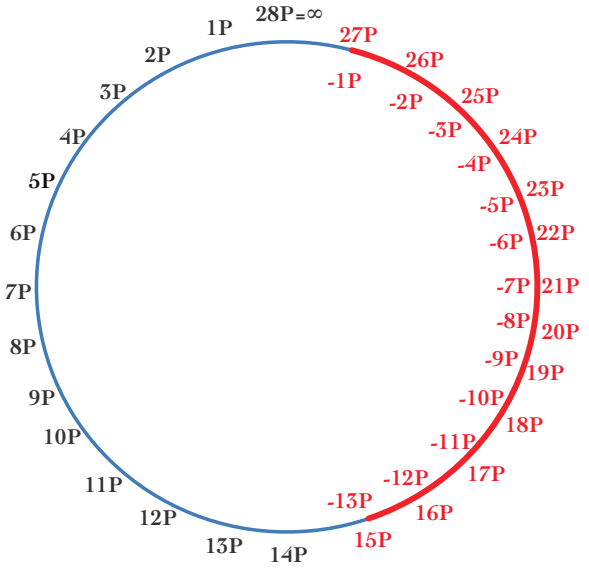


FIGURE 3.2 – Cas particulier des courbes elliptiques pour la cryptographie

calcul les points [15P,16P,.....26P, 27P] peut être respectivement remplacé par celui de [-13P, -12P,.....,-2P, -P]. Et dans ce cas, le calcul du point 27P peut être remplacé par celui du point -P qui est quasiment gratuit.

3.2.2/ ETUDE ANALYTIQUE

Dans cette partie, nous allons analyser les performances de notre solution via des formules mathématiques. Puisque les capteurs sont très limités en termes de ressources, calculer *[k]P* à travers *[d]P* en utilisant la formule 3.6-b dans l'intervalle [$\lfloor \frac{n}{2} \rfloor$+1, n-1] peut contribuer à l'accélération des calculs. Cependant, pour les RCSFs, le choix du scalaire k se fera uniquement

dans cet intervalle. Si nous scannons tous les scalaires, on peut voir intervalle par intervalle la somme de tous les *[k]P* dans l'équation 3.7

$$\sum_{k=1}^{n-1} kP = \sum_{k=1}^{\lfloor \frac{n}{2} \rfloor - 1} kP + \lfloor \frac{n}{2} \rfloor P + \sum_{k=\lfloor \frac{n}{2} \rfloor + 1}^{n-1} kP \tag{3.7}$$

Si on reste dans l'intervalle [$\lfloor \frac{n}{2} \rfloor$+1, n-1] en se basant sur la Figure 3.2 on a :

[15]P = [13]P + 2([1]P);
[16]P = [12]P + 2([2]P);
[17]P = [11]P + 2([3]P);
..... = ...;
..... = ...;
[26]P = [2]P + 2([12]P);
[27]P = [1]P + 2([13]P).

Ainsi :

$$\sum_{k=\lfloor \frac{n}{2} \rfloor + 1}^{n-1} kP = \sum_{k=1}^{\lfloor \frac{n}{2} \rfloor - 1} kP + 2 \sum_{k=1}^{\lfloor \frac{n}{2} \rfloor - 1} kP \tag{3.8}$$

En utilisant notre méthode de réduction du scalaire, on remplace respectivement les points [15]P, [16]P,.....[26]P, [27]P] par [-13]P, [-12]P,.....,[-2]P,[-1]P dans l'intervalle [$\lfloor \frac{n}{2} \rfloor$+1, n-1], ainsi l'expression :

$\sum_{k=\lfloor \frac{n}{2} \rfloor + 1}^{n-1} kP$ peut être remplacée par $\sum_{k=1}^{\lfloor \frac{n}{2} \rfloor - 1} |k|P$, (voir l'équation 3.10).

$$\sum_{k=1}^{n-1} kP = 2 \sum_{k=1}^{\lfloor \frac{n}{2} \rfloor - 1} kP + \lfloor \frac{n}{2} \rfloor P + 2 \sum_{k=1}^{\lfloor \frac{n}{2} \rfloor - 1} kP \tag{3.9}$$

En appliquant notre méthode, l' équation 3.9 peut être remplacée par l'équation 3.10 :

$$\sum_{k=1}^{n-1} kP = \sum_{k=1}^{\lfloor \frac{n}{2} \rfloor - 1} kP + \sum_{k=1}^{\lfloor \frac{n}{2} \rfloor - 1} |k|P + \lfloor \frac{n}{2} \rfloor P. \tag{3.10}$$

Si on scanne tous les scalaires k pour le calcul de *[k]P* dans l'intervalle [$\lfloor \frac{n}{2} \rfloor$+1, n-1], nous pouvons remarquer à travers les équations 3.9 et 3.10, que le gain ou l'accélération pour tous les scalaires est de $\sum_{k=1}^{\lfloor n/2 \rfloor - 1} 2kP$. Mais, il serait encore plus pertinent de calculer cette l'accélération pour un scalaire k donné. Ainsi, en se basant sur la Figure 3.2 on a :
Calculer ([22]P=[16]P+[6]P) est égal à calculer [6]P ; l'accélération est de [16]P= *2(22-(28/2))*.
Calculer ([26]P=[24]P+[2]P) est égal à calculer [2]P ; l'accélération est de [22]P= *2(26-(28/2))*.
Donc, pour un scalaire k donné, l'accélération α est donnée par l'équation suivante :

$$\alpha = 2(k - (\frac{n}{2})) \tag{3.11}$$

La complexité de la multiplication scalaire est déterminée par la configuration et la longueur de la suite binaire représentant le scalaire k. Il serait encore plus pertinant de déterminer l'accélération α en termes de bits. Rappelons que le nombre de bits nécessaire pour représenter un scalaire k donné est $\lfloor log_2(k) \rfloor$+1 ou $log_2(k)$ si k=2^x avec x un entier. En se basant sur l'équation 3.12, l'équation 3.13 permet de calculer l'accélération α en termes de bits.

$$log_2(k - 2(k - \frac{n}{2})) = log_2(k) + log_2(k + \frac{n - 2k}{k}) \tag{3.12}$$

Ainsi, l'accélération α en bits est :

$$\alpha = |log_2(k + \frac{n-2k}{k})| = |log_2(\frac{|d|}{k})|, . \tag{3.13}$$

$$\text{où } log_2(k + \frac{n-2k}{k}) < 0$$

Pour tout ordre n>=1, en scannant tous les scalaires de l'intervalle [$\lfloor \frac{n}{2} \rfloor$+1, n-1], la Figure 3.3 montre le comportement de notre méthode. Pour un scalaire choisi dans l'intervalle [2, $\lfloor \frac{n}{2} \rfloor$] , notre

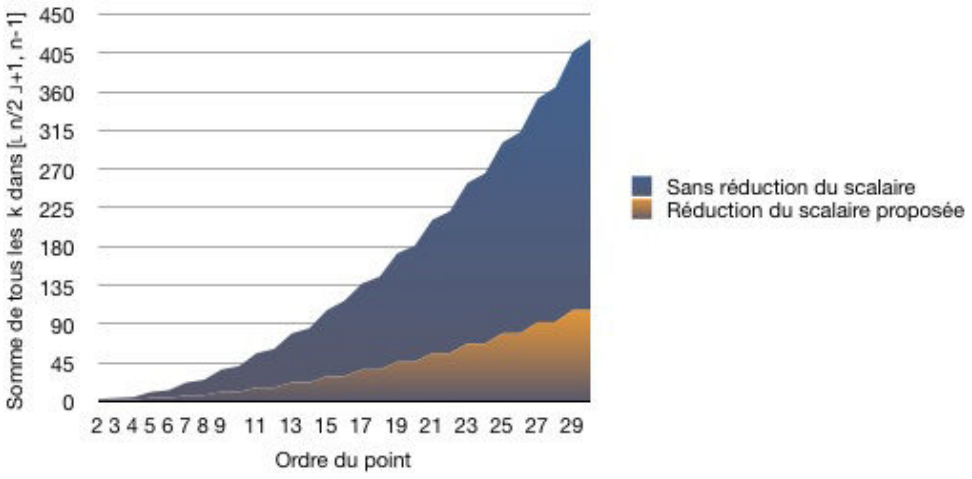


FIGURE 3.3 – Somme de tous les scalaires en fonction de l'ordre n dans [$\lfloor \frac{n}{2} \rfloor$+1, n-1]

méthode n'est pas efficace, elle est faite pour s'appliquer seulement dans l'intervalle [$\lfloor \frac{n}{2} \rfloor$+1, n-1]. Si on considère les scalaires de cet l'intervalle, le mécanisme proposé effectue une réduction significative comme on peut le noter sur la Figure 3.3. Nous calculons la valeur moyenne de l'ensemble des scalaires de cet intervalle où s'applique notre mécanisme de réduction (voir l'équation 3.14).

$$\frac{1}{n-1-\lfloor \frac{n}{2} \rfloor}(\sum_{k=\lfloor \frac{n}{2} \rfloor+1}^{n-1} kP) = \frac{1}{n-\lfloor \frac{n}{2} \rfloor}(\sum_{k=1}^{\lfloor \frac{n}{2} \rfloor-1} kP). \tag{3.14}$$

La Figure 3.4 permet de déterminer pour un ordre de point donné, le k moyen dans l'intervalle [$\lfloor \frac{n}{2} \rfloor$+1, n-1].

On peut encore faire des optimisations basées sur la valeur de l'ordre du point P choisi toujours dans le même intervalle [$\lfloor \frac{n}{2} \rfloor$+1, n-1]. C'est ainsi que nous nous intéressons à la somme de l'ensemble des valeurs du scalaire k selon que l'ordre du point P est pair ou impair :
Si l'ordre $n > 2$ est un nombre pair : $\sum_{k=\lfloor \frac{n}{2} \rfloor+1}^{n-1} kP = 3\sum_{k=1}^{\lfloor \frac{n}{2} \rfloor-1} kP$
Si l'ordre $n \geq 3$ est un nombre impair : $3\sum_{k=1}^{\lfloor \frac{n}{2} \rfloor-1} kP > \sum_{k=\lfloor \frac{n}{2} \rfloor+1}^{n-1} kP \geq 2\sum_{k=1}^{\lfloor \frac{n}{2} \rfloor-1} kP$.

La Figure 3.5 détermine l'accélération sur l'ensemble des scalaires pour un ordre pair et impair. Ainsi, on peut voir que si n est pair, la courbe est constante, c'est une droite d'équation $y=3$. Mais si n est impair, on constate que la droite d'équation y=3 est une asymptote horizontale à la courbe. Par conséquent, si on travaille avec un ordre pair, les calculs sont accélérés trois fois plus. En d'autres termes, on calcule trois fois moins avec notre méthode, alors que pour un ordre impair, l'accélération est > 2, et < 3.

