MajecSTIC 2009
Avignon, France, du 16 au 18 novembre 2009

Sécurité des Calculs Distribués Multiparties Application : Sécuriser le Calcul Distribué des Confiances.

Boussad Ait-salem[1]

1 : Université de Limoges, Laboratoire Xlim, 83, rue d'Isle, 87000 Limoges - France.

Contact : boussad.ait-salem@unilim.fr

Résumé
Le problème de la sécurité des calculs distribués entre plusieurs parties est de permettre à ces différentes parties de réaliser des calculs, basés sur leurs données privées $(p_1, p_2, ..., p_n)$, de manière à ce que tout le monde connaisse le résultat final du calcul : $f(p_1, p_2, ..., p_n)$, mais aucune partie ne puisse déduire les données privées des autres. Dans cet article, nous allons tout d'abord présenter quelques notions générales et quelques primitives cryptographiques de base utilisées dans ce domaine. Ensuite, nous aborderons le cas de la sécurité du produit scalaire distribué appliqué aux modèles de confiance. A cet effet, nous proposerons un schéma indépendant des applications et de la topologie des réseaux. Basé sur des primitives cryptographiques dont la sécurité a été prouvée, notre modèle offre un moyen sûr et efficace permettant d'assurer le calcul des confiances tout en préservant la confidentialité des données privées des différentes entités qui interviennent dans ce calcul.

Mots-clés : Modèles de confiance, Privacy-Preserving computation of trust, Produit scalaire distribué, Secure Multi party Computation (SMC).

1. Introduction

Avec la prolifération des applications et des données traitées sur internet, les calculs coopératifs, où chaque partie met en jeu ses propres informations, deviennent de plus en plus fréquents. Les parties engagées dans ce genre de calcul peuvent se faire confiance mutuellement, comme elles peuvent, au contraire, se méfier les unes des autres ou bien même être des concurrents économiques directs. En règle générale, il arrive fréquemment que plusieurs parties (personnes, entreprises, etc.) coopèrent, dans l'intérêt de tout le monde, afin de résoudre un problème bien particulier. Pour le bon déroulement de ces calculs, il est nécessaire que chaque partie connaisse les données privées des autres. Mais si personne ne fait confiance à l'autre ou s'il n'existe aucune partie jouissant de la confiance de toutes les autres parties, la confidentialité de ces données devient un problème crucial.
La problématique liée à la sécurité des calculs distribués entre plusieurs parties (SMC -Secure Multi-party Computation problem) [12, 13, 19] est de permettre le calcul d'une fonction quelconque sur un ensemble de données réparties entre plusieurs entités. Chaque entité possède une partie des données et le calcul doit être réalisé de manière à ce qu'aucune des parties ne puisse déduire de quelque manière que ce soit les données des autres entités à partir des résultats du calcul et de ses propres données. A. C. Yao [19] a été le premier à introduire le problème de la sécurité des protocoles faisant intervenir deux parties. Ensuite, les travaux comme ceux de [7,13] ont généralisé le problème à un nombre quelconque de parties. Dans [7,12], les auteurs ont prouvé que de tels calculs sécurisés peuvent être effectués pour toute fonction polynomiale. Dans leur approche, ils utilisent le fait qu'à toute fonction polynomiale, un circuit binaire réalisant cette fonction peut être associé. Par le passé, le SMC a attiré de nombreux chercheurs, et maintenant il est devenu essentiel dans de nombreux domaines de l'informatique : bases de données, calculs scientifiques, data mining, fouille de données, intégration de données, modèles de confiance, systèmes de détection d'intrusions, etc.
Dans cet article, nous allons nous focaliser sur un problème bien particulier qui est celui de la

