# Simulation de systèmes cyber physiques utilisant les polynômes de Chebyshev pour une distribution efficace des clés symétriques

INF6422 – Concepts avancés en sécurité informatique

***Abstract*—Avec l'inclusion d'appareils mobiles et autres capteurs formant l'internet des objets, une gestion des clés sécuritaire est devenue indispensable pour assurer la confidentialité des échanges de données entre les nœuds des réseaux cyber physiques. Pour cela, un schéma d'établissement des clés multi parties assure une sécurité renforcée notamment en utilisant des paramètres partagés par les nœuds voisins pour générer la clé. Dans cette optique, notre étude porte sur un schéma de gestion des clés multi distribuée (DMK pour *Distributed Multiparty Key*) utilisant les cartes chaotiques pour calculer des empreintes numériques et les polynômes de Chebyshev pour l'établissement des clés. Ce système a été simulé à l'aide de NS-3.30 afin de pouvoir analyser ses performances comparativement à d'autres algorithmes de chiffrement, en fonction de la topologie du réseau, ainsi que d'autres critères de performance.**

***Keywords*—** cartes chaotiques, polynômes de Chebyshev, gestion des clés distribuée, internet des objets

## I. Introduction

Les réseaux de capteurs sans fil forment le socle d'un nombre croissant d'infrastructures critiques utilisant l'internet des objets. Ces réseaux sont notamment utilisés dans l'industrie, dans le domaine médical, les activités militaires, l'agriculture, etc. En raison de leurs impacts économiques, stratégiques et humains, la sécurité des réseaux de capteurs sans fil est d'une importance capitale. Deux aspects apparaissent comme critiques pour l'internet des objets et les nœuds des réseaux de capteurs: la préservation de la confidentialité et l'authentification des nœuds [1] [2].

Malheureusement, les systèmes cyber physiques ont souvent des contraintes matérielles telles qu'une consommation limitée en énergie et des contraintes physiques. Cela est particulièrement vrai pour les réseaux de capteurs. De ce fait, il est important de pouvoir trouver un équilibre entre les fonctionnalités supportées par ces systèmes et la demande en ressources de celles-ci. Ces ressources peuvent autant être la puissance de calcul demandée que la mémoire requise pour effectuer certaines opérations [3]. Ces problématiques ont motivé plusieurs chercheurs à trouver des solutions pour assurer la sécurité des réseaux de l'internet des objets.

Notre étude se concentrera sur les opérations d'échange de clés et de chiffrement afin de garantir la confidentialité et l'authentification au sein des réseaux cyber physiques. Pour cela, nous confirmerons que l'utilisation d'un chiffrement basé sur les cartes chaotiques et les polynômes de Chebyshev (CP) est moins gourmande en termes de ressources que d'autres types d'algorithmes comme Diffie-Hellman avant d'analyser l'impact de la topographie du réseau sur la performance de ce type de chiffrement.

Plus précisément, nous évaluerons la performance d'un chiffrement basé sur les cartes chaotiques et les polynômes de Chebyshev comparativement à Diffie-Hellman et SHA256 en se fondant sur les critères suivants: le temps de latence et la mémoire utilisée pour le chiffrement. Notre recherche est un prolongement de l'article intitulé "Distributed MultipartyKey Management for Efficient Authentication in the Internet of Things" [8], dans lequel les auteurs étudient des critères de performance différents des nôtres. En plus d'introduire de nouveaux critères de performance, notre étude apporte également un nouvel angle d'analyse en considérant l'impact de la topologie du réseau sur la performance d'un chiffrement basé sur les CP.

Nous allons donc commencer cet article scientifique par un état de l'art des différentes méthodes de chiffrement recensées dans la littérature portant sur les réseaux de type internet des objets avant de présenter la théorie sous-jacente à la méthode étudiée. Plus précisément, le hachage basé sur la théorie du chaos et le chiffrement à l'aide des polynômes de Chebyshev vont être élaborés. Par la suite, la méthodologie de l'implantation du chiffrement et les différentes méthodes de test seront expliquées avant de présenter nos résultats.

## II. État de l'art

Afin de garantir la sécurité de l'internet des objets (IoT), plusieurs exigences doivent être satisfaites, et ce pour chacune des couches de l'architecture généralement reconnues pour l'IoT. Cette architecture se compose de trois couches: la couche de perception, la couche réseau et la couche applicative. Dans [2], les auteurs énoncent les requis de sécurité suivants:

- Couche de perception: 1. Chiffrement léger 2. Authentification 3. Validation des clés 4. Confidentialité des données
- Couche réseau: 1. Communication sécurisée 2. Routage sécurisé 3. Authentification 4. Gestion des clés 5. Détection d'intrusion
- Couche applicative: 1. Authentification 2. Protection de la vie privée 3. Gestion de la sécurité de l'information

Actuellement, un des défis auquel font face les chercheurs en sécurité dans le domaine de l'internet des objets et des réseaux de capteurs sans fil est la gestion des clés [4]. En effet, les données transmises par les capteurs doivent être protégées des attaques malicieuses notamment lorsqu'elles sont en transit. Pour cela, l'utilisation d'algorithmes connus de cryptographie à clé publique telle que RSA ou encore les courbes elliptiques (ECC) pourraient être envisagée. Toutefois l'implantation est limitée principalement en raison des capacités de calcul et énergétiques restreintes des nœuds formant les réseaux de capteurs sans fil. [5] [6]

Afin de pallier à ces limites, un volet de la recherche s'est penché sur la création de schémas de gestion de clés efficaces. Les auteurs ont proposé diverses options notamment basées sur les sous-ensembles polynomiaux [7]. La table 1 offre un aperçu des principales caractéristiques des travaux réalisés jusqu'à présent dans ce domaine. Dernièrement, Mahmood et al. ont proposé un modèle utilisant les cartes chaotiques pour le calcul d'empreinte numérique et l'utilisation des polynômes de Chebyshev pour le chiffrement [8].

Dans leur étude, Mahmood et al. utilisent les cartes chaotiques pour fournir des fonctions de hachage à sens unique tandis que les polynômes de Chebyshev sont utilisés pour générer des clés secrètes et des clés de session. Selon les auteurs, ce type de chiffrement permet de garantir la confidentialité lors de l'échange des données entre les nœuds tout en ayant une consommation (*coût de communication, nombre d'opérations de hachage, coût de calcul*) plus faible que des systèmes semblables utilisant RSA ou ECC.

TABLE I
RÉSUMÉ DES CARACTÉRISTIQUES SUR LA GESTION DES CLÉS DANS L'INTERNET DES OBJETS [4].

| | Principales caractéristiques |
|---|---|
| 1 | Schéma *Certificateless Effective Key Management protocol (CL-EKM)* pour réduire le coût énergétique [9] |
| 2 | 1. Algorithme d'authentification léger<br>2. Algorithme d'allocation dynamique [10] |
| 3 | 1. Technique de clé publique ou privée<br>2. Utilisation d'un algorithme pour déterminer le plus court chemin [11] |
| 4 | Environnement dynamique où des nœuds peuvent quitter ou se joindre au réseau [12] |
| 5 | 1. Utilisation de technique de regroupement<br>2. XOR et calculs polynomiaux [7] |

## III. THÉORIE

### A. Hachage basé sur le chaos

La cryptographie chaotique est un domaine de recherche permanent [13]. La théorie du chaos permet en autres de créer des fonctions de hachage [14]. Ces fonctions permettent d'authentifier des messages tout en améliorant la sensibilité au message d'origine [15] [16] [17].

Les systèmes de chaos sont des systèmes dynamiques non linéaires. En fonction de la plage de temps, ils sont décrits par des équations aux différences (*systèmes à temps discret*) ou par des équations différentielles (*systèmes à temps continu*) [18] [19].

Il existe donc deux modèles principaux de loi d'évolution. Le premier correspond aux transformations $f : M \to M$ sur un espace $M$, où les points décrivent les différents états du système. Le second est celui des flux de temps continu $f^t : M \to M, t \in \mathbb{R}$ c'est-à-dire des transformations satisfaisant $f^{t+s} = f^t \circ f^s$ pour $t, s \in \mathbb{R}$ et $f^0 = id$.

Afin de créer cette boîte noire, M. A. Czyzewski [19], a développé le concept de machine chaotique. Tel qu'illustré dans la figure 1. ce type de machine repose sur trois éléments principaux: une fonction push, un espace tampon et une fonction pull. Ces machines sont initialisées par un tuple ($K$, $t$, $m$) où $K$ est la clé secrète initiale, $t$ le paramètre temps et $m$ le paramètre d'espace.

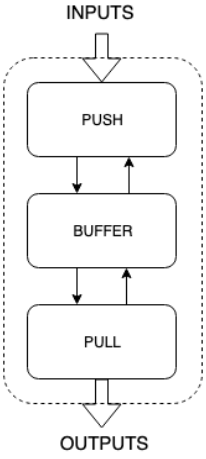


Fig. 1. Machine chaotique.

Cette boîte noire peut être expliquée de manière plus mathématique comme le font S. G. Lian et J. Sun [20]. Le hachage prend donc en entrée un texte et vise à le transformer en un texte chiffré. Cette transformation peut être exprimée avec la fonction suivante :

$$\mathbf{C} = D^{\alpha}(C^{\beta}(\mathbf{T}, K_C), K_D)$$

**T** et **C** représentent le texte et le texte chiffré. $C$ et $D$ sont des fonctions de confusion et diffusion, $K_C$ et $K_D$ sont les clés de ces fonctions et $\alpha$ et $\beta$ représentent l'ordre ou le nombre de fois que l'on repasse dans la fonction de confusion et de diffusion.