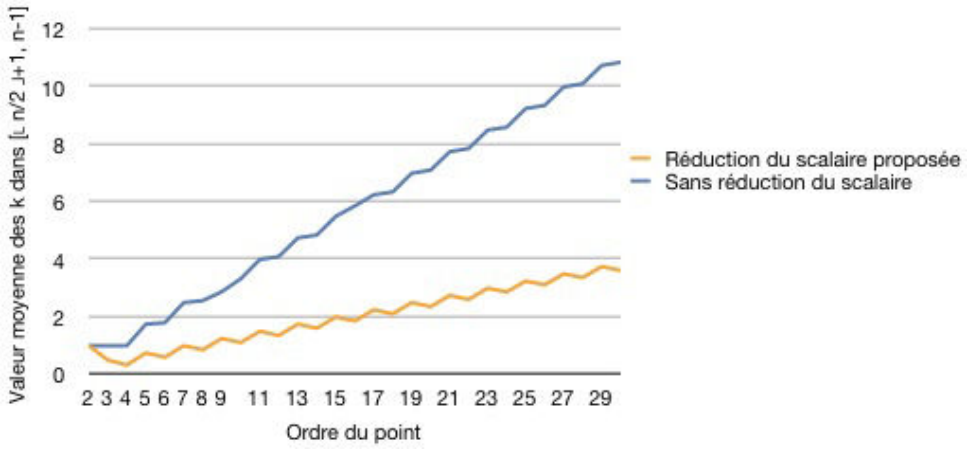


FIGURE 3.4 – Valeur moyenne des scalaires en fonction de l'ordre dans [$\lfloor \frac{n}{2} \rfloor$+1, n-1]

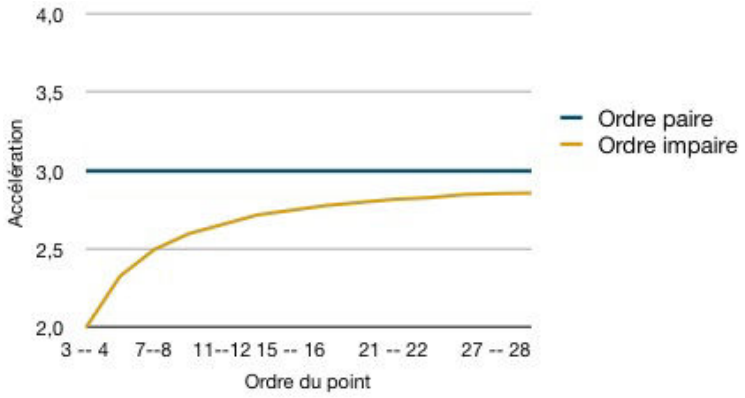


FIGURE 3.5 – Accélération moyenne entre ordre pair et impair

3.2.3/ ANALYSE PLUS PRÉCISE

L'unité de l'accélération peut être exprimée en bits, ou en nombre d'opérations d'addition/doublement. Par exemple, NAF a besoin de $log_2(k)$ doublement(s) et $(log_2(k)/3)$ addition(s). Parfois en utilisant notre méthode, le scalaire est bien réduit alors qu'il n'y a pas de gain (en termes de bits). Par exemple calculer *[15]P* revient à calculer *[13]P* en utilisant notre mécanisme. Dans ce cas il n'y a pas de gain puisqu'il faut 4 bits pour coder les scalaire *15* et *13.* Dans un deuxième exemple, calculer *[16]P* revient à calculer *[12]P* en utilisant notre mécanisme. Dans ce cas, l'accélération est de *1bit* puisqu'il faut *5bits* pour représenter *16* et *4 bits* pour représenter *12*. D'où la nécessité de définir le scalaire de l'intervalle $[\lfloor \frac{n}{2} \rfloor + 1, n-1]$ à partir duquel l'accélération α existe. On remarque que ceci, dépendra de la valeur de l'ordre n. Ainsi, dans l'interval $[\lfloor \frac{n}{2} \rfloor + 1, n-1]$, trois cas sont possibles pour l'accélération α :

- Si $log_2(n) = x$, où x est un entier, notre solution accélère la multiplication scalaire *[k]P* dans l'intervalle $[\lfloor \frac{n}{2} \rfloor + 1, n-1]$ par la réduction du nombre de bits du scalaire k. Comme on peut le noter dans l'exemple, à partir de $16 = 2^4$ (*4* entier)
- Si $log_2(n) = x$, où x n'est pas un entier, notre solution accélère la multiplication scalaire *[k]P* dans l'interval $[2^{\lfloor log_2 \frac{n}{2} \rfloor + 1}, n-1]$
- Si *k= (n-1)*, l'accélération α est maximale, la longueur l en bits du scalaire k est maximale (égale

à $log_2(n-1)$, elle est égale à 1 pour le scalaire d réduisant k. Dans ce cas le calcul est quasiment gratuit.

On a sur les Tableaux 3.2 et 3.3 l'accélération exprimée en bits pour quelques valeurs du scalaire k selon x soit entier ou non.

Valeurs de k	*Accélération α (bits)*
$\lfloor \frac{n}{2} \rfloor + 1$	1
$>= (\lfloor \frac{n}{2} \rfloor + 1)$	$1 < \alpha < log_2(k)$
(n-1)	$log_2(k)$
5360	521

TABLE 3.2 – Accélération α pour quelques valeurs de k pour x entier

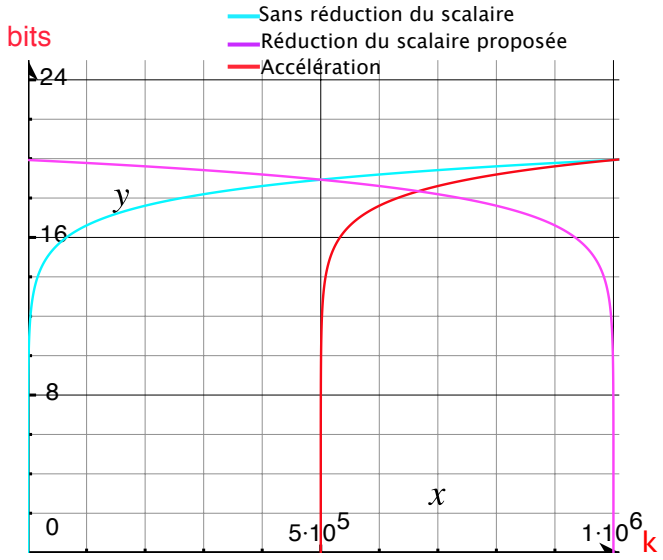
Valeurs de k	*Accélération α (bits)*
$2^{\lfloor log_2 \frac{n}{2} \rfloor + 1}$	1
$>= 2^{\lfloor log_2 \frac{n}{2} \rfloor + 1}$	$1 < \alpha < log_2(k)$
(n-1)	$log_2(k)$

TABLE 3.3 – Accélération α pour quelques valeurs de k pour x non entier

Pour un ordre de point donné (qui est dans ce cas-ci de l'ordre du scalaire k), la Figure 3.6 montre l'évolution en nombre de bits de l'accélération α et de notre mécanisme de réduction sur le scalaire k en fonction de celui-ci exprimé en décimal. Ainsi, on peut remarquer sur cette figure que notre mécanisme n'est intéressant que dans l'intervalle $[\lfloor \frac{n}{2} \rfloor + 1, n-1]$ et son nombre de bits tend vers 1 quand le sacalaire k tend vers sa valeur maximale. Il est important de noter qu'il n'est pas nécessaire de faire une comparaison avec les solutions existantes, le mécanisme proposé peut être quasiment appliqué à celles-ci. Rappelons que pour certains algorithmes d'accélération de la multiplication scalaire, la complexité ou temps d'exécution peut être évalulé en fonction du nombre d'opérations de doublements et d'additions basé sur la longueur binaire l du scalaire comme le montre le Tableau 3.4. Ainsi, la Figure 3.7 montre l'application de notre mécanisme sur les algorithmes comme Double-and-Add (DA), NAF, wNAF ect.. Pour tous ces algorithmes le nombre d'opérations de doublements est égal à l (longueur en bits de k). Ainsi, notre comparaison de la Figure 3.7 est réalisée sur les opérations d'additions seulement.

Méthodes	*Temps d'exécution moyen*
Binaire (DA)	$(l\text{-}1)T_{DBL} + \frac{l-1}{2}T_{ADD}$
Binaire (NAF)	$(l\text{-}1)T_{DBL} + \frac{l-1}{3}T_{ADD}$
Windows	$lT_{DBL} + \frac{l}{w+1}T_{ADD}$

TABLE 3.4 – Temps d'exécution de kP basé sur la représentation binaire de k



”

FIGURE 3.6 – L’évolution en nombre de bits de notre méthode et de l’accélération en fonction du scalaire

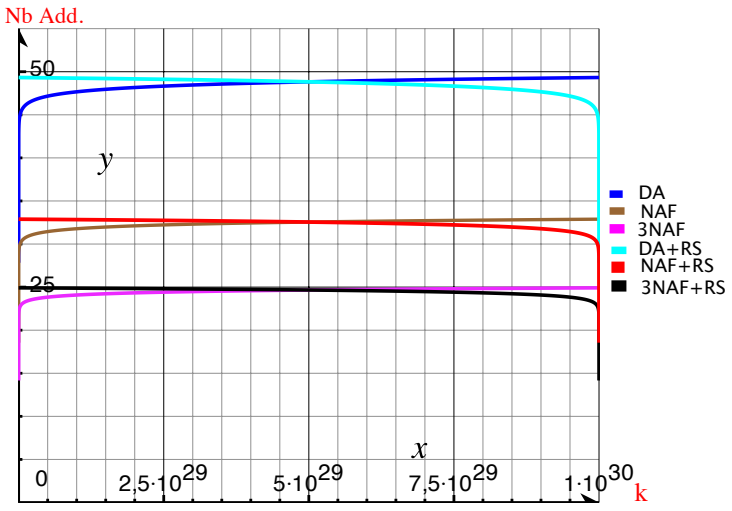


FIGURE 3.7 – Nombre d’opérations d’addition dans le calcul de kP en fonction de k

3.2.4/ EVALUATION DES PERFORMANCES

Pour tester les performances de notre solution, nous avons implémenté un simulateur en Java. Le programme est exécuté sur un processeur Intel Core i5-2520 en prenant en compte la différence entre ce processeur et celui du MSP 430 MCU. Durant les tests, nous avons choisi une courbe elliptique sur $\mathbb{F}_p$ utilisant les paramètres recommandés par un organisme standardisant les courbes elliptiques NIST-192 (National Institute of Standards and Technology) et qui sont donnés dans le Tableau 3.5. p est la taille du corps premier $\mathbb{F}_p$, et a, b sont les coefficients de notre courbe en forme simplifiée de Weierstrass. $P(x_P, y_P)$ sont les coordonnées du point générateur , et son ordre est n.

Paramètres	*Valeurs recommandées par NIST-192*
p	$2^{192} - 2^{64} - 1$
a	-3
b	0x 64210519 e59c80e7 0fa7e9ab 72243049 feb8deec c146b9b1
x_P	0x 188da80e b03090f6 7cbf20eb 43a18800 f4ff0afd 82ff1012
y_P	0x 07192b95 ffc8da78 631011ed 6b24cdd5 73f977a1 1e794811
n	0x ffffffff ffffffff ffffffff 99def836 146bc9b1 b4d22831

TABLE 3.5 – Paramètres des courbes elliptiques recommandés par NIST-192 NIST

Comme indiqué sur l'équation 3.6-b, pour effectuer une multiplication scalaire kP en cryptographie, on doit choisir un scalaire $k < n$ où n est l'ordre du point générateur P, ce qui signifie que $nP = \infty$. En plus, notre méthode doit théoriquement fonctionner dans l'intervalle $k \in]\lfloor \frac{n}{2} \rfloor, n-1]$. Pour prouver cette propriété, nous avons choisi 6 valeurs de 192 bits pour k, distribuées uniformément dans l'intervalle $]0, n-1]$ (voir Table 3.6).

k	*Valeurs en hexadecimal*
$n/6$	0x 2aaaaaaa aaaaaaaa aaaaaaaa 99a5295e 58bca19d 9e2306b2
$n/3$	0x 55555555 55555555 55555555 334a52bc b179433b 3c460d65
$n/2$	0x 7fffffff ffffffff ffffffff ccef7c1b 0a35e4d8 da691418
$2n/3$	0x aaaaaaaa aaaaaaaa aaaaaaaa 6694a579 62f28676 788c1aca
$5n/6$	0x d5555555 55555555 55555555 0039ced7 bbaf2814 16af217a
$n-1$	0x ffffffff ffffffff ffffffff 99def836 146bc9b1 b4d22830

TABLE 3.6 – Valeurs de k choisis pour l'évaluation des performances

Notre solution a été testée avec les systèmes de coordonnées affines et jacobiennes. Les scalaires sont respectivement représentés sous forme binaire et NAF combinés avec notre méthode de Réduction du Scalaire (RS). Les résultats des tests sont donnés dans les Tableaux 3.7 et 3.8, et illustrés graphiquement sur les Figures 3.8 et 3.9.