préservation des données privées engagées dans le calcul des valeurs de confiance dans un modèle basé sur la réputation. Effectivement, un des concepts importants en sécurité réseau est la notion de confiance. La confiance est interprétée comme étant l'ensemble des relations existant entre les entités participant à un protocole donné [10]. La relation de confiance entre une paire de noeuds est souvent basée sur le résultat de leurs interactions passées. Le mode d'évaluation de la confiance varie d'une application à une autre, tandis que la façon d'utiliser cette confiance, afin de répondre au bon déroulement du protocole, est déterminée par les différents participants au protocole. Dans cet article, nous allons nous intéresser à une catégorie bien particulière des modèles de confiance, qui est largement utilisée dans les modèles de communication décentralisés, à savoir les systèmes basés sur la réputation ou aussi appelés les systèmes à base de recommandations [4, 15, 21]. Dans ce type de modèles, la confiance d'une entité est calculée en tenant compte, en plus du comportement et des expériences passées des entités, des recommandations qu'elles s'échangent entre elles. Ces modèles offrent un mécanisme flexible et dynamique pour l'établissement de la confiance et s'appliquent bien aux systèmes distribués tel que les réseaux pair à pair (réseaux ad-hoc), les applications de commerce en ligne (e-commerce), ou bien aux systèmes de partage de ressources.

Dans les modèles basés sur la réputation, les recommandations échangées concernant la fiabilité d'un utilisateur sont souvent supposées être publiques. Cependant, ces recommandations représentent les opinions personnelles des uns et des autres, et sont parfois considérées comme étant sensibles. Par exemple, Bob a connu une mauvaise expérience avec Paul durant une vente aux enchères en ligne, mais Bob ne veut pas divulguer cette mauvaise expérience pour ne pas que Paul se venge et lui fasse la même chose une autre fois. Alice, qui n'a pas encore effectué de transactions avec Paul, veut utiliser les recommandations de Bob ainsi que celles des autres utilisateurs du site afin d'évaluer le degré de fiabilité de Paul. D'un autre côté, Alice a sa propre opinion de Bob et des autres, ce qui lui permettra d'affiner son jugement à propos de Paul. Le problème qu'on veut résoudre dans cet article est de donner un moyen sécurisé qui permettra à Alice de calculer la confiance de Paul sans que personne ne divulgue ses paramètres sensibles durant le calcul (C'est-à-dire que les recommandations des uns et des autres sont gardées secrètes).

Ce problème peut être formalisé comme le fait de sécuriser un calcul distribué d'un produit scalaire faisant intervenir plusieurs parties. Effectivement, si par exemple Alice veut calculer la valeur V de la confiance qu'elle accorde à Bob en utilisant les recommandations de Nicolas, Jacques et François dont les valeurs sont notées respectivement y_1, y_2, y_3, tout en sachant qu'elle a sa propre opinion à propos des trois "conseillers" (resp. x_1, x_2, x_3), elle aura à calculer simplement le produit scalaire des deux vecteurs (y_1, y_2, y_3) et (x_1, x_2, x_3) c'est-à-dire $V = \sum_{i=1}^{3} x_i y_i$. C'est la méthode la plus utilisée dans de nombreux modèles de confiance. Pour une définition plus précise du problème voir section 4.

1.1. Objectifs

Dans cet article, nous allons essayer de résoudre le problème de la sécurité du produit scalaire appliqué à la préservation de la confidentialité des données privées des différentes parties engagées dans le calcul des confiances. Le problème est, pour Alice, notée A, d'évaluer la confiance d'une entité qu'elle ne connaît pas encore notée E en se basant sur ce que pensent les autres entités notées par $(B_1, B_2, ..., B_n)$ de E et aussi sur ce que pense A de ces entités.

Ce problème a été résolu par Yao et *al.* dans [20] en calculant le produit scalaire de deux vecteurs :

– Le premier vecteur représente les confiances que A accorde aux autres entités (les B_i) ;

– Le second vecteur représente les recommandations des autres entités (les B_i) au sujet de E.