### B. Polynômes de Chebyshev et chiffrement

Depuis plusieurs années, la communauté scientifique considère l'utilisation des polynômes de Chebyshev pour le chiffrement à clé publique [21]. Les études portent principalement sur les chiffrements RSA et ElGamal ainsi que sur l'échange de clés Diffie-Hellman [22] [23].

Récemment, les chercheurs ont commencé à s'intéresser à l'utilisation des polynômes de Chebyshev et des cartes chaotiques [8] [24] [25] [26] En effet, le chaos est désormais traité comme une approche intéressante pour réduire la complexité computationnelle tout en répondant aux exigences de sécurité des méthodes de chiffrement [24].

Les polynômes de Chebyshev sont définis comme suit :

$$T_0(x) = 1$$

$$T_1(x) = x$$

$$T_2(x) = 2x^2 - 1$$

$$T_{p+1}(x) = 2xT_p(x) - T_{p-1}(x)$$

Il s'agit des polynômes de Chebyshev de base, mais cet article va utiliser une version améliorée de ces polynômes. Cet engouement pour ces nouveaux polynômes de Chebyshev s'explique par plusieurs de leurs propriétés dans le domaine des nombres premiers réels et modulaires:

- Soit $n \in \mathbb{N}$ et $x \in [-1, 1]$, un polynôme de Chebyshev $T_n(x) : [-1, 1] \to [-1, 1]$ est défini tel que:

$$T_n(x) = \cos(n \arccos(x)) \qquad [8] [26]$$

- Pour l'intervalle $(-\infty, +\infty)$, la définition du polynôme devient:

$$T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x)$$

$$\text{avec } p \in \mathbb{N},\ x \in [0, p-1] \text{ et } n \in \mathbb{N} \qquad [27]$$

- Pour tracer $T_n : \mathbb{R} \to \mathbb{R}$ de degré $n$, le polynôme de Chebyshev a la propriété de semi-groupe suivante:

$$T_r(T_s(x)) = T_s(T_r(x)) = T_{sr}(x)$$

Les propriétés des polynômes de Chebyshev posent plusieurs problèmes de calculs liés aux cartes chaotiques qui sont fort intéressants d'un point de vue de chiffrement [25] [28]:

1) *Chaotic Maps-based Discrete Logarithm (CMBDLP)*: Étant donné $y \in [0, 1]$, et $x$, il est computationnellement impossible de trouver $r$ tel que $T_r(x) = y$.
2) *Diffie-Hellman problem chaotic maps (CMDHP)*: Étant donné $x \in [0, p-1]$, $p \in \mathbb{P}$, $T_r \mod p$ et $T_s \mod p$ il est computationellement impossible de trouver $T_{rs}(x) \mod p$

En sachant que l'échange de clé à l'aide de Diffie-Hellman se fait avec la propriété des exposants: $(a^b)^c = (a^c)^b = a^{bc}$, il s'agit d'un problème semblable à celui de Chebychev qui se pose avec les logarithmes discrets pour retrouver la valeur des exposants. Cependant, selon [37] le problème pour retrouver la valeur de l'ordre des exposants des polynômes de Chebychev peuvent prendre quatre à huit fois plus de temps de calcul que le problème des logarithmes discrets de Diffie-Hellman.

### C. Combinaison des polynômes de Chebyshev et des cartes chaotiques

Ces deux grandes théories peuvent être combinées pour créer un système de chiffrement très sécuritaire. Tel que présenté précédemment, les cartes chaotiques ont besoin de clés pour chiffrer des textes. Ces clés secrètes vont pouvoir être générées par les polynômes de Chebyshev. Ainsi il est possible d'assurer la sécurité des communications sur des canaux publics.

Les propriétés chaotiques des polynômes leur permettent donc de générer ces clés. Les suites logiques sont largement utilisées pour ce type d'application et sont en réalité le conjuré des polynômes de Chebychev d'ordre deux. Selon Geisel, T. et Fairen, V. [38], c'est la propriété des semi-groupes, la propriété d'orthogonalité et la relation de récurrence des polynômes de Chebyshev qui leur permettent d'être un aussi bon candidat pour générer du chaos.

La fiabilité des cartes chaotiques est basée sur les fonctions de hachage à sens unique. Lors de la distribution des clés secrètes, celles-ci ne sont jamais exposées sur le réseau et seules les informations d'identification de clé sont échangées. De plus, ces dernières ne sont jamais transmises sur le même canal que celui utilisé pour l'échange des informations confidentielles.

### D. Architecture distribuée à groupes multiples

L'architecture étudiée dans cet article se compose de trois niveaux avec au sommet un serveur de confiance S, au centre des têtes de groupes GH et à la base les utilisateurs $U$. Cette architecture est illustrée à la figure 2.

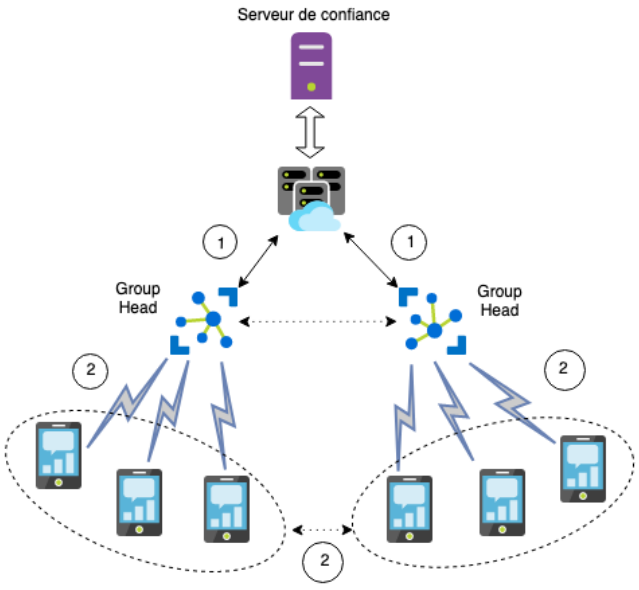


Fig. 2. Architecture distribuée à plusieurs groupes.

Il s'agit d'une architecture répandue dans le domaine de l'internet des objets. Généralement le serveur sera hébergé dans le nuage et accessible depuis internet. En effet, les services infonuagiques offrent une grande fiabilité et une disponibilité accrue des données notamment grâce à la redondance. De plus, les ressources disponibles varient selon la demande.

Dans cette architecture générique, les têtes de groupe peuvent prendre plusieurs formes en fonction du type de réseau dans lequel se trouvent les objets connectés. Si les objets se connectent via le réseau cellulaire, les têtes de groupe vont être des noeuds de ce réseau. Si les objets se connectent plutôt via Wi-Fi, Bluethooth, ZigBee, etc. les têtes de groupes peuvent être des points d'accès, des commutateurs ou encore d'autres appareils derrière ces derniers.

TABLE II
LISTE ET DÉFINITION DES VARIABLES UTILISÉES.

| Variable | Définition |
|---|---|
| S | Le serveur de confiance. |
| $\text{GH}_i$ | La $i^{\text{ème}}$ tête de groupe. |
| $ID_i$ | L'identifiant de la $i^{\text{ème}}$ tête de groupe. |
| $PW_i$ | Le mot de passe de la $i^{\text{ème}}$ tête de groupe. |
| $U_{i,j}$ | L'utilisateur $j$ de la tête de groupe $i$. |
| $ID_{i,j}$ | L'identifiant de l'utilisateur $j$ de la tête de groupe $i$. |
| $PW_{i,j}$ | Le mot de passe de l'utilisateur $j$ de la tête de groupe $i$. |
| $s$ | Entier aléatoire du serveur pour la clé de session. |
| $g$ | Entier aléatoire de la tête de groupe pour la clé de session. |
| $gh$ | Entier aléatoire de la tête de groupe pour la clé intra-groupe. |
| $u$ | Entier aléatoire de l'utilisateur pour la clé intra-groupe. |
| $x$ | Nombre aléatoire entre -1 et 1 pour la clé de session. |
| $r_{\text{GH}_i}$ | Nombre aléatoire du GH entre -1 et 1 pour la clé intra-groupe. |
| $r_{U_{i,j}}$ | Nombre aléatoire de $U$ entre -1 et 1 pour la clé intra-groupe. |
| $T_x(y)$ | Polynôme de Chebychev d'ordre $x$ avec $y$ entre -1 et 1. |
| $Q$ | Polynôme de Chebychev public du serveur. |
| $R_i$ | Polynôme de Chebychev public de la tête de groupe $i$. |
| $AID$ | Valeur d'authentification de la tête de groupe $i$ au serveur. |
| $\tau_i$ | Valeur d'authentification de la tête de groupe $i$ au serveur. |
| $\tau_{s,i}$ | Valeur d'authentification du serveur à la tête de groupe $i$. |
| $K$ | Entier aléatoire pour calculer $RvK$. |
| $RvK$ | Valeur pour augmenter l'entropie de la clé. |
| $KGS_i$ | Clé de session entre la tête de groupe $i$ et le serveur. |
| $h_{PW_{i,j}}$ | Valeur hachée du mot de passe et de l'identifiant de $U_{i,j}$ |
| $KGU_{i,j}$ | Clé intra-groupe entre l'utilisateur $j$ et la tête de groupe $i$. |
| $KU_{i,j}U_{k,l}$ | Clé inter-groupe entre $U_{i,j}$ et $U_{k,l}$. |
| $E_s[]$ | Chiffrement du message avec la clé $s$. |
| $h()$ | Hachage de ce qui se trouve entre parenthèses. |
| $MAC()$ | Information vérifiant l'authenticité d'un message. |
| $(x, y, z)$ | Concaténation de $x$, $y$ et $z$. |
| $\oplus$ | Opération xor. |
| $m_i[]$ | Message $i$. |
| $t_s$ | Estampille temporelle de l'envoi du message. |
| $t_{now}$ | Estampille temporelle du moment présent. |
| $\Delta t$ | Différence maximale d'estampille temporelle entre l'envoi et la réception du message. |

Cette architecture nous permet de mettre en place un schéma de gestion des clés multi distribuée (DMK pour *Distributed Multyparty Key*) tel que défini dans [8]. Ce système fonctionne en trois étapes:

1) Les têtes de groupe s'identifient au serveur de confiance et établissent une clé de session.
2) Les membres des groupes établissent des clés intragroupes avec leur tête de groupe.
3) Les utilisateurs établissent des clés intergroupes entre eux.