NAF	RS	DA	Gain	$n/6$	$n/3$	$n/2$	$2n/3$	$5n/6$	$n-1$
		√		6579	6572	7604	6555	6931	7471
√				6282	6326	5317	6239	6698	5114
	√			6578	6573	7600	6416	6600	27
√	√			6279	6325	5320	6100	6556	27
	√	√	$\alpha_{(rs/da)}$				2,12%	4,77%	99,63%
√	√		$\alpha_{(rs/naf)}$					1,46%	99,47%
√	√	√	$\alpha_{(rs-naf/da)}$				6,94%	5,41%	99,63%

TABLE 3.7 – Temps d'exécution (ms) en coordonnées affines

Dans tous les cas, on peut noter que le calcul avec la forme NAF est plus rapide que celui de la forme binaire. Ensuite quand $k \in]0, \frac{n}{2}]$, notre méthode de réduction du scalaire n'est pas appliquée. Cependant, si $k \in]\frac{n}{2}, n-1]$, nous pouvons accélérer les calculs par une réduction du scalaire. Quand la valeur de k est spécialement proche de $n-1$, les calculs sont presque instantanés puisque $(n-1)P = -P$.

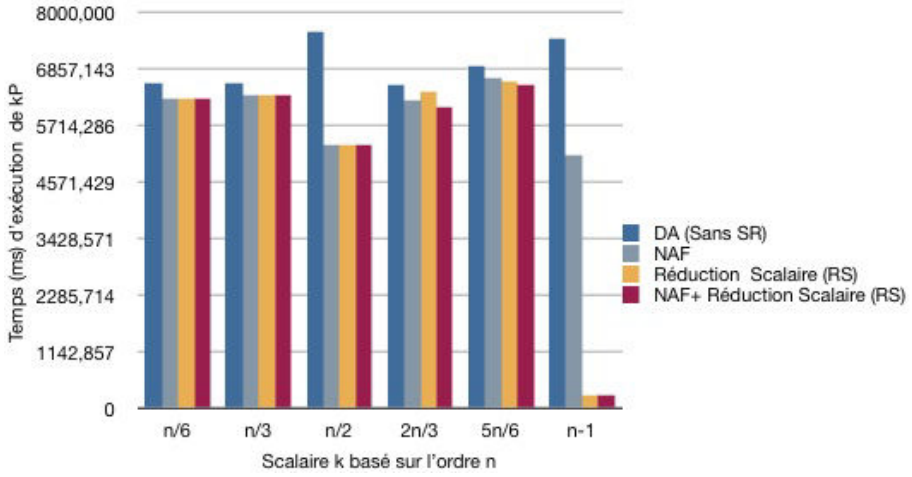


FIGURE 3.8 – Temps d'exécution (ms) en coordonnées affines

NAF	RS	DA	Gain	$n/6$	$n/3$	$n/2$	$2n/3$	$5n/6$	$n-1$
				3066	3102	3621	3072	3202	3520
√				3053	3074	3592	3071	3189	3541
	√			3070	3100	3622	3030	3107	9
√	√			3050	3075	3597	3029	3094	9
	√	√	$\alpha_{(rs/da)}$				1,36%	2,96%	99,74%
√	√		$\alpha_{(rs/naf)}$				1,33%	2,57%	99,74%
√	√	√	$\alpha_{(rs-naf/da)}$				1,39%	3,37%	99,74%

TABLE 3.8 – Temps d'exécution (ms) en coordonnées jacobiennes

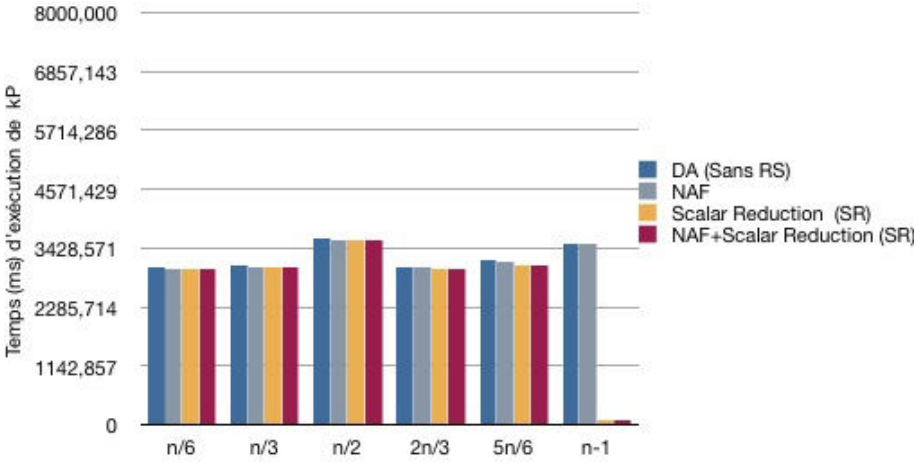


FIGURE 3.9 – Temps d'exécution (ms) en coordonnées jacobiennes

En coordonnées jacobiennes, puisqu'il n y a pas d'inversion modulaire, les calculs sont plus rapides qu'en coordonnées affines. Comme en coordonnées affines, notre réduction du scalaire s'applique pour les scalaire $k \in]\frac{n}{2}, n-1]$, où elle réduit le scalaire et accélère les calculs.

D'après les résultats de l'évaluation des performances, le mécanisme de réduction du scalaire accélère la multiplication scalaire en respectant les paramètres recommandés par le standard NIST-192 des courbes elliptiques pour la cryptographie. Cette accélération dépend fortement de la valeur

du scalaire utilisé mais aussi de l'ordre n du point générateur. Le scalaire k peut être réduit si $k \in]\frac{n}{2}, n-1]$.

Si on définit l'accélération ou gain en termes de temps, de notre solution par rapport à la méthode binaire traditionnelle DA ($\alpha_{rs/da}$), par rapport à NAF (($\alpha_{rs/naf}$), et en combinaison avec NAF par rapport à la méthode binaire ($\alpha_{rs-naf/da}$). Si T_{da}, T_{naf}, T_{rs} et T_{rs-naf} sont respectivement les temps de calcul de la méthode binaire, de NAF, de notre solution RS combinée avec NAF, le pourcentage de gain en termes de temps pour un algorithme x sur l'algorithme y peut être exprimé de la façon suivante :

$$\alpha_{x/y} = \frac{(T_y - T_x) * 100}{T_x} \tag{3.15}$$

Les résultats des calculs du gain sont donnés dans les Tableaux 3.7 et 3.8. On peut voir qu'il n' y a pas de gain dans l'intervalle $]0, \frac{n}{2}]$. Par contre la gain existe pour tous les cas dans l'intervalle $]\frac{n}{2}, n-1]$, il est faible au voisinage de $\frac{n}{2}$ et atteint presque 100 % au voisinage de (n-1). On peut remarquer que le gain de la combinaison de (RS et NAF) sur DA est plus significatif, suivi de celui de RS sur DA, et celui de RS sur NAF est plus faible. Ce qui montre que la combinaison avec d'autres algorithmes garde les performances de ces derniers.

3.3/ Conclusion

Dans ce chapitre, nous avons proposé un nouveau mécanisme d'accélération de la multiplication scalaire sur les courbes elliptiques pour la cryptographie basé sur l'ordre et l'opposé d'un point. D'une part, la méthode proposée réduit le temps de calcul dans l'intervalle [$\lfloor \frac{n}{2} \rfloor$+1, n-1]. D'autre part, nous avons montré que l'usage d'un ordre pair est plus efficace qu'un ordre impair. Ce mécanisme est efficace dans l'environnement des RCSFs. On peut aussi noter, qu'il peut être appliqué facilement à toutes les solutions d'accélération de la multiplication scalaire existantes comme on l'a montré avec NAF. De plus, à travers des analyses et des simulations basées sur des évaluations, nous avons montré que la méthode proposée accélère les calculs de la multiplications scalaire respectant le standard NIST-192 des courbes elliptiques pour la cryptographie. En perspective, nous voulons expérimenter cette technique avec des courbes elliptiques définies dans des corps premiers sur de vrais capteurs. Nous voulons ensuite nous intéresser à son comportement dans d'autres corps finis comme ceux binaires et d'extension etc..

4

Accélération du calcul des points précalculés dans les calculs distribués

Sommaire

Dans les Réseaux de Capteurs sans Fil (RCSFs), le parallélisme des calculs peut être une bonne solution pour répondre aux besoins des applications critiques à forte contrainte temporelle ou pour équilibrer les charges de calcul afin d'avoir une meilleure connectivité et une durée de vie plus longue du réseau. Cependant, ce parallélisme n'est pas gratuit, surtout quand il s'agit du calcul de multiplication scalaire dans les courbes elliptiques puisqu'il introduit des charges supplémentaires liées à la communication, au stockage mémoire et au calcul. Dans ce chapitre, notre objectif est d'accélérer la multiplication scalaire à travers des points résultants du partitionnement du scalaire afin de minimiser les coûts additionnels de calcul et de stockage mémoire durant les calculs parallèles des partitions. La Figure 4.1 montre un exemple d'application de surveillance médicale à domicile de personnes agées vivant seules. Lorsqu'un événement critique (malaise par exemple) survient, une alerte est envoyée via l'Internet à un service d'urgence pour une intervention immédiate. La sécurité des données du patient doit être assurée, et pour cela, le mécanisme utilisé ne doit pas constituer un obstacle pour alerter en temps réel le service d'urgence. Pour ce faire, on peut accélérer les calculs par le biais du parallélisme en les distribuant sur plusieurs capteurs.

4.1/ Calculs parallèles de la multiplication scalaire

Le parallélisme pour l'accélération de la multiplication scalaire peut s'effectuer à travers un ou plusieurs niveaux arithmétiques : sur les formules des opérations telles que l'addition et le doublement, entre les opérations elles mêmes, ou sur le scalaire en le partitionnant. Dans la plupart des travaux actuels, ce nouveau concept de parallélisme de de multiplication scalaire est développé sur des architectures multi-processeurs.

Sur les formules des opérations de points, les calculs sont effectués en parallèle par plusieurs processeurs en tenant compte de leur ordre et de leur interdépendance. Par exemple, on peut voir sur le Tableau 4.1 l'exécution d'un doublement (voir équation 4.1) de point sur une architecture à trois

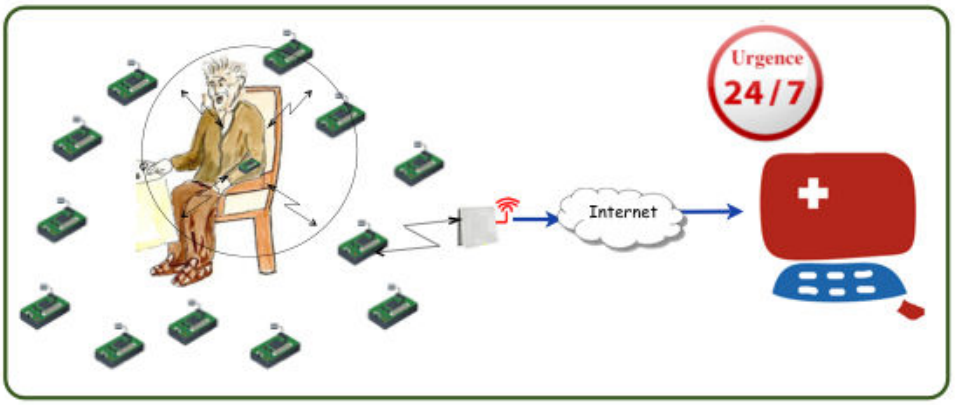


FIGURE 4.1 – Application médicale critique à temps réel

processeurs [76]. Seules les opérations de multiplication et de carré sont représentées, les autres (addition, soutraction, multpllication par une constante etc.) étant négligeables. Les paramètres X_1^2 et Z_1^4 sont pré-calculés dans le système de coordonnées utilisé.