Yao et *al.* ont basé leurs travaux sur ceux effectués auparavant dans le cas de la sécurisation des produits scalaires à deux parties [1, 11], pour proposer un protocole efficace et sécurisé faisant intervenir plus de deux parties, c'est-à-dire que les valeurs des deux vecteurs peuvent appartenir à plusieurs entités. Les données privées de chacune des différentes parties sont gardées secrètes sauf le résultat final du produit scalaire qui, quant à lui, peut être connu par tous les intervenants.

Le protocole proposé par Yao et *al.* est équivalent en terme de complexité de calcul et de communication aux protocoles proposés dans le cas à deux parties. Mais son défaut majeur réside dans le fait qu'il ne protège pas complètement contre les attaques suivant un modèle semi honnête défini par Goldreich dans [13] (voir section 4.1) qu'il prétend pourtant respecter, dans la mesure où une

collusion de 4 parties peut casser le protocole.
Dans [5] nous avons amélioré la proposition de Yao et *al.* après avoir démontré qu'elle ne respectait pas le modèle semi honnête. Nous avons prouvé, en donnant quelques scénarios d'attaques, qu'avec juste une collusion composée de quatre parties, ce qui est en pratique facile à réaliser, on peut déduire les données privées d'un participant au protocole. Pour remédier à ce problème, nous avons proposé un nouveau schéma plus efficace, basé sur un chiffrement homomorphe et sur le protocole d'envoi superposé appelé aussi le problème du dîner des cryptographes. La sécurité de notre modèle respecte parfaitement le modèle semi honnête et, plus particulièrement, il est optimal en terme de résistance aux collusions.

1.2. Organisation

Dans ce qui suit, nous allons commencer par faire le tour des différents modèles de confiance qui existent. Par la suite, nous rappellerons quelques notions cryptographiques de base nécessaires à notre protocole. Dans la section 4, nous présenterons notre protocole. Enfin, nous conclurons en section 5.

2. Quelques caractéristiques des modèles basés sur la réputation

La notion de confiance dans un réseau est définie comme étant la relation existant entre les différents utilisateurs du réseau [10]. La façon d'évaluer la confiance dépend de l'application concernée. Nous pouvons trouver dans la littérature plusieurs algorithmes d'évaluation de la confiance d'un utilisateur. On peut citer entre autres ces deux articles : [3] et [18]. Ces confiances sont ensuite utilisées et échangées dans le cadre d'une application pour répondre aux exigences de sécurité du réseau. Les mécanismes qui permettent cette distribution sont appelés les modèles d'établissement des confiances.
Les modèles basés sur la confiance peuvent être centralisés ou décentralisés. Centraliser le calcul des confiances, en affectant cette tâche à une seule partie du système, pose quelques problèmes. Le premier problème est la formation d'un noeud critique, c'est-à-dire que tout le monde dépend de l'intégrité et du bon fonctionnement de cette autorité commune. Un deuxième problème qu'on peut soulever dans ce cas est que toutes les confiances seront établies par un seul et unique juge, ce qui ne laisse guère de place aux opinions des différents utilisateurs du système. En ce qui concerne l'approche décentralisée, chaque utilisateur est responsable du calcul des confiances qu'il accorde aux autres. C'est cette dernière approche qui est très largement utilisée.
Le calcul des confiances peut être réactif ou proactif. Dans la méthode proactive, le calcul des confiances est réalisé périodiquement. Dans ce cas, on utilise beaucoup plus de bande passante (la méthode génère beaucoup plus de communication). D'un autre côté, la méthode réactive initie le calcul des confiances uniquement quand un utilisateur en fait explicitement la demande. Le choix de l'une des deux méthodes dépend de l'application et du réseau.
La question qui est souvent posée lors de la mise en oeuvre d'un modèle de confiance est la suivante : Est ce qu'on ne prend en considération que les recommandations positives, que les recommandations négatives ou bien les deux en même temps ? Le fait de tenir compte des recommandations positives et négatives simultanément, permet d'avoir un résultat beaucoup plus précis et correspondant plus à la réalité. D'un autre côté, inclure des valeurs négatives peut engendrer quelques attaques de type déni de service : Un utilisateur malicieux peut n'attribuer exagérément que des valeurs négatives à un autre utilisateur, ce qui fausse le résultat final et pénalise l'utilisateur victime.