La première étape comprend une section d'authentification à double sens entre les têtes de groupe et le serveur de confiance. Cette authentification se fait à l'aide des identifiants et mots de passe des têtes de groupe ainsi qu'à l'aide de la propriété des semi-groupes des polynômes de Chebychev. Ensuite, une clé de session est établie en utilisant tous les paramètres secrets des GH ainsi que les polynômes de Chebychev préalablement établis.

Dans la deuxième étape, les nœuds membres du groupe sont authentifiés auprès de leur GH respectif à l'aide de leurs identifiants, mots de passe et de la propriété des semi-groupes des polynômes de Chebychev. Le serveur utilise les paramètres des groupes multiples pour générer une clé commune et l'échange avec les nœuds et le GH.

À la troisième étape, une clé intergroupe est calculée pour la communication entre les utilisateurs. Elle est constituée des identifiants et mots de passe des utilisateurs ainsi que des polynômes de Chebychev de leur GH respectif.

La section suivante présente les algorithmes utilisés pour mettre en place cette architecture distribuée à groupe multiple.

### *E. Algorithmes*

Avant de commencer à présenter les algorithmes permettant l'authentification et la génération des clés, précisons que le tableau II liste les variables utilisées ainsi que les opérateurs mathématiques nécessaires, et ce pour faciliter la compréhension des algorithmes.

Afin d'implémenter l'architecture distribuée à groupes multiples que nous venons de présenter, nous avons du développer les algorithmes d'échange de clé basé sur les polynômes de Chebyshev et les cartes chaotiques. Les figures 3, 4 et 5 illustrent le mécanisme d'authentification et d'échange de clé entre le serveur de confiance S et les têtes de groupe $GH_i$, entre les utilisateurs $U_{i,j}$ et la tête de groupe $GH_i$ ainsi qu'entre les différents utilisateurs.

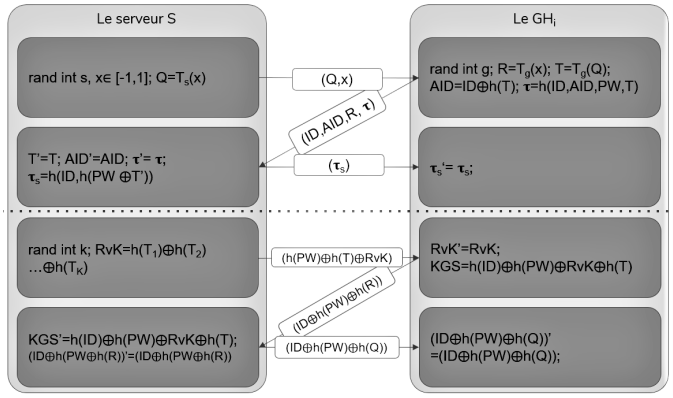


Fig. 3. Échange de clés entre les têtes de groupe et le serveur de confiance.

Les algorithmes détaillés utilisés pour réaliser notre étude sont présentés en annexe de cet article. Il convient de souligner que les figures 3, 4 et 5 résument ces algorithmes en omettant plusieurs détails comme la vérification d'intégrité des messages à l'aide des estampilles temporelles et des hachages des messages. Le chiffrement des messages est aussi omis pour simplifier la lecture de ces pseudo-codes.

En ce qui concerne l'échange de clé entre le serveur de confiance et les têtes de groupe, nous nous sommes inspirés du travail de Mahmood et al [8] et du travail de Lee C., C. C. et Li C. T. [35]. Le haut de la figure 3 présente l'authentification des GH et de S. L'authentification est schématisée en haut de la ligne pointillée et comporte trois messages qui ne sont pas chiffrés. Le premier message provenant du serveur est générique et donc envoyé à tous les GH. L'authentification est présentée avec plus de détail à la section VII-A.

Les trois messages en dessous de la ligne pointillée servent à la génération de clé de session et partent au début du serveur, car ce dernier doit attendre d'avoir bien authentifié tous les GH avant de commencer la génération de clé, étant donné que le paramètre $RvK$ requiert de l'information provenant de l'authentification de tous les GH. Les deux derniers messages de la génération de clé servent à valider la clé de session et sont chiffrés à l'aide de cette dernière. La génération de clé de session est présentée avec plus de détail à la section VII-B.La clé de session utilisée entre les GH et S prend la forme suivante: $KGS_i = h(ID_i) \oplus h(PW_i) \oplus RvK \oplus h(T_i)$.

Au sujet de l'échange de clé entre les utilisateurs $(i,\ j)$ et la tête de groupe $GH_i$, nous nous sommes appuyés sur des travaux plus génériques de cryptographie basée sur les polynômes de Chebyshev entre un serveur et des utilisateurs [36]. Comme pour la génération de clé de session, la génération de clé intragroupe comporte aussi une phase d'authentification. Cette phase est présentée en haut de la ligne pointillée à la figure 4. Elle ne comporte que deux messages et présente une authentification à double sens à l'aide des

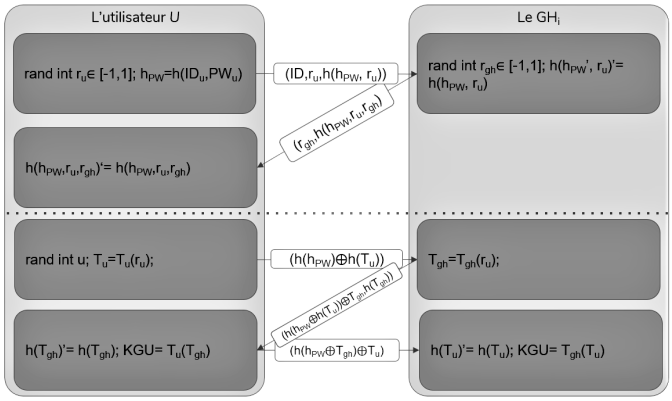


Fig. 4. Échange de clés entre les utilisateurs et la tête de groupe.

identifiants et des mots de passe des utilisateurs et des têtes de groupe. L'authentification est présentée plus en détail à la section VII-C.

Un fois l'utilisateur identifié et après qu'il ait identifié l'autre utilisateur, il peut initier la génération de clé intragroupe à l'aide du nombre entre -1 et 1 qu'il a généré. Cette phase comporte trois messages qui servent à envoyer les polynômes de Chebychev composant la clé. Aucun des messages de la figure 4 est chiffré et les polynômes de Chebychev composant la clé ne sont jamais envoyés en clair sur le réseau, mais peuvent être récupérés à l'aide de la propriété des semis-groupes. La clé créée vaut donc: $KGU_{i,j} = T_{gh}(T_u(r_{U_{i,j}})) = T_u(T_{gh}(r_{U_{i,j}}))$. La génération de clé intragroupe est présentée avec plus de détail à la section VII-D.

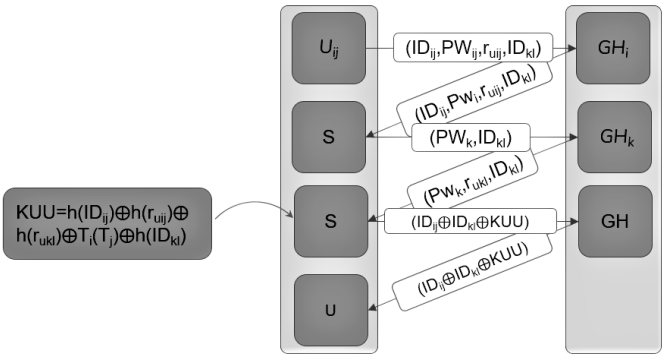


Fig. 5. Échange de clés entre les utilisateurs.

Il faut maintenant avoir un protocole pour générer une clé de chiffrement pour les communications entre les utilisateurs. Le protocole illustré dans la figure 5 permet d'établir la clé entre l'utilisateur $(i,\ j)$, sous le $GH_i$ et l'utilisateur $(k,\ l)$ sous le $GH_k$. Ce dernier est présenté avec plus de détail à la section VII-E. Tous les messages envoyés qui apparaissent dans la figure 5 sont chiffrés à l'aide des clés de session et intragroupes respective. La clé finalement établie par le serveur est: $KU_{i,j}U_{k,l} = h(ID_{U_{i,j}}) \oplus h(r_{U_{i,j}}) \oplus h(r_{U_{k,l}}) \oplus T_i(T_j) \oplus h(ID_{k,l})$. Après avoir été générée, elle est envoyée aux deux GH concernés puis aux utilisateurs $U_{i,j}$ et $U_{k,l}$. Cette génération de clé est fortement inspirée de l'article de Mahmood et al [8].

Cette clé intergroupe est générée sur demande de communication entre utilisateurs pour minimiser la consommation totale d'énergie par le réseau.