$$\begin{cases} 2(X_1, Y_1, Z_1, X_1^2, Z_1^2, Z_1^4) \\ = (X_3, Y_3, Z_3, X_3^2, Z_3^2, Z_3^3/Z_3^4) \\ X_3 = \alpha^2 - 2\beta \\ Y_3 = \alpha(\beta - X_3) - 8Y_1^4 \\ Z_3 = (Y_1 + Z_1)^2 - Y_1^2 - Z_1^2 \end{cases} \quad \begin{cases} \alpha = 3X_1^2 + aZ_1^4 \\ \beta = 2[(X_1 + Y_1^2)^2 - X_1^2 - Y_1^4] \end{cases} \tag{4.1}$$

Opérations	*Processeur 1*	*Processeur 2*	*Processeur 3*
1. Calcul Carré	α^2	$(Y_1 + Z_1)^2$	Y_1^2
2.Calcul Carré	Y_1^4	Z_3^2	$(X_1 + Y_1^2)^2$
3.Calcul Multiplication	X_3^2	$\alpha(\beta - X_3)$	Z_3^3/Z_3^4

TABLE 4.1 – Calculs parallèles du doublement [76]

Ces calculs parallèles se font en trois étapes. A la première étape, chaque processeur calcule un carré. C'est ainsi que le processeur 1 calcule α^2, le processeur 2 et le processeur 3 calculent respectivement $(Y_1+Z_1)^2$ et Y_1^2 en parallèle avec le premier. On peut noter que dans une étape, chaque processeur utilise des paramètres indépendants des autres. A la deuxième étape, chaque processeur effectue encore une opération de carré en se servant des paramètres calculés précédemment. Le processeur 2 calcule par exemple Z_3^2 qui est composé de $(Y_1+Z_1)^2$ et Y_1^2 calculés précédemment et de Z_1^2 donné. A la troisième étape, le processeur 1 calcule un carré, le processeur 2 calcule une multiplication, et le processeur 3 calcule une multiplication (Z_3^3) si l'opération suivante est une addition de points, ou un carré (Z_3^4) si l'opération suivante est un doublement de point. L'intérêt de faire les opérations de même nature (carré par exemple) en parallèle est de permettre aux processeur d'être en phase.

Entre les opérations de points, des techniques permettent de paralléliser les séries d'opérations d'addition et de doublement sur une architecture à deux processeurs avec une mémoire partagée dont l'un des processeurs exécute les doublements et l'autre les additions [5]. Le premier processeur lit initialement le point P, et commence à scanner les bits k_i puis effectue des doublements

de point. Il écrit 2^iP dans le buffer quand le bit k_i rencontré est différent de zéro. Le deuxième processeur lit 2^iP du buffer et effectue une addition de points. Les calculs sont terminés quand il n' y a plus de 2^iP dans le buffer.

Pour d'autres techniques de parallélisme, le principe consiste à partitionner le scalaire k (représenté sur l bits) en m blocs de taille fixe sur des architectures SIMD [1]. Ce partitionnement génère des points à calculer appelés points *pré-calculés* qui nécessitent d'être calculés ou d'être stockés à priori avant le démarrage des calculs parallèles.

Des travaux récents [95], inspirés de [74] utilisent la technique du partitionnement du scalaire en m blocs de longueur v bits dans les réseaux de capteurs sans fils. Le scalaire représenté sur l bits est divisé en m blocs B_i de longueur $v = \frac{l}{m}$ chacun selon m capteurs choisis pour participer aux calculs.

$$kP = B_0 2^{0v}P + B_1 2^{1v}P + B_2 2^{2v}P + ... + B_{m-1}2^{(m-1)v}P \tag{4.2}$$

$$\text{Où } B_i = \sum_{j=iv}^{iv+v-1} l_j 2^j P \text{ avec } l_j \text{ le bit au rang j de la suite binaire } l$$

Ce partitionnement génère des points précalculés $P_i = 2^{ib}P$. Par exemple considérons un scalaire k de *160* bits et un point P, on veut calculer kP sur 4 capteurs. Le scalaire k est découpé en quatre blocs de longueur de quarante bits chacun.

$$kP = \underbrace{(B_{0\cdot0}B_{0\cdot1}....B_{0\cdot39})_2.2^0}_{blocB_0} P + \underbrace{(B_{1\cdot40}B_{1\cdot41}.....B_{1\cdot79})_2.2^{40}}_{blocB_2} P + v\underbrace{(B_{2\cdot80}B_{2\cdot81}.....B_{2\cdot119})_2.2^{80}}_{blocB_2} P + \underbrace{(B_{3\cdot120}B_{3\cdot121}.....B_{3\cdot159})_2.2^{120}}_{blocB_3} P$$

Les points précalculés sont : $2^{40}P$, $2^{80}P$ et $2^{120}P$.

On peut noter que toutes les techniques de parallélisme basées sur le partitionnement du scalaire génèrent des points pré-calculés, qui devront à priori être calculés et stockés, ce qui entraine une consommation supplémentraire de mémoire et d'énergie. C'est ainsi qu'on peut noter que lorsque les caluls doivent s'effectuer en parallèle dans un système réparti ou distribué, les solutions existantes basées sur le partitionnement du scalaire génèrent des points à calculer et à stocker à priori. Pour éviter ce stockage ou réduire sa durée, nous proposons dans ce chapitre d'accélérer le calcul de ces points. L'objectif étant de les stocker le moins longtemps possible ou à la limite d'éviter leur stockage. La technique que nous utilisons est basée sur la configuration binaire des points pré-calculés et est déduite de l'algorithme Double-and-Add (DA). Elle implique deux niveaux arithmétiques en coordonnées jacobiennes : le niveau point et le niveau scalaire.

4.2/ Contribution 4 : Accélération du calcul des points précalculés

4.2.1/ Partitionnement du scalaire et génération des points précalculés

Comme on peut le voir dans la section précédente, pour effectuer des calculs parallèles sur kP entre m nœuds capteurs, on divise la longueur l du scalaire k exprimé en binaire en m blocs de longueur $v = l/m$ bits et que chaque bloc est exécuté par un capteur. Afin d'utiliser les algorithmes existants (Double-and-Add, NAF, etc.), nous devons aussi les partitionner en m blocs et chaque bloc m_i de l'algorithme doit opérer sur un bloc m_i du scalaire. L'algorithme 29 et l'algorithme 30 montrent respectivement le bloc i pour Double-and-Add et NAF.

Algorithm 29 Double-And-Add pour le nœud *i*

input : d=$(d_{v-1},.....,d_1,d_0)_2$, P $\in E(\mathbb{F}_p)$
output : dP

begin
  $Q \longleftarrow \infty$ **for** $j \leftarrow 0$ **to** $v-1$ **do**
    // début du balayage bit par bit de droite à gauche
    if d_j=*1* **then**
      $Q \longleftarrow Q + 2^{vi}P$ // 2^{vi} est le point pré-calculé
    $P \longleftarrow 2P$
  Return(Q)

Algorithm 30 NAF pour le nœud *i*

input : NAF(d)=$(d_{v-1},.....,d_1,d_0)_2$, P $\in E(\mathbb{F}_p)$
output : dP

begin
  $Q \longleftarrow \infty$ **for** $j \longleftarrow 0$ **to** $v-1$ **do**
    // début du balayage bit par bit de droite à gauche
    $P \longleftarrow 2Q$
    if d_j=*1* **then**
      $Q \longleftarrow Q + 2^{vi}P$ // $2^{vi}P$ est le point pré-calculé
    if d_j=*-1* **then**
      $Q \longleftarrow Q - 2^{vi}P$ // $2^{vi}P$ est le point pré-calculé
  Return(Q)

4.2.2/ Fiabilité du partitionnement du scalaire

Après un partitionnement du scalaire k en m blocs de longueur v, le nœud qui distribue les calculs copie un des blocs dans sa mémoire et distribue les (*m-1*) blocs restants aux autres nœuds. Pour montrer la fiabilité du partitionnement suivi d'une distribution des tâches, nous allons prendre l'exemple de la méthode naive où le nœud distributeur garde le premier bloc B_0 et distribue les autres blocs dans l'ordre. Dans ce cas, l'intrus considère que le distributeur garde le bloc B_0 et le premier bloc reçu comme B_1, le deuxième comme B_2 ect., sachant qu'en réalité pour plus de sécurité, on gardera un bloc différent de B_0 et la distribution se fera de façon aléatoire. Ainsi, pour un intrus, aprés avoir capté les (*m-1*) blocs des m blocs, il peut retrouver difficilement la valeur du scalaire k en faisant des essais exhaustifs. Du fait qu'il lui manque le bloc B_0 de v bits et qu'il n'aura pas à trouver l'ordre des blocs, il lui faudra essayer 2^v possibilités au cas où il aurait déjà récupéré les autres blocs restants. Pour ce cas, on peut dire en bref, que le scalaire utilisé est connu à v bits près.

Par ailleurs, si le choix du bloc du nœud distributeur et l'envoi en clair des autres blocs sont faits au hasard de façon aléatoire, il sera encore plus difficile pour l'intrus de trouver le scalaire k. Non seulement il lui manque un bloc, mais il doit aussi trouver l'ordre des blocs restants. Alors il lui faudra essayer (m $!2^v$) possibilités au cas où il aurait déjà récupéré les autres blocs restants. Dans tous les cas, il faut noter que l'envoi des blocs en clair ne compromet pas totalement la sécurité du scalaire

Une autre possibilité serait d'envoyer les (*m-1*) blocs en mode sécurisé via un chiffrement symétrique qui demande moins de ressources.

Si l'intrus reçoit après calcul, les *(m-1)* résultats venant des capteurs, la sécurité du scalaire k n'est pas compromise. Il est même moins difficile de déterminer k à partir de kP que des $(m-1)$ résultats renvoyés. Il faudra pour le résultat de calcul de chaque bloc d_iP obtenu, trouver d_i à partir de d_iP. Si l'intrus fait des essais exhaustifs, il sera obligé, pour chaque bloc de faire (2^v) possibilités le calcul d_iP, pour trouver d_i. Les v bits du bloc du distributeur n'étant pas connus, il y aura encore (2^v) possibilités pour déterminer le scalaire d_i associé. Si les blocs restants sont reçus dans le désordre, il faudra $m.m!(2^v)$ possibilités

4.2.3/ Méthodes d'accélération du calcul des points précalculés

Dans le cas du calcul parallèle de la multiplication scalaire après partitionnement du scalaire k en blocs de longueur v bits, les points pré-calculés sont de la forme $d_iP=2^{vi}P$ pour le bloc i, où $d_i=2^{vi}$. L'entier 2^{vi} est représenté par $(vi+1)$ bits, une suite binaire dans laquelle seul le bit de poids fort est à 1, les autres sont tous à 0. En coordonnées Jacobiennes, le doublement est moins coûteux que l'addition. Rappelons que le coût du doublement est de $8M+3S$ et celui de l'addition est $12M+4S$. Dans les algorithmes comme Double-and-Add basés sur le doublement et l'addition puisque le ratio des bits à 1 est presque nul dans les points pré-calculés. Nous pouvons remplacer l'addition par le doublement et ainsi obtenir deux doublements. Le concept du calcul direct de plusieurs doublements successifs a été pour une première fois suggéré pour les courbes elliptiques définis sur $\mathbb{F}_{2^n}$ en coordonnées affines [52]. Ce concept permet d'accélérer les doublements répétés en calculant directement 2^xP (avec x entier) et P $\in E(\mathbb{F}_p)$ sans calculer les points intermédiaires $2P, 2^2P, \ldots, 2^{x-1}P$ [82]. Cependant, ces méthodes sont limitées et fonctionnent seulement avec de petits entiers x (2, 3 ou 4). Néanmoins, il existe des formules efficaces pour calculer directement des doublements pour 2^xP avec $P \in E(\mathbb{F}_p)$ pour tout $x >= 1$ [90]. Ainsi, les formules pour le doublement de point, le quadruple de point, et le calcul 2^xP ou 2^x-*uple* sans doublements répétés en coordonnées jacobiennes sont définies à travers les équations : 4.3, 4.4 et 4.5.