3. Primitives de Bases

Nous allons présenter dans cette section les différentes primitives cryptographiques et les principaux protocoles utilisés dans notre modèle.

3.1. Envoi superposé

L'envoi superposé est un protocole selon lequel les utilisateurs se mettent d'accord pour tous émettre, même quand il n'ont rien à transmettre, en utilisant un format tel que la combinaison de toutes les émissions produit :

– un résultat nul, si aucun utilisateur n'a essayé de transmettre des informations,
– le message d'un utilisateur si un seul d'entre eux a essayé de transmettre,
– des messages brouillés entre eux si plusieurs utilisateurs ont essayé de transmettre en même temps.

En 1988, David Chaum a proposé le premier protocole d'envoi superposé tel que, quelle que soit l'issue finale du protocole, on ne puisse rien apprendre sur les intentions de transmettre des utilisateurs [6]. Pour décrire ce protocole, il a utilisé une parabole dans laquelle trois cryptographes se trouvent à une même table et veulent savoir si quelqu'un a payé, sans pour autant savoir lequel d'entre eux à réellement payé (c'est-à-dire que quelqu'un dise "J'ai payé" avec anonymat d'émission). Les réseaux d'ordinateurs suivant ce protocole sont appelés les réseaux du dîner des cryptographes ou DC-nets, et le protocole est connu sous le nom de protocole DC-net.

Envoi Superposé

Soit un ensemble d'utilisateurs $U_1, ..., U_n$. Supposons que chaque couple d'utilisateurs (U_i, U_j) possède un secret commun, $s_{i,j} = -s_{j,i}$ de l-bits.
Chaque utilisateur U_i encode un message $Message(i)$ en une chaîne de *l*-bits à zéro s'il n'a rien à transmettre, ou bien en un message de *l*-bit s'il a quelque chose à transmettre.

Déroulement du tour :

1. Chaque utilisateur U_i diffuse :
   $Diffusion(i) = Brouillage(i) + Message(i)$ où $Brouillage(i) = \sum_{j=1,\, j\neq i}^{n} s_{i,j}$.
2. Chaque utilisateur calcule le résultat du tour :
   $Résultat = \sum_{i=1}^{n} Diffusion(i)$.

Comme chaque $s_{i,j}$ apparaît sous deux formes (Une fois inséré par U_i $(s_{i,j})$ et une deuxième fois inséré par U_j $(-s_{i,j})$), les brouillages s'annulent mutuellement et chaque utilisateur obtient $Résultat = \sum_{i=1}^{n} Message(i)$.

3.1.1. Propriétés de sécurité

Nous allons essayer maintenant de donner quelques preuves informelles du niveau de sécurité de DC-net. Soit la somme modulaire $+$, pour deux mots binaires S et M, la théorie de Shannon nous permet de dire que :
– si on ne connaît rien sur S, on ne peut rien apprendre sur $M + S$ à partir de M,
– si on ne connaît rien sur S, on ne peut rien apprendre sur M à partir de $M + S$.