### *F. Cryptographie post-quantique*

Actuellement, peu d'auteurs s'intéressent aux polynômes de Chebyshev comme solution de chiffrement post-quantique. Les

chercheurs focalisent leur attention sur les chiffrements post-quantiques suivants [29] [30]:

1) Les réseaux euclidiens: *Lattice-based cryptography*
2) Les codes linéaires aléatoires: *Code-based cryptography*
3) L'inversion de polynômes multivariés: *Multivariate polynomial cryptography*
4) Les arbres de hachage: *Hash-based signatures*
5) L'isogénie en utilisant des courbes supersingulières

G. Brands et C.B. Roellgen [31] se sont intéressés à l'utilisation des ordinateurs quantiques pour retrouver l'ordre des polynômes de Chebychev. Leur approche essaie de déterminer la valeur de l'ordre $n$ de $T_n(x) = \cos(n \arccos(x))$ en se basant sur la périodicité des fonctions cosinus.

L'objectif des auteurs est de prouver à l'aide d'une démarche mathématique que l'inversion d'une fonction cosinus ne mène pas à des états périodiques qui pourraient permettre l'utilisation de l'algorithme de Shor pour résoudre le problème. Le but est de trouver une valeur de $k$ satisfaisant l'équation suivante :

$$r = \pm\frac{\arccos\left(T_n(x)\right)}{\arccos\left(x\right)} + k\frac{2\pi}{\arccos\left(x\right)} = \pm d + ke$$

Celà revient à résoudre l'équation diophantienne de la forme $ax + by = c$. Cette équation n'a de solution que si $a$, $b$ possèdent un plus grand commun diviseur dont $c$ en est un multiple.

Les chercheurs vont donc prouver à l'aide de deux preuves que les polynômes de Chebychev ne mènent pas qu'à des cas triviaux où l'équation diophantienne est résolvable. Ensuite, ceux-ci s'intéresseront aux approximations des équations diophantiennes pour voir si ces dernières ont un avantage comparativement à une méthode de résolution par force brute.

Pour ce faire, il sera utile de représenter l'équation précédente sous la forme suivante: $\pm[d \cdot M] = k \cdot [e \cdot M] + n \cdot M$ où $M$ est un grand nombre entier. La solution est donc:

$$k = E \cdot D^{-1}(mod(M))$$

où $D \in \{[d \cdot M], [d \cdot M] + 1\}$ et $E \in \{[e \cdot M], [e \cdot M] + 1\}$. Malheureusement l'inverse modulaire est statistiquement bien distribué ce qui fait que la solution ne semble pas avoir d'avantage par rapport à la résolution par force brute.

Ainsi G. Brands et C.B. Roellgen [31] ont prouvé que l'inversion des fonctions cosinus ne mène généralement pas à des états périodiques. Il a premièrement été démontré analytiquement que seules des équations diophantiennes triviales mènent à ces états périodiques. De plus, il convient de souligner que généralement des modules sont utilisés dans la génération des polynômes de Chebychev pour éviter ces cas triviaux. Finalement, les chercheurs ont également démontré qu'il n'est pas possible d'approximer des états périodiques plus rapidement que de manière aléatoire. Ils ont donc prouvé qu'il n'est pas possible d'utiliser l'algorithme de Shor lors d'attaques, car il n'y a pas d'état périodique de manière analytique et leur approximation n'est pas plus rapide qu'une recherche aléatoire.

Nous ne pouvons donc pas assumer pour le moment que le chiffrement à l'aide des CP va résister à l'arrivée des ordinateurs quantiques, mais seulement qu'actuelllement rien n'a encore été découvert pour déchiffrer les CP à l'aide d'outils quantiques notamment l'algorithme de Shor.

## IV. MÉTHODOLOGIE

### A. Simulation

En raison de la difficulté de mettre en place des expériences réelles dans un réseau de capteurs sans fil, la simulation est devenue un outil essentiel pour la recherche dans ce domaine [32]. Plusieurs simulateurs sont recensés dans la littérature notamment NS-2 (*The Network Simulator*), NS-3, Cooja, Castalia, OMNET++, GloMoSim, TOSSIM et Avrora [33].

Ces simulateurs utilisent divers langages de programmation: C++, Java, Python, etc. Parmi ces simulateurs, NS-2 et NS-3 semblent principalement plébiscités par la communauté scientifique. NS-2 a été développé en 1989 et est désormais supporté par le DARPA et la *National Science Foundation*. De son côté, NS-3 a commencé à être développé en 2006. NS-3 est un simulateur réseau d'événements discrets spécialement conçu pour la recherche et les projets académiques.

Afin de réaliser notre étude, nous avons opté pour l'utilisation de NS-3. En effet, NS-3 est écrit en C++, un langage avec lequel nous avons de l'expérience. De plus, l'utilisation de NS-3 nous permet d'utiliser des librairies externes pour implémenter les algorithmes de hachage basés sur le chaos et les algorithmes traditionnels.

Pour cela, nous avons opté pour l'utilisation de la librairie libchaos développée par M. A. Czyzewski pour générer des cartes chaotiques et de la librairie cryptopp pour les algorithmes Diffie-Hellman, SHA256 et AES.

À l'aide de NS-3.30 installé sur une machine Ubuntu, nous avons pu simuler l'environnement représenté dans la figure 2. Cet environnement se compose de plusieurs nœuds mobiles membres d'un groupe communiquant par Wifi avec un point d'accès sans-fil formant la tête de groupe (GH pour *Group Head*). Cette tête de groupe est reliée à un hôte résidant sur le même réseau Ethernet que le serveur.

### B. Métriques de performance

Dans un premier temps, notre objectif est de confirmer que ce système de gestion des clés distribuées à groupes multiples (DMK) est plus efficace dans les réseaux de capteurs que l'utilisation de chiffrement traditionnel à clé publique. Pour cela, nous comparerons la performance des polynômes de Chebyshev dans le système DMK avec celle de Diffie-Hellman (DH).

Autrement dit, pour un environnement E et des critères de performance P fixes, nous étudierons quelle caractéristique C (DH ou CP) offre la meilleure performance [34]:

$$E\ fixe, P\ fixe \Rightarrow P(C1) \leqslant P(C2)$$

Pour évaluer la performance des chiffrements, nous avons sélectionné diverses métriques. Ces métriques sont:

1) La taille des paquets servant à l'établissement des clés de session
2) La mémoire utilisée pour les calculs liés au chiffrement
3) Le temps de latence. Ce temps est défini comme la somme du temps de calcul et du temps de cheminement (envoi-réception)

$$\textit{Temps latence} = \textit{Temps calcul} + \textit{Envoi Réception}$$

4) La consommation d'énergie liée aux calculs

Dans un second temps, nous analyserons l'impact de la topologie sur la performance du modèle DMK. En d'autres termes, pour des critères de performance et des caractéristiques fixes, nous identifierons quel environnement offre la meilleure performance [34]:

$$C\ fixe, P\ fixe \Rightarrow P(E1) \leqslant P(E2)$$

Cette analyse nous permettra de savoir si la performance du système DMK est indépendante du nombre de nœuds et de GH ou si, au contraire, elle est impactée par la topologie du réseau.

### C. Topologies testées

Afin de réaliser notre étude, nous avons simulé plusieurs topologies à l'aide de NS3. La première topologie est une topologie équilibrée dans laquelle le réseau est constitué de cinq têtes de groupes et de cinq utilisateurs par groupe, soit un total de 25 utilisateurs. La seconde topologie est également une topologie équilibrée. Toutefois, le nombre d'utilisateurs total a été augmenté à 27 et le nombre de têtes de groupe a été abaissé pour avoir trois groupes, soit neuf

utilisateurs par groupe. Enfin, la troisième topologie est pour sa part déséquilibrée. Le réseau se compose de cinq têtes de groupe avec un nombre croissant d'utilisateurs pour chaque groupe (total de 25 utilisateurs). La figure 6 résume les caractéristiques de ces trois topologies.

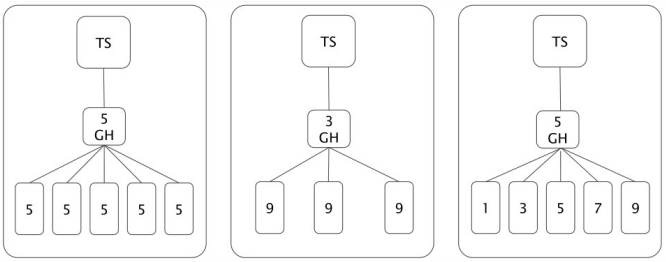


Fig. 6. Caractéristiques des topologies testées.

## V. RÉSULTATS

### A. Polynômes de Chebyshev vs. Diffie-Hellman

Au sujet de la taille des paquets, celle-ci est dépendante de la taille des clés utilisées. En partant de ce constat, il apparaît que cette métrique n'est pas très pertinent pour analyser la performance du chiffrement. Bien que nous étions en mesure d'encoder la clé pour les polynômes de Chebyshev sur 8 octets, une taille nettement inférieure à celle des clés de Diffie-Hellman de 80 octets qui aboutissait sur un paquet de 128 octets, afin de ne pas biaiser nos résultats, nous avons décidé d'utiliser une taille de paquet de 128 octets pour les deux types de chiffrement, ce qui aboutit à un temps d'envoi-réception similaire pour les deux algorithmes.

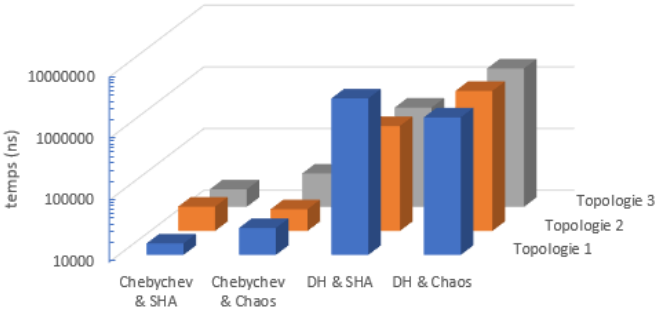


Fig. 7. Latence de préparation.