$$\begin{cases} Entree \;\; P_1(X_1, Y_1, Z_1) \\ Sortie \; P_2(X_2, Y_2, Z_2) = 2P_1 \end{cases} \quad \begin{cases} \alpha_1 = 3X_1^2 + aZ_1^4 \\ \beta_1 = 4X_1Y_1^2 \\ \gamma_1 = 8Y_1^4 \\ X_2 = \alpha^2 - 2\beta \\ Y_2 = \alpha(\beta - X_2) - \gamma \\ Z_2 = 2Y_1Z_1 \end{cases} \quad Doublement \qquad (4.3)$$

$$\begin{cases} Entree \;\; P_1(X_1, Y_1, Z_1) \\ Sortie \; P_4(X_4, Y_4, Z_4) = 4P_1 \end{cases} \quad \begin{cases} \alpha = 3X_1^2 + aZ_1^4 \\ \beta = \alpha^2 - 8X_1Y_1^2 \\ \gamma = -8Y_1^4 + \alpha(12X_1Y_1^2 - \alpha^2) \\ \omega = 16aY_1^4Z_1^4 + 3\beta^2 \\ X_4 = -8\beta\gamma^2 + \omega^2 \\ Y_4 = -8\gamma^4 + \omega(12\beta\gamma^2 - \omega^2) \\ Z_4 = 4Y_1Z_1\gamma \end{cases} \quad Quadruple \qquad (4.4)$$

$$\begin{cases} Entree \;\; P_1(X_1, Y_1, Z_1) \\ Sortie \; P_{2^x}(X_{2^x}, Y_{2^x}, Z_{2^x}) = 2^x P_1 \end{cases} \quad \begin{cases} \alpha_1 = X_1 \\ \beta_1 = 3X_1^2 + a \\ \gamma_1 = -Y_1 \\ Pour\ i\ allant\ de\ 2\ a\ x,\ calculer\ \alpha_i,\ \beta_i,\ \gamma_i \\ \alpha_i = \beta_{i-1}^2 - 8\alpha_{i-1}\gamma_{i-1}^2 \\ \beta_i = 3\alpha_i^2 + 16^{i-1}a(\prod_{j=1}^{i-1}\gamma_j)^4 \\ \gamma_i = -8\gamma_{i-1}^4 - \beta_{i-1}(\alpha_i - 4\alpha_{i-1}\gamma_{i-1}^2) \\ \omega_x = 12\alpha_x\gamma_x^2 - \beta_x^2 \\ X_{2^x} = \beta_x^2 - 8\alpha_x\gamma_x^2 \\ Y_{2^x} = 8\gamma_x^4 - \beta_x\omega_x \\ Z_{2^x} = 2x\prod_{i=1}^{x}\gamma_i \end{cases} \quad 2^x\text{-}uple \tag{4.5}$$

En s'inspirant de l'algorithme traditionnel Double-and-Add, nous proposons un premier algorithme Double-and-Double dans lequel nous remplaçons l'addition par un doublement. A partir des $(vi+1)$ bits représentant le scalaire $k = 2^{vi}$, on peut scanner bit par bit de la gauche vers la droite les vi bits, et pour chaque bit scanné, on effectue une opération de doublement de point. On peut encore accélérer les calculs en utilisant des formules de calcul directe à la place des opérations répétées sur un point. Par exemple, utiliser la formule de *Quadruple* de point à la place de deux doublements répétés. Dans ce cas, le pas utililisé dans l'algorithme de balayage dépendra de la formule utilisée. Ainsi, dans l'exemple de *Quadruple*, le pas sera de deux. On peut généraliser cette technique à travers la formule 2^x*-uple* d'un point où x est un entier, dans ce cas le pas sera de x dans l'algorithme utilisé pour le balayage. Les algorithmes 31, 32, 33 représentent respectivement nos propositions pour *Double-and-Double*, *Quadruple-and-Quadruple* et du cas géréral 2^x-uple-*and*-2^x-uple.

Algorithm 31 Double-And-Double pour le nœud i

input : d=2^v=(d_v,d_{v-1},.....,d_1,d_0)$_2$, P $\in E(\mathbb{F}_p)$
output : dP

begin
  $Q \longleftarrow P$
  for $j \leftarrow v-1$ **to** 0 **do**
    // Début du balayage (bit par bit) à la (v-1)ème bit
    $Q \longleftarrow 2Q$ // Doublement d'un point
  Return(Q)

4.2.4/ Comparaison et évaluation des performances

En termes d'opérations de doublements, le coût total pour le calcul de w doublements consécutifs à travers la formule de l'équation 4.3 est $4wM+2(2w+1)S$ [77]. Si T_{ADD} et T_{DBL} donnent respectivement les temps de calculs d'une opération d'addition et de doublement, alors pour un scalaire k de longueur l bits, le Tableau 4.2 présente le coût moyen de la méthode Double-and-Double et de quelques méthodes classiques similaires.

Algorithm 32 Quadruple-and-Quadruple pour le nœud i

input : d=2^v=(d_v,d_{v-1},.....,d_1,d_0)$_2$, P $\in E(\mathbb{F}_p)$
output : dP

```
begin
    Q ⟵ P
    j ⟵ v
    repeat
        j ⟵ j − 2    // Début du balayage (2 bits par 2bits) à la (v-1)ème bit
        if j<0 then
            Q ⟵ 2Q                                   // Doublement d'un point
        else
            Q ⟵ 4Q                                   // Quadruple d'un point
    until j <=0
    Return(Q)
```

Algorithm 33 2^x-uple-and-2^x-uple pour le nœud i

input : d=2^v=(d_v,d_{v-1},.....,d_1,d_0)$_2$, P $\in E(\mathbb{F}_p)$
output : dP

```
begin
    Q ⟵ P
    j ⟵ v          // Début du balayage (x bit par x bit) à la (v-1)ème bit
    repeat
        if j<x then
            Q ⟵ 2^j Q                                // 2^j-uple d'un point
        else
            Q ⟵ 2^x Q                                // 2^x-uple d'un point
        j ⟵ j − x
    until j <=0
    Return(Q)
```

Methodes	*Temps d'exécution moyen*
Binaire [22]	$(l\text{-}1)T_{DBL} + \frac{-1}{2}T_{ADD}$
Binaire NAF[50]	$(l\text{-}1)T_{DBL} + \frac{l-1}{3}T_{ADD}$
Windows [50]	$lT_{DBL} + \frac{l}{w+1}T_{ADD}$
Double-and-Double	$(l\text{-}1)T_{DBL}$

TABLE 4.2 – Temps moyen d'exécution de la multiplication scalaire kP

Théoriquement, l'efficacité d'une formule en coordonnées jacobiennes peut être déterminée en fonction du nombre d'opérations de multiplication *(M)* et de carré *(S)* qui la composent. Rappelons que l'addition, la soustraction et la multiplication par une constante sont négligeables devant le

carré et la multiplication. Le Tableau 4.3 montre le coût de chaque formule. Généralement il est supposé, qu'un carré est l'équivalent de *0,6* à *0,8* multiplication [24, 51, 49]. Ainsi, pour un scalaire d'une taille de l bits, nous pouvons déterminer le ratio ($S/M = r$) à partir duquel chaque formule justifie une meilleure efficacité comparée aux autres. Comme on peut le voir sur la Figure 4.2, la formule de Quadruple est plus efficace à partir d'un ratio r >= 0,5. Celle de 2^x-uple est plus performant que celle de doublement à partir d'un ratio r = 9.5 à peu près.

Formules	*Nombre d'opérations*
Doublement	4M + 4S
Quadruple	9M + 10S
2^x-uple(x>=2)	6M + 8S +(x-2)(5M + 5S)

TABLE 4.3 – Coût en termes d'opérations

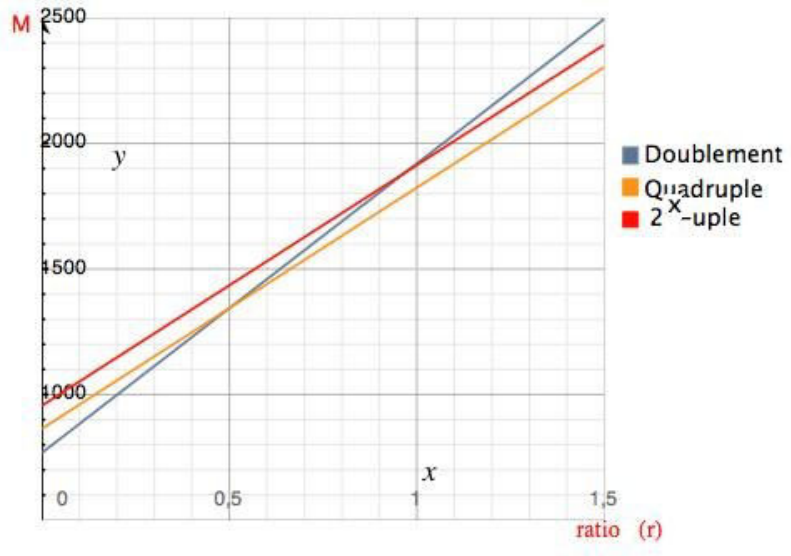


FIGURE 4.2 – Nombre de multiplications en fonction du ratio r = S/M

Pour tester les performances de chaque formule en termes de temps d'exécution, nous avons choisi une courbe elliptique sur un corps premier $\mathbb{F}_p$ et nous avons implémenté quelques formules avec un simulateur java en utilisant des paramètres recommandés par NIST- 192, dans le système de coordonnées jacobiennes. Nous avons utilisé des scalaires de longueur de 192 bits. Le programme est exécuté sur une machine à plusieurs processeurs avec la prise en compte de la différence entre chaque processeur et celui du MSP 430 MCU qui est utilisé dans les capteurs. La première étape consiste à exécuter les programmes avec un seul processeur. Après, nous avons utilisé deux processeurs, ensuite trois processeurs et enfin quatre processeurs. Les résultats des tests sont donnés dans le Tableau 4.4.

Methodes	*1 Processeur*	*2 Processeurs*	*3 Processeurs*	*4 Processeurs*
Doublement	3508	1690	1111	841
2^3-uple	4032	2016	1344	1008
Quadruple	3141	1504	995	744
2^4-uple	3744	1872	1248	936
2^x-uple	17435	4779	2400	1506

TABLE 4.4 – Temps d'exécution (ms) en coordonnées jacobiennes

L'efficacité théorique d'une formule associée à notre algorithme de balayage semble dépendre des opérations arithmétiques élémentaires (l'addition , la soustraction, la multiplication et l'inversion sur les éléments du corps fini) qui la composent, et du pas de balayage. Plus le pas est grand, plus notre algorithme de balayage est rapide et efficace. L'objectif pour un scalaire d_i=2^{vi} est d'avoir une formule efficace dont le pas est supérieur ou égal à *vi*. Ainsi, toute la difficulté est de trouver une formule efficace avec un grand pas.

4.3/ CONCLUSION

Nous avons proposé une méthode permettant d'accélérer les calculs des points pré-calculés afin d'éviter de les stocker ou à défaut, de les stocker le moins longtemps possible. En se basant sur la configuration binaire de ces points pré-calculés nous nous sommes inspirés de l'algorithme Double-and-Add (ou Doublement-et-Addition) pour mettre en place des algorithmes distribués afin de balayer tous les bits à zéro (sauf celui de poids fort) du scalaire pour une exécution parallèle des points pré-calculés. Après une étude comparative en nombre d'opérations de quelques formules existantes relatives à des opérations *(2^{vi}P, avec vi un entier)* sur un point P, nous avons ensuite déterminé leur efficacité à partir du ratio S/M . Dans la pratique, nous avons montré que la formule *Quadruple* (avec un pas de balayage égal à 2) présente le meilleur temps d'exécution alors que théoriquement celle de 2^x-uple (avec un grand pas) semble être plus prometteuse. Ainsi, dans nos travaux futurs nous allons chercher des formules à grands pas plus efficaces.