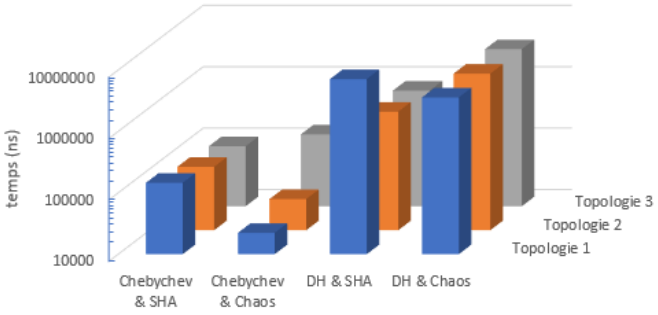


Fig. 8. Latence de réception.

Les figures 7 et 8 présentent respectivement le temps de latence pour la préparation et pour la réception des polynômes de Chebyshev comparativement à Diffie-Hellman et fonction de l'utilisation de SHA256 ou des cartes chaotiques. La latence de préparation est définie comme le temps entre la création d'un paquet et son envoi tandis que la latence de réception correspond au temps entre la réception et le déchiffrement d'un paquet. Ces figures montrent que les polynômes de Chebyshev couplés aux cartes chaotiques offrent une bien meilleure performance que l'utilisation de Diffie-Hellman et de SHA256, et ce pour toutes les topologies testées.

De plus, les chiffres montrent que cette différence est principalement attribuable à l'utilisation des polynômes de Chebyshev. En effet, en général, l'utilisation des cartes chaotiques demande un temps de préparation supérieur à celui nécessaire pour SHA256. Toutefois, cet écart est contrebalancé par une meilleure performance du côté du temps de réception en utilisant les cartes chaotiques.

En ce qui concerne la consommation d'énergie liée aux calculs, nous n'avons malheureusement pas trouvé comment mesurer cette métrique avec NS3. Au sujet de la mémoire utilisée pour les calculs liés au chiffrement, nous avons pu obtenir des données. Ces dernières nous permettent d'avoir une idée de la différence entre l'utilisation des polynômes de Chebyshev et de Diffie-Hellman.

Toutefois, ces données ne sont qu'une approximation de la consommation de mémoire pour réaliser les calculs de clé. En effet, nous avons extrait la mémoire vive (RAM) du processus courant (*/proc/self/status*). Nous avons également utilisé les données accessibles à l'aide du module Memcheck de Valgrind. Les tables III et IV présentent ces résultats.

TABLE III
RAM (EN KB) UTILISÉE POUR LE CALCUL DES CLÉS CÔTÉ GH - PROCESSUS COURANT.

| | 1GH | 2GH | 3GH | 4GH | 5GH | Avg |
|---|---|---|---|---|---|---|
| DH (Avg) | 65496 | 65252 | 65480 | 65052 | 65500 | 65356 |
| CP (Avg) | 65248 | 64904 | 65232 | 65572 | 64920 | 65175 |
| DH - CP | 248 | 348 | 248 | -520 | 580 | 180,8 |

TABLE IV
OCTETS ALLOUÉS - MEMCHECK

| | Topologie | Octets alloués |
|---|---|---|
| CP + SHA | 1 | 8116801 |
| CP + SHA | 2 | 8106936 |
| CP + SHA | 3 | 8117974 |
| CP + cartes chaotiques | 1 | 8196534 |
| CP + cartes chaotiques | 2 | 8196098 |
| CP + cartes chaotiques | 3 | 8204886 |
| DH + SHA | 1 | 222895143 |
| DH + SHA | 2 | 222913347 |
| DH + SHA | 3 | 222840568 |
| DH + cartes chaotiques | 1 | 223061481 |
| DH + cartes chaotiques | 2 | 223137849 |
| DH + cartes chaotiques | 3 | 222973240 |

Nos données montrent que l'utilisation des polynômes de Chebyshev est moins gourmande en termes de mémoire que l'utilisation de Diffie-Hellman. Soulignons toutefois que l'usage des cartes chaotiques pour la fonction de hachage nécessite légèrement plus de mémoire que l'utilisation de SHA256.

### B. Impact de la topologie

Les figures 9 et 10 illustrent le temps de latence de préparation et de réception au niveau des utilisateurs, des têtes de groupe et du serveur de confiance en fonction des différentes topologies étudiées

pour l'utilisation des polynômes de Chebyshev et des cartes chaotiques. Les résultats montrent que le temps de latence de préparation et de réception est bien plus élevé au niveau des têtes de groupes que sur le serveur de confiance et au niveau des utilisateurs. Les résultats montrent également que le temps de latence de préparation et de réception est dépendant de la topologie.

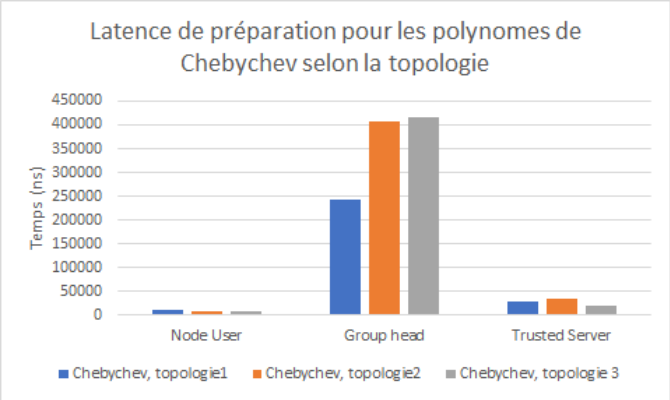


Fig. 9. Latence de préparation.

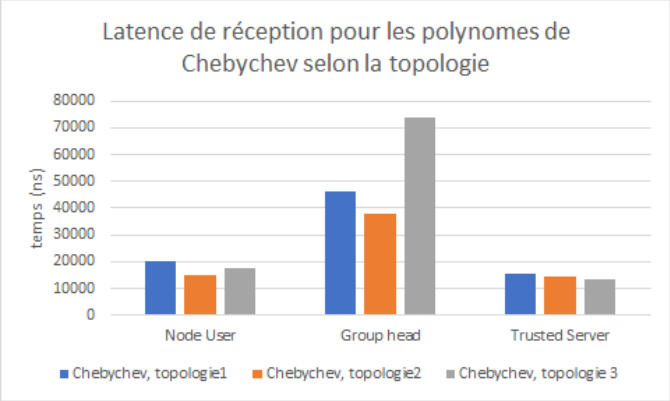


Fig. 10. Latence de réception.

Les résultats agrégés présentés dans la figure 11 confirment que la topologie a un impact significatif sur la performance de notre modèle. En effet, plus il y a d'utilisateurs dans le réseau (topologie 1 en comparaison à la topologie 2) et plus le temps de latence sera élevé. En outre,les résultats montrent aussi que l'équilibre de la topologie a un impact déterminant sur la performance des polynômes de Chebyshev. Une topologie balancée (topologie 1) offre une performance nettement supérieure qu'une topologie non balancée (topologie 3).

## VI. CONCLUSION

À travers notre étude, nous avons pu démontrer que, tel que l'affirmait Mahmood et al. [8], l'utilisation des polynômes de Chebyshev et des cartes chaotiques pour l'authentification et l'échange de clés au sein des réseaux cyber physiques offrent une meilleure performance que les algorithmes traditionnels, notamment Diffie-Hellman. Notre étude montre également que la topologie du réseau a un impact non négligeable sur la performance de ce type de chiffrement.

Toutefois, notre étude reste encore actuellement partiellement achevée. En effet, nous avons eu de la difficulté à mesurer la mémoire utilisée pour les calculs liés au chiffrement. En outre, à l'heure de

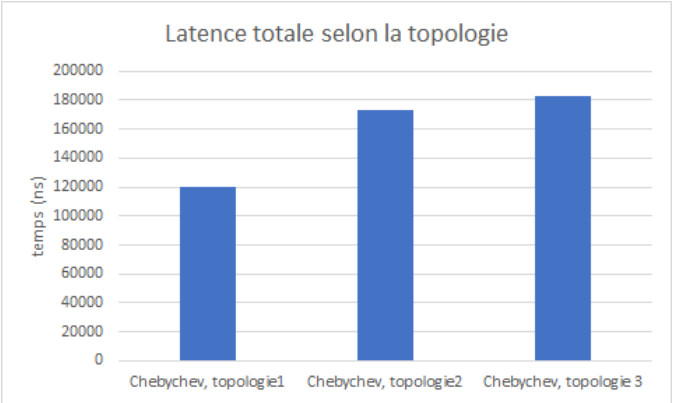


Fig. 11. Latence totale.

rédiger ses lignes, nous n'avons toujours pas trouvé comment extraire des métriques pour la consommation d'énergie liée au calcul des clés. L'étude de ces deux métriques constitue à n'en pas douter un axe de recherche future.

De plus, il convient de souligner qu'en raison des délais qui nous étaient imposés pour la livraison de cet article, nous n'avons pas pu implémenter la troisième et dernière étape du modèle DMK, par manque de temps. Il serait également intéressant d'étudier les métriques relatifs à cette étape dans une future recherche.

Enfin, soulignons que l'architecture distribuée à groupes multiples étudiée à certaines limitations. En effet, advenant que le serveur de confiance ne soit pas hébergé dans le nuage et/ou qu'aucun système de redondance ne soit mis en place pour ce serveur, alors le serveur de confiance constitue un point de défaillance unique (*single point of failure*). Il en est de même en ce qui concerne les têtes de groupes qui représentent des points de défaillances uniques pour les utilisateurs.

## REFERENCES


[1] C. Li, "Security of Wireless Sensor Networks: Current Status and Key Issues," Smart Wireless Sensor Networks, Dec. 2010.

[2] M. El-hajj, A. Fadlallah, M. Chamoun, and A. Serhrouchni, "A Survey of Internet of Things (IoT) Authentication Schemes," Sensors, vol. 19, no. 5, p. 1141, Jan. 2019.

[3] Xia, F., Ma, L., Dong, J., & Sun, Y. (2008, July). Network QoS management in cyber-physical systems. In 2008 International Conference on Embedded Software and Systems Symposia (pp. 302-307). IEEE.

[4] N. Abd El-mawla, M. Badawy, and H. Arafat, "Security and Key Management Challenges over WSN (a Survey)," IJCSES, vol. 10, no. 01, pp. 15–34, Feb. 2019.

[5] T. T. Chakavarika, S. K. Gupta, and B. K. Chaurasia, "Energy Efficient Key Distribution and Management Scheme in Wireless Sensor Networks," Wirel. Pers. Commun., vol. 97, no. 1, pp. 1059–1070, Nov. 2017.

[6] W. Wen, "Energy Efficient Secure Key Management Schemes for WSNs and IoT," p. 143.

[7] Z. Mahmood, H. Ning, and A. Ghafoor, "A Polynomial Subset-Based Efficient Multi-Party Key Management System for Lightweight Device Networks," Sensors, vol. 17, no. 4, p. 670, Mar. 2017.

[8] Mahmood, Z., Ullah, A., and Ning, H. (2018). Distributed Multiparty Key Management for Efficient Authentication in the Internet of Things. IEEE Access, 6, 29460-29473.

[9] D. Mall, K. Konaté, and A.-S. K. Pathan, "ECL-EKM: An enhanced Certificateless Effective Key Management protocol for dynamic WSN," in 2017 International Conference on Networking, Systems and Security (NSysS), 2017, pp. 150–155.

[10] S. Athmani, A. Bilami, and D. E. Boubiche, "EDAK: An Efficient Dynamic Authentication and Key Management Mechanism for heterogeneous WSNs," Future Generation Computer Systems, vol. 92, pp. 789–799, Mar. 2019.

[11] S.-H. Seo, J. Won, S. Sultana, and E. Bertino, "Effective Key Management in Dynamic Wireless Sensor Networks," IEEE Transactions on Information Forensics and Security, vol. 10, no. 2, pp. 371–383, Feb. 2015.
[12] F. I. Kandah, O. Nichols, and Li Yang, "Efficient key management for Big Data gathering in dynamic sensor networks," in 2017 International Conference on Computing, Networking and Communications (ICNC), 2017, pp. 667–671.
[13] D. Solev, P. Janjic, and L. Kocarev, "Introduction to Chaos," in Chaos-Based Cryptography: Theory, Algorithms and Applications, L. Kocarev and S. Lian, Eds. Berlin, Heidelberg: Springer, 2011, pp. 1–25.
[14] D. Xiao, X. Liao, and S. Deng, "Chaos Based Hash Function," in Chaos-Based Cryptography: Theory, Algorithms and Applications, L. Kocarev and S. Lian, Eds. Berlin, Heidelberg: Springer, 2011, pp. 137–203.
[15] H. S. Kwok, W. K. S. Tang, "A Chaos-Based Cryptographic Hash Function For Message Authentification," International Journal of Bifurcation and Chaos, vol. 15, no. 12, pp. 4043-4050, 2005
[16] Q. Wu, "A Chaos-Based Hash Function," in 2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, 2015, pp. 1–4.
[17] M. Amin, O. S. Faragallah, and A. A. Abd El-Latif, "Chaos-based hash function (CBHF) for cryptographic applications," Chaos, Solitons and Fractals, vol. 42, no. 2, pp. 767–772, Oct. 2009.
[18] M. T. Mohammed, A. E. Rohiem, A. El-moghazy, and A. Z. Ghalwash, "Chaotic Based Secure Hash Algorithm," IJETTCS, vol. 2, no. 2, p. 7, 2013.
[19] M. A. Czyzewski, "Chaos Machine: Different Approach to the Application and Significance of Numbers," Cryptology ePrint Archive, Report 2016/468.
[20] Lian, S.G., Sun, J. and Wang, Z., Security analysis of a chaos-based image encryption algorithm. Physica A. v351. 645-661.
[21] L. Kocarev, J. Makraduli, P. Amato, "Public-Key Encryption Based on Chebyshev Polynomials," Circuits Syst Signal Process, vol. 24, no. 5, 2005.
[22] S. Vairachilai, M. K. Kavithadevi, and R. Gnanajeyaraman, "Public Key Cryptosystems Using Chebyshev Polynomials Based on Edge Information," in 2014 World Congress on Computing and Communication Technologies, 2014, pp. 243–245.
[23] G. J. Fee and M. B. Monagan, "Cryptography using Chebyshev polynomials," p. 15.
[24] C. Meshram, M. S. Obaidat, and S. G. Meshram, "Chebyshev chaotic map-based ID-based cryptographic model using subtree and fuzzy-entity data sharing for public key cryptography," Security and Privacy, vol. 1, no. 1, p. e12, 2018.
[25] G. Gao, X. Peng, Y. Tian, and Z. Qin, "A Chaotic Maps-Based Authentication Scheme for Wireless Body Area Networks," International Journal of Distributed Sensor Networks, vol. 12, no. 7, p. 2174720, Jul. 2016
[26] H. Zhu, "Cryptanalysis and provable improvement of a chaotic maps-based mobile dynamic ID authenticated key agreement scheme," Security and Communication Networks, vol. 8, no. 17, pp. 2981–2991, 2015.
[27] T.-T. Truong, M.-T. Tran, and A.-D. Duong, "Improved Chebyshev Polynomials-Based Authentication Scheme in Client-Server Environment," Security and Communication Networks, 2019.
[28] L. Zhang, "Cryptanalysis of the public key encryption based on multiple chaotic systems," Chaos, Solitons and Fractals, vol. 37, no. 3, pp. 669–674, Aug. 2008.
[29] L. Chen et al., " Report on Post-Quantum Cryptography ", National Institute of Standards and Technology, NIST IR 8105, avr. 2016.
[30] K. Basu, D. Soni, M. Nabeel, et R. Karri, " NIST Post-Quantum Cryptography- A Hardware Evaluation Study ", p. 16.
[31] G. Brands et C. B. Roellgen, " QRKE: Quantum-Resistant Public Key Exchange - The RVB algorithm ", p. 27.
[32] M. P. Chhimwal, D. S. Rai, and D. Rawat, "Comparison between Different Wireless Sensor Simulation Tools," IOSR Journal of Electronics and Communication Engineering (IOSR-JECE), vol. 5, no. 2, pp 54-60, Mar.-Apr. 2013.
[33] A. Diaz and P. Sanchez, "Simulation of Attacks for Security in Wireless Sensor Network," Sensors (Basel), vol. 16, no. 11, Nov. 2016.
[34] J. M. Fernandez and P. Bureau, "Optimising Malware," in 2006 IEEE International Performance Computing and Communications Conference, Phoenix, AX, USA, 2006, pp. 577–586.
[35] Lee, C. C., Li, C. T., and Hsu, C. W. (2013). A three-party password-based authenticated key exchange protocol with user anonymity using extended chaotic maps. Nonlinear Dynamics, 73(1-2), 125-132.
[36] Guo, X., and Zhang, J. (2010). Secure group key agreement protocol based on chaotic Hash. Information Sciences, 180(20), 4069-4074.
[37] Fee, G. J., and Monagan, M. B. (2004, July). Cryptography using Chebyshev polynomials. In Proc. 2004 Maple Summer Workshop (Vol. 5, No. 10).
[38] Geisel, T., and Fairen, V. (1984). Statistical properties of chaos in Chebyshev maps. Physics Letters A, 105(6), 263-266.

## VII. Annexe

### *A. Authentification des têtes de groupe et du serveur de confiance*

**1** Le serveur S

**1.1** Génération d'un entier aléatoire $s$ et de $x \in [-1, 1]$.

**1.2** Création d'une clé publique $Q$.

$$Q = T_s(x)$$

**1.3** Envoi du message $m_1$ à tous les GH.

$$m_1 = [Q, x, t_s, h(Q, x, t_s)]$$

**2** Le $\text{GH}_i$

**2.1** Réception et déchiffrement du message $m_1$ de S.

**2.2** Vérification de l'intégrité du message $m_1$.

$$h'(Q, x, t_s) \stackrel{?}{=} h(Q, x, t_s)$$
$$\Delta t \stackrel{?}{\geq} t_{now} - t_s$$

**2.3** Génération d'un entier aléatoire $g$.

**2.4** Calcul des paramètres suivants :

$$R_i = T_g(x)$$
$$T_i = T_g(Q) = T_g(T_s(x))$$
$$AID = ID_i \oplus h(T_i)$$
$$\tau_i = h(ID_i, AID, PW_i, T_i)$$

**2.5** $\text{GH}_i$ envoie le message $m_2$ à S.

$$m_2 = [ID_i, AID, R_i, \tau_i, t_s, h(ID_i, AID, R_i, \tau_i, t_s]$$

**3** Le serveur S

**3.1** Réception et déchiffrement du message $m_2$ de $\text{GH}_i$.

**3.2** Vérification de l'intégrité du message $m_2$.

$$h'(ID_i, AID, R_i, \tau_i, t_s) \stackrel{?}{=} h(ID_i, AID, R_i, \tau_i, t_s)$$
$$\Delta t \stackrel{?}{\geq} t_{now} - t_s$$

**3.3** Calcul des paramètres suivants :

$$T_i' = T_s(R_i) = T_s(T_g(x))$$
$$AID' = ID_i \oplus h(T_i')$$
$$\tau_i' = h(ID_i, AID', PW_i, T_i')$$