Conclusion générale et Perspectives

Dans cette thèse, notre objectif a été de trouver des mécanismes et solutions de sécurité peu gourmands en énergie pour les réseaux de capteurs sans fil. Nous avons d'abord mis en avant les contraintes des capteurs et les défis de sécurité à relever dans ces réseaux. Nous nous sommes intéressés à l'étude de diffrentes solutions de contrôle d'accès afin de mettre en évidence leurs objectifs, dispositifs, complexité, limites. Cependant, nous avons pu voir que ces solutions s'appuient en général sur un mécanisme de gestion de clés et dont la phase de pré-distribution des clés comporte un secret à priori partagé. Par conséquent, le mécanisme de contrôle d'accès hérite toutes les vulnérabilités du mécanisme de gestion de clés sur lequel il est basé.

En perspective, on peut envisager, des solutions de contrôle d'accès intégrant la gestion de clés dans un seul et unique protocole.

Dans notre première contribution, nous avons proposé d'améliorer la sécurité du protocole de Vaidya et al. [101] par la protection contre le DdS et la falsification tout en optimisant sa consommation énergétique. C'est ainsi que nous avons proposé une nouvelle solution qui garde tous les avantages du protocole initial. Les résultats finaux obtenus, aussi bien sur simulateur qu'avec des expérimentations réelles sur la plate forme Senslab ont montré que la solution proposée est plus économe en énergie.

La deuxième contribution reprend la première solution de sécurité proposée et introduit le concept de probabilité de risque lié à la vulnérabilité du milieu de déploiement. Ainsi, par simulation, nous avons déterminé pour une architecture physique de RCSF dans un milieu de déploiement donné, le degré de risque ou de vulnérabilité à partir duquel la solution proposée est énergétiquement meilleure. Nous avons ensuite prouvé sa sécurité par vérification automatique de quelques propriétés de base de sécurité à l'aide de l'outil de validation AVISPA.

Nous pensons dans le futur pouvoir faire une analyse plus pertinente des solutions avec des automates afin de faire intervenir d'autres paramètres indicatifs aussi importants que la consommation énergétique, comme par exemple le temps, le stockage mémoire, etc. toujours en rapport avec les ressources limitées des capteurs.

Dans la troisième contribution, nous avons proposé un nouveau mécanisme d'accélération de la multiplicaion scalaire sur les courbes elliptiques pour la cryptographie basé sur l'ordre et l'opposé d'un point. Nous avons ainsi montré à travers des analyses et des simulations basées sur des évaluations que cette méthode réduit le temps de calcul et peut être appliquée facilement à toutes les solutions d'accélération de la multiplication scalaire existantes.

En perspective, nous voulons expérimenter cette technique avec des courbes elliptiques définies dans des corps premiers sur de vrais capteurs. Nous voulons ensuite nous intéresser à son comportement dans d'autres corps finis comme ceux binaires et d'extension etc..

Dans notre dernière contribution, nous avons mis en place une méthode permettant d'accélérer les calculs des points pré-calculés afin d'éviter de les stocker ou à défaut, de les stocker le moins longtemps possible. A partir de la configuration binaire de ces points pré-calculés nous nous sommes inspirés de l'algorithme Double-and-Add pour mettre en place des algorithmes distribués et scan-

ner la suite des *(l-1)bits* sur les *l* bits du scalaire. Enfin nous avons déterminé leur efficacité à partir du ratio S/M. Dans la pratique, nous avons montré que la formule *Quadruple* (avec un pas de balayage égal à 2) présente le meilleur temps d'exécution alors que théoriquement celle de 2^x-uple (avec un grand pas) semble être plus prometteuse.

Ainsi, dans nos travaux futurs nous allons chercher des formules à grands pas plus efficaces que nous souhaiterions évaluer sur la plateforme de capteurs à grande échelle Senslab.

A

Annexe : Coût des algorithmes : (Doublement-et-Addition et NAF)

Algorithm 34 Doublement-et-Addition bit fort/ bit faible

input : $k=(k_{l-1},.....,k_1,k_0)_2$, P $\in E(\mathbb{F}_p)$
output : $Q=[k]P$

begin
  $Q \longleftarrow P$;
  for $i \leftarrow$ *l-2* **to** *0* **do**
    // début du scanne bit par bit (à partir du deuxième bit de de poids fort)
    $Q \longleftarrow 2Q$ // On effectue un doublement
    if k_i=*1* **then**
      $Q \longleftarrow Q + P$ // On effectue une addition
  return (Q)
  *// **Au total, on effectue: (l-1) doublements et (h-1) additions (h étant le nombre de bits à 1)***

Algorithm 35 Doublement-et-Addition bit fort/ bit faible

input : $k=(k_{l-1},.....,k_1,k_0)_2$, P $\in E(\mathbb{F}_p)$
output : $Q=[k]P$

begin
  $Q \longleftarrow P; R \longleftarrow \infty$;
  for $i \leftarrow$ *0* **to** *l-1* **do**
    // début du scanne bit par bit (à partir du bit de de poids faible)
    if k_i=*1* **then**
      $R \longleftarrow R + Q$ // On effectue une addition
    $Q \longleftarrow 2Q$ // On effectue un doublement
  return (Q)
  *// **Au total, on effectue: (l) doublements et (h) additions (h étant le nombre de bits à 1) dont une addition gratuite (addition d'un point avec le point à l'infini***

Algorithm 36 Doublement-et-Addition bit faible/bit fort

input : $k=(k_{l-1},.....,k_1,k_0)_2$, P $\in E(\mathbb{F}_p)$
output : $Q=[k]P$

begin
  $Q \longleftarrow \infty$;
  for $i \leftarrow 0$ **to** *l-1* **do**
    // début du scanne bit par bit (du bit de poids faible vers le bit de poids fort)
    if $k_i=1$ **then**
      $Q \longleftarrow Q + P$ // On effectue une addition
    $P \longleftarrow 2P$ // On effectue un doublement
  return (Q) *// Au total, on effectue: (l) doublements et (h) additions (h étant le nombre de bits à 1) dont une addition gratuite (addition d'un point avec le point à l'infini*

Algorithm 37 Doublement-et-Addition bit fort/ bit faible

input : $k=(k_{l-1},.....,k_1,k_0)_2$, P $\in E(\mathbb{F}_p)$
output : $Q=[k]P$

begin
  $Q \longleftarrow \infty$;
  for $i \leftarrow l\text{-}1$ **to** *0* **do**
    // début du scanne bit par bit de droite à gauche
    $Q \longleftarrow 2Q$ // On effectue un doublement
    if $k_i=1$ **then**
      $Q \longleftarrow Q + P$ // On effectue une addition
  return (Q) *// Au total, on effectue: (l) doublements dont un gratuite (doublement du point à l'infini), et (h) additions (h étant le nombre de bits à 1)*

Algorithm 38 Méthode binaire NAF (1)

input : $NAF(k), P \in E(\mathbb{F}_p)$
output : $Q = [k]P$

begin
  $Q \longleftarrow P$;
  for $i \leftarrow l\text{-}2$ **to** 0 **do**
    // début du scanne chiffre par chiffre (à partir du deuxième chiffre de de poids fort vers celui de poids faible)
    $Q \longleftarrow 2Q$ // On effectue un doublement
    if $(k_i = 1)$ **then**
      $Q \longleftarrow Q + P$; // On effectue une addition
    if $(k_i = -1)$ **then**
      $Q \longleftarrow Q - P$; // On effectue une soustraction
  return (Q) // ***Au total, on effectue: (l-1) doublements et (h-1) additions/soustractions (h étant le nombre de bits à 1)***

Algorithm 39 Méthode binaire NAF(2)

input : $NAF(k), P \in E(\mathbb{F}_p)$
output : $Q = [k]P$

begin
  $Q \longleftarrow \infty$;
  for $i \leftarrow l\text{-}1$ **to** 0 **do**
    // début du scanne chiffre par chiffre (à partir du chiffre de de poids fort vers celui de poids faible)
    $Q \longleftarrow 2Q$ // On effectue un doublement
    if $(k_i = 1)$ **then**
      $Q \longleftarrow Q + P$; // On effectue une addition
    if $(k_i = -1)$ **then**
      $Q \longleftarrow Q - P$; // On effectue une soustraction
  return (Q) // ***Au total, on effectue: (l) doublements et (h) additions/soustractions (h étant le nombre de bits à 1) dont une addition/soustraction gratuite(addition d'un point avec le point à l'infini***

Table des figures

Liste des tables

Liste des publications

Article en revue

[1] Faye, Youssou and Niang, Ibrahima and Noël, Thomas, A Survey of Access Control Schemes in Wireless Sensor Networks, WASET, World Academy of Science, Engineering and Technology,59 : 814–823, WASET, 2011.Note : Selected paper from the ICWCSN 2011, Int. Conf. on Wireless Communication and Sensor Networks.

[2] Faye Youssou, Shou Yanbo, Guyennet, Hervé and Niang Ibrahima, Accelerated Parallel Precomputing-based Scalar Reduction on Elliptic Curve Cryptography for Wireless Sensor Networks, Journal of cryptology, Springer. Note : En soumission.

Conférences

[3] Faye, Youssou and Guyennet, Hervé and Niang, I. and Shou, Yanbo, Fast Scalar Multiplication on Elliptic Curve Cryptography in Selected Intervals Suitable For Wireless Sensor Networks, CSS'13, 5th Int. Symposium on Cyberspace Safety and Security,(CSS-2013), pages 171-183,Springer, LNCS, Zhangjiajie, China, November 13-15 2013.

[4] Faye, Youssou and Guyennet, Hervé and Niang, Ibrahima. A User Authentication-Based Probabilistic Risk Approach for Wireless Sensor Networks. In iWMANET 2012, Int. Workshop on Mobile Ad-Hoc Wireless Networks, conjointly organized with IEEE International Conference on Selected Topics(iCOST) 2012, Avignon, France, pages 124-129, July 2012. IEEE Computer Society .

[5] Faye, Youssou and Guyennet, Hervé and Niang, Ibrahima, Authentification Robuste Basée sur une Analyse Probabiliste de Risque d'Attaque par Déni de Service Dans les Réseaux de Capteurs Sans Fil, CARI 2012, 11ème Colloque Africain sur la Recherche en Informatique et en Mathématiques Appliquées, Alger, Algeria, pages 12-17, October 2012. Note : Sponsorisée par INRIA. (Taux d'acceptation 38,27%)

[6] Faye, Youssou and Guyennet, Hervé and Niang, Ibrahima and Shou, Yanbo, Optimisation d'un Protocole d'Authentification dans les Réseaux de Capteurs Sans Fil, SAR-SSI 2012, 7ème Conf. sur la sécurité des architectures réseaux et systèmes d'information (SAR-SSI), ISBN 978-2-9542630-0-7, Cabourg - France, 2012, pages 133–138, May 2012.

BIBLIOGRAPHIE

[1] Partitioned computation to accelerate scalar multiplication for elliptic curve cryptosystems.

[2] IEEE Std 1363-2000. 1363-2000 - ieee standard specifications for public-key cryptography, 2000.

[3] Ian F Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. A survey on sensor networks. *Communications magazine, IEEE*, 40(8) :102–114, 2002.

[4] Ross Anderson, Haowen Chan, and Adrian Perrig. Key infection : Smart trust for smart dust. In *Network Protocols, 2004. ICNP 2004. Proceedings of the 12th IEEE International Conference on*, pages 206–215. IEEE, 2004.

[5] Bijan Ansari and Huapeng Wu. Parallel scalar multiplication for elliptic curve cryptosystems. In *Communications, Circuits and Systems, 2005. Proceedings. 2005 International Conference on*, volume 1, pages 71–73. IEEE, 2005.

[6] B. Arazi. Certification of dl/ec keys. In *In Proceedings of the IEEE P1363 Study Group for Future Public-Key Cryptography Standards.*

[7] Alessandro Armando, David Basin, Yohan Boichut, Yannick Chevalier, Luca Compagna, Jorge Cuéllar, P Hankes Drielsma, Pierre-Cyrille Héam, Olga Kouchnarenko, Jacopo Mantovani, et al. The avispa tool for the automated validation of internet security protocols and applications. In *Computer Aided Verification*, pages 281–285. Springer, 2005.

[8] Alessandro Armando and Luca Compagna. Automatic sat-compilation of protocol insecurity problems via reduction to planning. In *Formal Techniques for Networked and Distributed SytemsÑFORTE 2002*, pages 210–225. Springer, 2002.