**3.4** Validation des paramètres :

$$AID \stackrel{?}{=} AID'$$
$$\tau_i \stackrel{?}{=} \tau_i'$$

**3.5** Calcul d'un nouveau paramètre :

$$\tau_{s,i} = h(ID_i, h(PW_i \oplus T_i'))$$

**3.6** S envoie le message $m_3$ à $GH_i$.

$$m_3 = [\tau_{s,i}, t_s, h(\tau_{s,i}, t_s]$$

**4** Le $GH_i$

**4.1** Réception et déchiffrement du message $m_3$ de S.
**4.2** Vérification de l'intégrité du message $m_3$.

$$h'(\tau_{s,i}, t_s) \stackrel{?}{=} h(\tau_{s,i}, t_s)$$
$$\Delta t \stackrel{?}{\geq} t_{now} - t_s$$

**4.3** $GH_i$ calcule le paramètre suivant :

$$\tau'_{s,i} == h(ID_i, h(PW_i \oplus T_i))$$

**4.4** Validation du paramètre $\tau_{s,i}$.

$$\tau_{s,i} \stackrel{?}{=} \tau'_{s,i}$$

*B. Création de clé de session entre les têtes de groupe et le serveur de confiance*

**1** Le serveur S

**1.1** Génération d'un naturel aléatoire $K$.
**1.2** Génération de $RvK$.

$$RvK = h(T_1) \oplus h(T_2) \oplus ... \oplus h(T_K)$$

**1.3** Envoi du message $m_1$ à $GH_i$.

$$m_1 = [h(PW_i) \oplus h(T_i) \oplus RvK, t_s, h(h(PW_i) \oplus h(T_i) \oplus RvK, t_s)]$$

**2** Le $GH_i$

**2.1** Réception et déchiffrement du message $m_1$ de S.
**2.2** Vérification de l'intégrité du message $m_1$.

$$h'(h(PW_i) \oplus h(T_i) \oplus RvK, t_s) \stackrel{?}{=} h(h(PW_i) \oplus h(T_i) \oplus RvK, t_s)$$
$$\Delta t \stackrel{?}{\geq} t_{now} - t_s$$

**2.3** Extraction du paramètre $RvK$.

$$RvK' = h(T_i) \oplus (h(PW_i) \oplus (h(PW_i) \oplus h(T_i) \oplus RvK))$$

**2.4** Création de la clé de chiffrement symétrique.

$$KGS_i = h(ID_i) \oplus h(PW_i) \oplus RvK \oplus h(T_i)$$

**2.5** $GH_i$ envoie le message $m_2$ à S.

$$m_2 = E_{KGSI}[ID_i \oplus h(PW_i) \oplus h(R_i), t_s, h(ID_i \oplus h(PW_i) \oplus h(R_i), t_s)]$$

**3** Le serveur S

**3.1** Génération de la clé de chiffrement symétrique.

$$KGS'_i = h(ID_i) \oplus h(PW_i) \oplus RvK \oplus h(T_i)$$

**3.2** Réception et déchiffrement du message $m_2$ à l'aide de la clé $KGS'_i$.

**3.3** Vérification de l'intégrité du message $m_2$.

$$h'(ID_i \oplus h(PW_i) \oplus h(R_i), t_s) \stackrel{?}{=} h(ID_i \oplus h(PW_i) \oplus h(R_i), t_s)$$
$$\Delta \stackrel{?}{\geq} t_{now} - t_s$$

**3.4** Validation du message.

$$ID'_i \oplus h(PW'_i) \oplus h(R'_i) \stackrel{?}{=} ID_i \oplus h(PW_i) \oplus h(R_i)$$

**3.5** Envoi d'un message de validation à $GH_i$.

$$m_3 = E_{KGS_i}[ID_i \oplus h(PW_i) \oplus h(Q), t_s, h(ID_i \oplus h(PW_i) \oplus h(Q), t_s)]$$

**4** Le $GH_i$

**4.1** Réception et déchiffrement du message $m_3$ à l'aide de la clé $KGS'_i$.
**4.2** Vérification de l'intégrité du message $m_3$.

$$h'(ID_i \oplus h(PW_i) \oplus h(Q), t_s) \stackrel{?}{=} h(ID_i \oplus h(PW_i) \oplus h(Q), t_s)$$
$$\Delta t \geq t_{now} - t_s$$

**4.3** Validation du message.

$$ID'_i \oplus h(PW'_i) \oplus h(Q') \stackrel{?}{=} ID_i \oplus h(PW_i) \oplus h(Q)$$

*C. Authentification des utilisateurs et des têtes de groupe*

**1** L'utilisateur $U_{i,j}$

**1.1** Génération d'un nombre aléatoire $r_{U_{i,j}} \in [-1, 1]$.
**1.2** Calcul du paramètre suivant.

$$h_{PW_{i,j}} = h(ID_{i,j}, PW_{i,j})$$

**1.3** Envoi du message $m_1$ à $GH_i$.

$$m_1 = [ID_{i,j}, r_{U_{i,j}}, h(h_{PW_{i,j}}, r_{U_{i,j}}), t_s, h(ID_{i,j}, r_{U_{i,j}}, h(h_{PW_{i,j}}, r_{U_{i,j}}), t_s)]$$

**2** Le $GH_i$

**2.1** Réception du message $m_1$ de $U_{i,j}$.
**2.2** Vérification de l'intégrité du message $m_1$.

$$h'(ID_{i,j}, r_{U_{i,j}}, h(h_{PW_{i,j}}, r_{U_{i,j}}), t_s) \stackrel{?}{=} h(ID_{i,j}, r_{U_{i,j}}, h(h_{PW_{i,j}}, r_{U_{i,j}}), t_s)$$
$$\Delta t \stackrel{?}{\geq} t_{now} - t_s$$

**2.3** Calcul et validation du paramètre $h(h_{PW_{i,j}}, r_{U_{i,j}})$.

$$h'(h_{PW'_{i,j}}, r_{U_{i,j}}) \stackrel{?}{=} h(h_{PW_{i,j}}, r_{U_{i,j}})$$

**2.4** Génération d'un nombre aléatoire $r_{\mathrm{GH}_{i,j}} \in [-1, 1]$.
**2.5** $GH_i$ envoie le message $m_2$ à $U_{i,j}$.

$$m_2 = [r_{\mathrm{GH}_{i,j}}, h(h_{PW_{i,j}}, r_{U_{i,j}}, r_{\mathrm{GH}_{i,j}}), t_s, h(r_{\mathrm{GH}_{i,j}}, h(h_{PW_{i,j}}, r_{U_{i,j}}, r_{\mathrm{GH}_{i,j}}), t_s)]$$

**3** L'utilisateur $U_{i,j}$

**3.1** Réception du message $m_2$ de $GH_i$.

**3.2** Vérification de l'intégrité du message $m_2$.

$$h'(r_{\text{GH}_{i,j}}, h(h_{PW_{i,j}}, r_{U_{i,j}}, r_{\text{GH}_{i,j}}), t_s) \stackrel{?}{=} h(r_{\text{GH}_{i,j}}, h(h_{PW_{i,j}}, r_{U_{i,j}}, r_{\text{GH}_{i,j}}), t_s)$$
$$\Delta t \geq t_{now} - t_s$$

**3.3** Calcul et validation du paramètre $h(h_{PW_{i,j}}, r_{U_{i,j}}, r_{\text{GH}_{i,j}})$.

$$h'(h_{PW_{i,j}}, r_{U_{i,j}}, r_{\text{GH}_{i,j}}) \stackrel{?}{=} h(h_{PW_{i,j}}, r_{U_{i,j}}, r_{\text{GH}_{i,j}})$$

*D. Création de clés intragroupes entre les utilisateurs et les têtes de groupe*

**1** L'utilisateur $U_{i,j}$

**1.1** Génération d'un entier aléatoire $u$.

**1.2** Calcul du polynôme de Chebychev de $U_{i,j}$

$$T_u(r_{U_{i,j}})$$

**1.3** Envoi du message $m_1$ à $\text{GH}_i$.

$$m_1 = [h(h_{PW_{i,j}}) \oplus h(T_u(r_{U_{i,j}})), t_s, h(h(h_{PW_{i,j}}) \oplus h(T_u(r_{U_{i,j}})), t_s)]$$

**2** Le $\text{GH}_i$

**2.1** Réception du message $m_1$ de $U_{i,j}$.

**2.2** Vérification de l'intégrité du message $m_1$.

$$h'(h(h_{PW_{i,j}}) \oplus h(T_u(r_{U_{i,j}})), t_s) \stackrel{?}{=} h(h(h_{PW_{i,j}}) \oplus h(T_u(r_{U_{i,j}})), t_s)$$
$$\Delta t \stackrel{?}{\geq} t_{now} - t_s$$

**2.3** Extraction du paramètre $h(T_u(r_{U_{i,j}}))$.

$$h(T_u(r_{U_{i,j}})' = h(h'_{PW_{i,j}}) \oplus h(h_{PW_{i,j}}) \oplus h(T_u(r_{U_{i,j}}))$$

**2.4** Génération d'un entier aléatoire $gh$.

**2.5** Calcul du polynôme de Chebychev de $\text{GH}_i$

$$T_{gh}(r_{U_{i,j}})$$

**2.6** $\text{GH}_i$ envoie le message $m_2$ à $U_{i,j}$.

$$m_2 = [h((h_{PW_{i,j}}) \oplus h(T_u(r_{U_{i,j}}))) \oplus T_{gh}(r_{U_{i,j}}), h(T_{gh}(r_{U_{i,j}})), t_s, h(h((h_{PW_{i,j}}) \oplus h(T_u(r_{U_{i,j}}))) \oplus T_{gh}(r_{U_{i,j}}), h(T_{gh}(r_{U_{i,j}})), t_s)]$$