[9] Amit K Awasthi and Sunder Lal. A remote user authentication scheme using smart cards with forward secrecy. *Consumer Electronics, IEEE Transactions on*, 49(4) :1246–1248, 2003.

[10] John Wiley & Sons B. Schneier. Applied cryptography. 1996.

[11] J.J. Rodrigues B Vaidya, J.S. Silva. Robust dynamic user authentication scheme for wireless sensor networks. In *5th ACM Symposium on QoS and Security for wireless and mobile networks*, pages 88–91. Q2SWinet, 2009.

[12] Satyajit Banerjee and Debapriyay Mukhopadhyay. Symmetric key based authenticated querying in wireless sensor networks. In *Proceedings of the first international conference on Integrated internet ad hoc and sensor networks*, page 22. ACM, 2006.

[13] Paul Barrett. Implementing the rivest shamir and adleman public key encryption algorithm on a standard digital signal processor. In *Advances in CryptologyÑCRYPTOÕ86 : Proceedings*, pages 311–323. Springer, 2006.

[14] David Basin, Sebastian Mödersheim, and Luca Vigano. *An on-the-fly model-checker for security protocol analysis*. Springer, 2003.

[15] Zinaida Benenson, Felix C Freiling, Ernest Hammerschmidt, Stefan Lucks, and Lexi Pimenidis. Authenticated query flooding in sensor networks. In *Security and Privacy in Dynamic Environments*, pages 38–49. Springer, 2006.

[16] Zinaida Benenson, Felex C Gartner, and Dogan Kesdogan. An algorithmic framework for robust access control in wireless sensor networks. In *Wireless Sensor Networks, 2005. Proceeedings of the Second European Workshop on*, pages 158–165. IEEE, 2005.

[17] Zinaida Benenson, Nils Gedicke, and Ossi Raivio. Realizing robust user authentication in sensor networks. *Real-World Wireless Sensor Networks (REALWSN)*, 14, 2005.

[18] D Bernstein. High-speed diffie-hellman, part 2. In *INDOCRYPT*, volume 6, pages 11–13, 2006.

[19] Daniel J Bernstein. Curve25519 : new diffie-hellman speed records. In *Public Key Cryptography-PKC 2006*, pages 207–228. Springer, 2006.

[20] Daniel J Bernstein and Tanja Lange. Analysis and optimization of elliptic-curve single-scalar multiplication. *Contemporary Mathematics*, 461 :1–20, 2008.

[21] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. The keccak reference. Submission to NIST (Round 3), 2011.

[22] Ian F Blake, Gadiel Seroussi, and Nigel Smart. *Elliptic curves in cryptography*, volume 265. Cambridge university press, 1999.

[23] Carlo Blundo, Alfredo De Santis, Amir Herzberg, Shay Kutten, Ugo Vaccaro, and Moti Yung. Perfectly-secure key distribution for dynamic conferences. In *Advances in cryptologyÑCRYPTOÕ92*, pages 471–486. Springer, 1993.

[24] Michael Brown, Darrel Hankerson, Julio López, and Alfred Menezes. *Software implementation of the NIST elliptic curves over prime fields*. Springer, 2001.

[25] Ran Canetti, Juan Garay, Gene Itkis, Daniele Micciancio, Moni Naor, and Benny Pinkas. Multicast security : A taxonomy and some efficient constructions. In *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 708–716. IEEE, 1999.

[26] Haowen Chan, Adrian Perrig, and Dawn Song. Random key predistribution schemes for sensor networks. In *Security and Privacy, 2003. Proceedings. 2003 Symposium on*, pages 197–213. IEEE, 2003.

[27] Chin-Chen Chang, Ying-Tse Kuo, and Chu-Hsing Lin. Fast algorithms for common-multiplicand multiplication and exponentiation by performing complements. In *Advanced Information Networking and Applications, 2003. AINA 2003. 17th International Conference on*, pages 807–811. IEEE, 2003.

[28] Yannick Chevalier, Luca Compagna, Jorge Cuellar, P Hankes Drielsma, Jacopo Mantovani, Sebastian Mödersheim, Laurent Vigneron, et al. A high level protocol specification language for industrial security-sensitive protocols. *Proc. SAPS*, 4 :193–205, 2004.

[29] Sung Jin Choi and Hee Yong Youn. An efficient key pre-distribution scheme for secure distributed sensor networks. In *Embedded and Ubiquitous Computing–EUC 2005 Workshops*, pages 1088–1097. Springer, 2005.

[30] Jaewook Chung and Anwar Hasan. More generalized mersenne numbers. In *Selected areas in cryptography*, pages 335–347. Springer, 2004.

[31] Mathieu Ciet, Marc Joye, Kristin Lauter, and Peter L Montgomery. Trading inversions for multiplications in elliptic curve cryptography. *Designs, codes and cryptography*, 39(2) :189–206, 2006.

[32] Henri Cohen, Atsuko Miyaji, and Takatoshi Ono. Efficient elliptic curve exponentiation using mixed coordinates. In *Advances in CryptologyÑASIACRYPTÕ98*, pages 51–65. Springer, 1998.

[33] V. Cortier. Vérification automatique des protocoles cryptographiques. "thèse de doctorat, ecole normale supérieure de cachan", 2003.

[34] H.Y. Youn C.W. Park, S.J. Choi. A novel key pre- distribution scheme with lu matrix for secure wireless sensor networks. In *International Con- ference on Computational Intelligence and Security*, pages 494–499. Springer, 2005.

[35] Manik Lal Das, Ashutosh Saxena, and Ved P Gulati. A dynamic id-based remote user authentication scheme. *Consumer Electronics, IEEE Transactions on*, 50(2) :629–631, 2004.

[36] Whitfield Diffie and Martin E Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6) :644–654, 1976.

[37] Vassil Dimitrov, Laurent Imbert, and Pradeep Kumar Mishra. Efficient and secure elliptic curve point multiplication using double-base chains. In *Advances in Cryptology-ASIACRYPT 2005*, pages 59–78. Springer, 2005.

[38] Vassil S Dimitrov, Graham A Jullien, and William C Miller. An algorithm for modular exponentiation. *Information Processing Letters*, 66(3) :155–159, 1998.

[39] Christophe Doche and Laurent Imbert. Extended double-base number system with applications to elliptic curve cryptography. In *Progress in Cryptology-Indocrypt 2006*, pages 335–348. Springer, 2006.

[40] Stefan Dulman, Paul Havinga, Johann Hurink, et al. Wave leader election protocol for wireless sensor networks. 2002.

[41] Bruno Dutertre, Steven Cheung, and Joshua Levy. Lightweight key management in wireless sensor networks by leveraging initial trust. Technical report, Technical Report SRI-SDL-04-02, SRI International, 2004.

[42] Harold Edwards. A normal form for elliptic curves. *Bulletin of the American Mathematical Society*, 44(3) :393–422, 2007.

[43] Thomas Eisenbarth, Sandeep Kumar, Christof Paar, Axel Poschmann, and Leif Uhsadel. A survey of lightweight-cryptography implementations. *IEEE Design & Test of Computers*, 24(6) :522–533, 2007.

[44] Kirsten Eisenträger, Kristin Lauter, and Peter L Montgomery. Fast elliptic curve arithmetic and improved weil pairing evaluation. In *Topics in cryptologyÑCT-RSA 2003*, pages 343–354. Springer, 2003.

[45] Laurent Eschenauer and Virgil D Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 41–47. ACM, 2002.

[46] Youssou Faye, Ibrahima Niang, and Thomas Noel. A survey of access control schemes in wireless sensor networks. *Proc. World Acad. Sci. Eng. Tech*, 59 :814–823, 2011.

[47] Saurabh Ganeriwal, Srdjan Čapkun, Chih-Chieh Han, and Mani B Srivastava. Secure time synchronization service for sensor networks. In *Proceedings of the 4th ACM workshop on Wireless security*, pages 97–106. ACM, 2005.

[48] Gunnar Gaubatz, Jens-Peter Kaps, and Berk Sunar. Public key cryptography in sensor networksÑrevisited. In *Security in Ad-hoc and Sensor Networks*, pages 2–18. Springer, 2005.

[49] Catherine H Gebotys and Robert J Gebotys. Secure elliptic curve implementations : An analysis of resistance to power-attacks in a dsp processor. In *Cryptographic Hardware and Embedded Systems-CHES 2002*, pages 114–128. Springer, 2003.

[50] Daniel M Gordon. A survey of fast exponentiation methods. *Journal of algorithms*, 27(1) :129–146, 1998.

[51] Johann Großschädl, Roberto M. Avanzi, Erkay Savaş, and Stefan Tillich. Energy-efficient software implementation of long integer modular arithmetic. In *Proceedings of the 7th International Conference on Cryptographic Hardware and Embedded Systems*, CHES'05, pages 75–90. Springer-Verlag, 2005.

[52] Jorge Guajardo and Christof Paar. *Efficient algorithms for elliptic curve cryptosystems*. Springer, 1997.

[53] Vipul Gupta, Michael Wurm, Yu Zhu, Matthew Millard, Stephen Fung, Nils Gura, Hans Eberle, and Sheueling Chang Shantz. Sizzle : A standards-based end-to-end security architecture for the embedded internet. *Pervasive and Mobile Computing*, 1(4) :425–445, 2005.

[54] Nils Gura, Arun Patel, Arvinderpal Wander, Hans Eberle, and Sheueling Chang Shantz. Comparing elliptic curve cryptography and rsa on 8-bit cpus. In *Cryptographic Hardware and Embedded Systems-CHES 2004*, pages 119–132. Springer, 2004.

[55] Darrel Hankerson, Scott Vanstone, and Alfred J Menezes. *Guide to elliptic curve cryptography*. Springer, 2004.

[56] Michael Healy, Thomas Newe, and Elfed Lewis. Analysis of hardware encryption versus software encryption on wireless sensor network motes. In *Smart Sensors and Sensing Technology*, pages 3–14. Springer, 2008.

[57] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. System architecture directions for networked sensors. In *ACM SIGOPS operating systems review*, volume 34, pages 93–104. ACM, 2000.

[58] Hui-Feng Huang. A novel access control protocol for secure sensor networks. *Computer Standards & Interfaces*, 31(2) :272–276, 2009.

[59] Hui-Feng Huang and Kuo-Ching Liu. A new dynamic access control in wireless sensor networks. In *Asia-Pacific Services Computing Conference, 2008. APSCC'08. IEEE*, pages 901–906. IEEE, 2008.

[60] Min-Shiang Hwang, Chin-Chen Chang, and Kuo-Feng Hwang. An elgamal-like cryptosystem for enciphering large messages. *Knowledge and Data Engineering, IEEE Transactions on*, 14(2) :445–446, 2002.

[61] Texas Instruments. Chipcon cc2420 2.4ghz ieee 802.15.4 rf transceiver., 2007.

[62] Texas Instruments. Cc1101 low-power sub-1ghz rf transceiver., 2011.

[63] Hyun-Sung Kim and Sung-Woon Lee. Enhanced novel access control protocol over wireless sensor networks. *Consumer Electronics, IEEE Transactions on*, 55(2) :492–498, 2009.

[64] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177) :203–209, 1987.

[65] Ted Krovetz and Phillip Rogaway. Fast universal hashing with small keys and no preprocessing : The polyr construction. In *Information Security and CryptologyÑICISC 2000*, pages 73–89. Springer, 2001.

[66] Bocheng Lai, Sungha Kim, and Ingrid Verbauwhede. Scalable session key construction protocol for wireless sensor networks. In *IEEE Workshop on Large Scale RealTime and Embedded Systems (LARTES)*, page 7. Citeseer, 2002.

[67] Leslie Lamport. Password authentication with insecure communication. *Communications of the ACM*, 24(11) :770–772, 1981.

[68] Yee Wei Law, Jeroen Doumen, and Pieter Hartel. Survey and benchmark of block ciphers for wireless sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 2(1) :65–93, 2006.

[69] Cheng-Chi Lee, Li-Hua Li, and Min-Shiang Hwang. A remote user authentication scheme using hash functions. *ACM SIGOPS Operating Systems Review*, 36(4) :23–29, 2002.

[70] Chia-Yin Lee, Chu-Hsing Lin, and Chin-Chen Chang. An improved low computation cost user authentication scheme for mobile communication. In *Advanced Information Networking and Applications, 2005. AINA 2005. 19th International Conference on*, volume 2, pages 249–252. IEEE, 2005.

[71] Hwaseong Lee, Kilho Shin, and Dong Hoon Lee. Practical access control protocol for secure sensor networks. In *2009 IEEE 13th International Symposium on Consumer Electronics*, pages 806–809, 2009.

[72] Hendrik W Lenstra et al. *Elliptic curves and number-theoretic algorithms*. Universiteit van Amsterdam, Mathematisch Instituut, 1986.

[73] Hendrik W Lenstra Jr. Factoring integers with elliptic curves. *Annals of mathematics*, pages 649–673, 1987.

[74] Chae Hoon Lim and Pil Joong Lee. More flexible exponentiation with precomputation. In *Advances in cryptologyÑCRYPTOÕ94*, pages 95–107. Springer, 1994.

[75] Donggang Liu, Peng Ning, and Rongfang Li. Establishing pairwise keys in distributed sensor networks. *ACM Transactions on Information and System Security (TISSEC)*, 8(1) :41–77, 2005.

[76] Patrick Longa. *Accelerating the scalar multiplication on elliptic curve cryptosystems over prime fields*. PhD thesis, University of Ottawa, 2007.

[77] Patrick Longa and Ali Miri. Fast and flexible elliptic curve point arithmetic over prime fields. *Computers, IEEE Transactions on*, 57(3) :289–302, 2008.

[78] Samuel Madden, Michael J Franklin, Joseph M Hellerstein, and Wei Hong. The design of an acquisitional query processor for sensor networks. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 491–502. ACM, 2003.

[79] Nicolas Meloni. Fast and secure elliptic curve scalar multiplication over prime fields using special addition chains. Technical report, Cryptology ePrint Archive, Report 2006/216, 2006.

[80] Nicolas Méloni and M Anwar Hasan. Elliptic curve scalar multiplication combining yaoÕs algorithm and double bases. In *Cryptographic Hardware and Embedded Systems-CHES 2009*, pages 304–316. Springer, 2009.

[81] Peter L Montgomery. Modular multiplication without trial division. *Mathematics of computation*, 44(170) :519–521, 1985.

[82] Volker Müller. *Efficient algorithms for multiplication on elliptic curves*. Springer, 1998.

[83] Dalit Naor, Moni Naor, and Jeff Lotspiech. Revocation and tracing schemes for stateless receivers. In *Advances in CryptologyÑCRYPTO 2001*, pages 41–62. Springer, 2001.

[84] Katsuyuki Okeya, Katja Schmidt-Samoa, Christian Spahn, and Tsuyoshi Takagi. Signed binary representations revisited. In *Advances in Cryptology–CRYPTO 2004*, pages 123–139. Springer, 2004.

[85] Adrian Perrig, John Stankovic, and David Wagner. Security in wireless sensor networks. *Communications of the ACM*, 47(6) :53–57, 2004.

[86] Adrian Perrig, Robert Szewczyk, JD Tygar, Victor Wen, and David E Culler. Spins : Security protocols for sensor networks. *Wireless networks*, 8(5) :521–534, 2002.

[87] M. Rivain. Fast and regular algorithms for scalar multiplication. Technical report, Cryptology ePrint Archive, Report 2011/338, 2011.

[88] MJB Robshaw and Yiqun Lisa Yin. Elliptic curve cryptosystems. *An RSA Laboratories Technical Note*, 1 :997, 1997.

[89] Michaël Rusinowitch, Mathieu Turuani, et al. Protocol insecurity with finite number of sessions is np-complete. *CSFW'01*, page 174190, 2001.

[90] Yasuyuki Sakai and Kouichi Sakurai. Efficient scalar multiplications on elliptic curves without repeated doublings and their practical performance. In *Information Security and Privacy*, pages 59–73. Springer, 2000.

[91] J. Santos Santiago. Analyse automatique de protocoles avec atse. *AFADL'04*, 2004.

[92] Erkay Savas and Cetin Kaya Koç. The montgomery modular inverse-revisited. *Computers, IEEE Transactions on*, 49(7) :763–766, 2000.

[93] Stefaan Seys and Bart Preneel. Efficient cooperative signatures : A novel authentication scheme for sensor networks. In *Security in Pervasive Computing*, pages 86–100. Springer, 2005.

[94] Jau-Ji Shen, Chih-Wei Lin, and Min-Shiang Hwang. A modified remote user authentication scheme using smart cards. *Consumer Electronics, IEEE Transactions on*, 49(2) :414–416, 2003.

[95] Yanbo Shou, Herve Guyennet, and Mohamed Lehsaini. Parallel scalar multiplication on elliptic curves in wireless sensor networks. In *Distributed Computing and Networking*, pages 300–314. Springer, 2013.

[96] Hung-Min Sun. An efficient remote use authentication scheme using smart cards. *Consumer Electronics, IEEE Transactions on*, 46(4) :958–961, 2000.

[97] Kun Sun, An Liu, Roger Xu, Peng Ning, and Douglas Maughan. Securing network access in wireless sensor networks. In *Proceedings of the second ACM conference on Wireless network security*, pages 261–268. ACM, 2009.

[98] Min Tian, Jizhi Wang, Yinglong Wang, and Shujiang Xu. An efficient elliptic curve scalar multiplication algorithm suitable for wireless network. In *Networks Security Wireless Communications and Trusted Computing (NSWCTC), 2010 Second International Conference on*, volume 1, pages 95–98. IEEE, 2010.

[99] Huei-Ru Tseng, Rong-Hong Jan, and Wuu Yang. An improved dynamic user authentication scheme for wireless sensor networks. In *Global Telecommunications Conference, 2007. GLOBECOM'07. IEEE*, pages 986–990. IEEE, 2007.

[100] E. Masato V. Suppakitpaisarn, H. Imai. Fastest multi-scalar multiplication based on optimal double-base chains.

[101] Binod Vaidya, Min Chen, and Joel JPC Rodrigues. Improved robust user authentication scheme for wireless sensor networks. In *Wireless Communication and Sensor Networks (WCSN), 2009 Fifth IEEE Conference on*, pages 1–6. IEEE, 2009.

[102] S Vanstone. Ecc holds key to next-gen cryptography. *Rapport technique, Certicom*, 2004.

[103] Haodong Wang and Qun Li. Efficient implementation of public key cryptosystems on mote sensors (short paper). In *Information and Communications Security*, pages 519–528. Springer, 2006.

[104] Kirk HM Wong, Yuan Zheng, Jiannong Cao, and Shengwei Wang. A dynamic user authentication scheme for wireless sensor networks. In *Sensor Networks, Ubiquitous, and Trustworthy Computing, 2006. IEEE International Conference on*, volume 1, pages 8–pp. IEEE, 2006.

[105] Boichut Yohan. Approximations pour la vérification automatique de protocoles de sécurité " thèse de doctorat ", 2006.

[106] SHEN Yu-long, Ma Jian-Feng, and PEI Qing-qi. An access control scheme in wireless sensor networks. In *Network and Parallel Computing Workshops, 2007. NPC Workshops. IFIP International Conference on*, pages 362–367. IEEE, 2007.

[107] Peng Zeng, K-KR Choo, and Da-Zhi Sun. On the security of an enhanced novel access control protocol for wireless sensor networks. *Consumer Electronics, IEEE Transactions on*, 56(2) :566–569, 2010.

[108] Yanchao Zhang, Wei Liu, Wenjing Lou, and Yuguang Fang. Location-based compromise-tolerant security mechanisms for wireless sensor networks. *Selected Areas in Communications, IEEE Journal on*, 24(2) :247–260, 2006.

[109] Yun Zhou, Yanchao Zhang, and Yuguang Fang. Access control in wireless sensor networks. *Ad Hoc Networks*, 5(1) :3–13, 2007.

[110] Sencun Zhu, Sanjeev Setia, and Sushil Jajodia. Leap+ : Efficient security mechanisms for large-scale distributed sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 2(4) :500–528, 2006.

Résumé :

Un réseau de capteurs sans fil (RCSF) est constitué d'un grand nombre de nœuds capteurs autonomes qui collaborent ensemble pour la surveillance d'une zone, d'une machine, d'une personne etc.. Dans certaines applications, les données critiques doivent être protégées contre toute utilisation frauduleuse et être accessibles en temps réel. Le besoin d'apporter une solution de sécurité fiable et adaptée paraît donc essentiel. Les solutions de sécurité utilisées dans les réseaux traditionnels ne sont pas directement applicables dans les RCSFs, car développer des primitives de sécurité en utilisant de faibles ressources devient un véritable défi. Dans cette thèse, nous proposons des solutions nouvelles peu gourmandes en ressources qui tiennent compte des faibles capacités de défense d'un réseau autonome. Dans cette optique nous appliquons des mécanismes cryptographiques basés sur les fonctions de hachage et les courbes elliptiques. Un focus sur différents mécanismes de sécurité peu gourmands en ressources nous permet la mise en évidence des rapports de forces entre les RCSFs et leurs vulnérabilités. Notre première contribution vise à améliorer la sécurité et les performances en termes d'énergie sur des protocoles d'authentification existants tout en utilisant les mêmes mécanismes. Dans la deuxième contribution, on utilise le concept de probabilité de risque afin de déterminer la consommation énergétique dans différentes architectures de déploiement. Dans la troisième contribution nous présentons un nouveau mécanisme d'accélération de la multiplication scalaire sur les courbes elliptiques définies dans des corps finis premiers. Ce mécanisme basé sur l'opposé et l'ordre d'un point, réduit le nombre d'opérations de points dans un intervalle donné, et présente en plus l'avantage de pouvoir être combiné avec les techniques existantes. Enfin dans notre dernière contribution, nous nous sommes intéressés à l'accélération du calcul des points résultants du partitionnement du scalaire qui introduisent des coûts additionnels de calcul et de stockage mémoire. Nous comparons différentes formules de points existantes en mettant en évidence leur efficacité.

Mots-clés : Réseaux de Capteurs Sans Fil, Authentification, Courbes elliptiques pour la cryptographie

Abstract:

A Wireless Sensor Network (WSN) consists of a large number of sensor nodes which collaborate so as to monitor environnement. For various WSNs' applications, the collected data should be protected by preventing unauthorized users from gaining the information. The need to find a reliable and adaptive security solution is very important. Most current standard security protocols designed for traditional networks cannot be applied directly in WSN. For this reason, providing a variety of security functions with limited resources is a real challenge. Our research work seeks to find secure efficient solutions that take into account the rather weak defense of an autonomous network. In this way, we apply lightweight cryptography mechanisms based on hash function and elliptic curves. A focus on different security mechanisms and lightweight security algorithms can highlight the strength ratio between WSNs and their vulnerabilities. Our first contribution is on a secure energy efficient solution, it uses the same mechanism and aims to enhance the security weaknesses of existing solutions. The second contribution uses the concept of probability risk analysis to show to which level the proposed solution justifies the better energy consumption for a given network architecture. In the third contribution, we present a new technique to accelerate scalar multiplication on elliptic curves cryptography over prime field for light-weight embedded devices like sensor nodes. Our method reduces the computation of scalar multiplication by an equivalent representation of points based on point order in a given interval and can also act as a support for most existing methods. Finally our last contribution presents a fast pre-computation algorithm in a parallel scalar multiplication to avoid the storage of pre-computation points which requires extra memory. We also provide a comparison of different formulas so as to find out their efficiency.

Keywords: Wireless Sensor Network, Authentication, Elliptic Curves Cryptography

■ École doctorale SPIM 16 route de Gray F - 25030 Besançon cedex

■ tél. +33 (0)3 81 66 66 02 ■ ed-spim@univ-fcomte.fr ■ www.ed-spim.univ-fcomte.fr