**3** L'utilisateur $U_{i,j}$

**3.1** Réception du message $m_2$ de $\text{GH}_i$.

**3.2** Vérification de l'intégrité du message $m_2$.

$$h'(h((h_{PW_{i,j}}) \oplus h(T_u(r_{U_{i,j}}))) \oplus T_{gh}(r_{U_{i,j}}), h(T_{gh}(r_{U_{i,j}})), t_s) \stackrel{?}{=} h(h((h_{PW_{i,j}}) \oplus h(T_u(r_{U_{i,j}}))) \oplus T_{gh}(r_{U_{i,j}}), h(T_{gh}(r_{U_{i,j}})), t_s)$$
$$\Delta \stackrel{?}{\geq} t_{now} - t_s$$

**3.4** Extraction du polynôme $T_{gh}(r_{U_{i,j}})$.

$$T'_{gh}(r_{U_{i,j}}) = h'((h_{PW_{i,j}}) \oplus h(T_u(r_{U_{i,j}}))) \oplus h((h_{PW_{i,j}}) \oplus h(T_u(r_{U_{i,j}}))) \oplus T_{gh}(r_{U_{i,j}})$$

**3.5** Validation du polynôme $T_{gh}(r_{U_{i,j}})$.

$$h(T'_{gh}(r_{U_{i,j}})) = h(T_{gh}(r_{U_{i,j}}))$$

**3.6** Envoie le message $m_3$ à $\text{GH}_i$.

$$m_3 = [h(h_{PW_{i,j}} \oplus T_{gh}(r_{U_{i,j}})) \oplus T_u(r_{U_{i,j}}), t_s, h(h(h_{PW_{i,j}} \oplus T_{gh}(r_{U_{i,j}})) \oplus T_u(r_{U_{i,j}}), t_s)]$$

**4** Le $\text{GH}_i$

**4.1** Réception du message $m_3$ de $U_{i,j}$.

**4.2** Vérification de l'intégrité du message $m_3$.

$$h'(h(h_{PW_{i,j}} \oplus T_{gh}(r_{U_{i,j}})) \oplus T_u(r_{U_{i,j}}), t_s) \stackrel{?}{=} h(h(h_{PW_{i,j}} \oplus T_{gh}(r_{U_{i,j}})) \oplus T_u(r_{U_{i,j}}), t_s)$$
$$\Delta t \geq t_{now} - t_s$$

**4.3** Extraction du polynôme $T_u(r_{U_{i,j}})$.

$$T'_u(r_{U_{i,j}}) = h'((h_{PW_{i,j}}) \oplus h(T_{gh}(r_{U_{i,j}}))) \oplus h((h_{PW_{i,j}}) \oplus h(T_{gh}(r_{U_{i,j}}))) \oplus T_u(r_{U_{i,j}})$$

**4.4** Validation du polynôme $T_u(r_{U_{i,j}})$.

$$h(T'_u(r_{U_{i,j}})) = h(T_{gh}(r_{U_{i,j}}))$$

**5** Le $\text{GH}_i$ et $U_{i,j}$

**5.1** Calcul de la clé de chiffrement symétrique.

$$KGU_{i,j} = T_{gh}(T_u(r_{U_{i,j}})) = T_u(T_{gh}(r_{U_{i,j}}))$$

*E. Création de clés inter-groupes entre deux utilisateurs*

**1** L'utilisateur $U_{i,j}$

**1.1** $U_{i,j}$ envoie le message $m_1$ à $\text{GH}_i$.

$$m_1 = E_{KGU_{i,j}}[ID_{i,j}, PW_{i,j}, r_{U_{i,j}}, ID_{k,l}, t_s, \text{MAC}(ID_{i,j}, PW_{i,j}, r_{U_{i,j}}, ID_{k,l}, t_s)]$$

**2** Le $\text{GH}_i$

**2.1** Réception du message $m_1$ et déchiffrement à l'aide de la clé $KGU_{i,j}$.

**2.2** Vérification de l'intégrité du message $m_1$.

$$\text{MAC}'(ID_{i,j}, PW_{i,j}, r_{U_{i,j}}, ID_{k,l}, t_s) \stackrel{?}{=} \text{MAC}(ID_{i,j}, PW_{i,j}, r_{U_{i,j}}, ID_{k,l}, t_s)$$
$$\Delta \stackrel{?}{\geq} t_{now} - t_s$$

**2.3** Envoi du message $m_2$ au serveur S.

$$m_2 = E_{KGS_i}[ID_{i,j}, PW_i, r_{U_{i,j}}, ID_{k,l}, t_s, \text{MAC}(ID_{i,j}, PW_i, r_{U_{i,j}}, ID_{k,l}, t_s)]$$

**3** Le serveur S

**3.1** Réception du message $m_2$ et déchiffrement à l'aide de la clé $KGS_i$.

**3.2** Vérification de l'intégrité du message $m_2$.

$$\text{MAC}'(ID_{i,j}, PW_i, r_{U_{i,j}}, ID_{k,l}, t_s) \stackrel{?}{=} \text{MAC}(ID_{i,j}, PW_i, r_{U_{i,j}}, ID_{k,l}, t_s)$$
$$\Delta \stackrel{?}{\geq} t_{now} - t_s$$

**3.3** Envoi du message $m_3$ à GH$_k$.

$$m_3 = E_{KGS_k}[PW_k, ID_{k,l}, t_s, \text{MAC}(PW_k, , ID_{k,l}, t_s)]$$

**4** Le GH$_k$

**4.1** Réception du message $m_3$ et déchiffrement à l'aide de la clé $KGS_k$.

**4.2** Vérification de l'intégrité du message $m_3$.

$$\text{MAC}'(PW_k, ID_{k,l}, t_s) \stackrel{?}{=} \text{MAC}(PW_k, ID_{k,l}, t_s)$$
$$\Delta \stackrel{?}{\geq} t_{now} - t_s$$

**4.3** Envoi du message $m_4$ à S.

$$m_4 = E_{KGS_k}[PW_k, r_{U_{k,l}}, ID_{k,l}, t_s, \text{MAC}(PW_k, r_{U_{k,l}}, ID_{k,l}, t_s)]$$

**5** Le serveur S.

**5.1** Réception du message $m_4$ et déchiffrement à l'aide de la clé $KGS_k$.

**5.2** Vérification de l'intégrité du message $m_4$.

$$\text{MAC}'(PW_k, r_{U_{k,l}}, ID_{k,l}, t_s) \stackrel{?}{=} \text{MAC}(PW_k, r_{U_{k,l}}, ID_{k,l}, t_s)$$
$$\Delta \stackrel{?}{\geq} t_{now} - t_s$$

**5.2** Calcul de la clé $KU_{i,j}U_{k,l}$.

$$KU_{i,j}U_{k,l} = h(ID_{U_{i,j}}) \oplus h(r_{U_{i,j}}) \oplus h(r_{U_{k,l}}) \oplus T_i(T_j) \oplus h(ID_{k,l})$$

**5.4** Envoi du message $m_5$ à GH$_i$ et GH$_k$.

$$m_{5_i} = E_{KGS_i}[ID_{i,j}, ID_{k,l}, KU_{i,j}U_{k,l}, t_s, h(ID_{i,j}, ID_{k,l}, KU_{i,j}U_{k,l}, t_s)]$$
$$m_{5_k} = E_{KGS_k}[ID_{k,l}, ID_{i,j}, KU_{i,j}U_{k,l}, t_s, h(ID_{k,l}, ID_{i,j}, KU_{i,j}U_{k,l}, t_s)]$$

**6** Les GH$_i$ et GH$_k$.

**6.1** Réception du message $m_5$ et déchiffrement à l'aide des clés $KGS_i$ ou $KGS_k$.

**6.2** Vérification de l'intégrité du message $m_5$.

$$h'(ID_{i,j}, ID_{k,l}, KU_{i,j}U_{k,l}, t_s) \stackrel{?}{=} h(ID_{i,j}, ID_{k,l}, KU_{i,j}U_{k,l}, t_s)$$
$$h'(ID_{k,l}, ID_{i,j}, KU_{i,j}U_{k,l}, t_s) \stackrel{?}{=} h(ID_{k,l}, ID_{i,j}, KU_{i,j}U_{k,l}, t_s)$$
$$\Delta \stackrel{?}{\geq} t_{now} - t_s$$

**6.3** Envoi du message $m_6$ à $U_{i,j}$ et $U_{k,l}$.

$$m_{6_i} = E_{KGU_{i,j}}[ID_{i,j}, ID_{k,l}, KU_{i,j}U_{k,l}, t_s, h(ID_{i,j}, ID_{k,l}, KU_{i,j}U_{k,l}, t_s)]$$
$$m_{6_k} = E_{KGU_{k,l}}[ID_{k,l}, ID_{i,j}, KU_{i,j}U_{k,l}, t_s, h(ID_{k,l}, ID_{i,j}, KU_{i,j}U_{k,l}, t_s)]$$

**7** Les $U_{i,j}$ et $U_{k,l}$.

**7.1** Réception du message $m_6$ et déchiffrement à l'aide des clés $KGU_{i,j}$ ou $KGU_{k,l}$.

**7.2** Vérification de l'intégrité du message $m_6$.

$$h'(ID_{i,j}, ID_{k,l}, KU_{i,j}U_{k,l}, t_s) \stackrel{?}{=} h(ID_{i,j}, ID_{k,l}, KU_{i,j}U_{k,l}, t_s)$$
$$h'(ID_{k,l}, ID_{i,j}, KU_{i,j}U_{k,l}, t_s) \stackrel{?}{=} h(ID_{k,l}, ID_{i,j}, KU_{i,j}U_{k,l}, t_s)$$
$$\Delta \stackrel{?}{\geq} t_{now} - t_s$$