Ainsi, si les $S_{i,j}$ ne sont utilisés qu'une seule fois, il suffit d'après la première propriété qu'on ne connaisse pas l'un d'entre eux pour qu'on ne puisse rien apprendre sur $Brouillage(i)$. De plus, d'après la deuxième propriété, si on ne sait rien sur $Brouillage(i)$ on ne peut rien apprendre sur $Message(i)$ à partir de $Diffusion(i)$. Ces deux remarques nous permettent de dire que pour toute $Diffusion(i)$ il suffit qu'un seul utilisateur U_j ne trahisse pas U_i en dévoilant $S_{i,j}$ pour qu'on ne puisse rien apprendre sur $Message(i)$.
Pour que les affirmations précédentes soient vraies il faut absolument que les $S_{i,j}$ ne soient utilisés qu'une fois, et que ceux-ci ne soient pas prévisibles. La proposition de David Chaum était que les utilisateurs échangent des disques optiques avec des grandes quantités de bits aléatoires pour former les $S_{i,j}$. Avec cette solution on peut émettre anonymement autant de bits qu'il y en a dans le disque optique. Dans la pratique, la solution la plus généralement adoptée est que les utilisateurs n'échangent pas des $S_{i,j}$ mais des graines $G_{i,j}$ qui, avec des Générateurs de Nombres Pseudo-Aléatoires (GNPAs), permettent de générer un $S_{i,j}$ à chaque tour.
Si on utilise des générateurs de nombres pseudo-aléatoires, la sécurité du protocole repose sur la sécurité du générateur et n'est plus inconditionnelle comme dans le protocole de base. Cependant, la simplification apportée par l'utilisation de ces générateurs et le fait que de nos jours la sécurité est souvent basée sur le même type d'hypothèses, font que cette dégradation de la sécurité est généralement acceptée. L'utilisation de l'envoi superposé donne lieu à des groupes d'émission non-observable composés de l'ensemble des utilisateurs participant au protocole, et ceci quel que soit le type d'attaquant.

3.2. Chiffrement Homomorphe

Les schémas homomorphes sont décrits dans [8,17]. Ils sont composés de trois fonctions (Gen, Enc, Dec), Gen est la fonction de génération des clés privées s_k et des clés publiques p_k. Enc et Dec sont respectivement les fonctions de chiffrement et de déchiffrement. Un chiffrement est appelé homomorphe, si les propriétés suivantes sont vérifiées : $Enc_{p_k}(x;r).Enc_{p_k}(y;r') = Enc_{p_k}(x+y;r.r')$, où x et y représentent les messages en clair, r et r' représentent des chaînes de bits aléatoires. Une autre propriété de cette fonction est que $Enc_{pk}(x;r)^y = Enc_{pk}(x.y;r^y)$.

3.3. Générateur de Nombres Pseudo Aléatoires

Les GNPAs ont été introduits par Goldreich, Goldwasser and Micali [14]. Un générateur de nombres pseudo aléatoires est défini dans [16] comme étant un algorithme déterministe qui, ayant comme entrée une séquence de bits aléatoires, fournit en sortie une autre séquence de bits aléatoires. La donnée en entrée du GNPA est appelée graine.

Les exigences minimales en terme de sécurité pour un GNPA est que la longueur l de la graine qui doit être suffisamment grande pour que la recherche dans 2^l soit irréalisable pour un attaquant. Les recommandations concernant les critères de sélection d'une graine peuvent être trouvés dans [2].

3.4. Protocole de Diffie-Hellman

Le protocole de Diffie Hellman (DH) est l'un des protocoles les plus sécurisés et les plus fréquemment utilisés des protocoles d'échange de clés. Il a été proposé par Whitfield Diffie and Martin Hellman en 1976 [9].

Le schéma d'échange de clés secrètes de DH entre deux utilisateurs $Alice$ et Bob peut être résumé comme suit :

- Alice et Bob choisissent un groupe (soit un corps fini, dont ils n'utilisent que la multiplication, soit une courbe elliptique) et une génératrice g de ce groupe.
- Alice choisit un nombre au hasard a, élève g à la puissance a, et envoie à Bob g^a.
- Bob fait de même avec le nombre b.
- Alice, en élevant le nombre reçu de Bob à la puissance a, obtient g^{ba}.
- Bob fait le calcul analogue et obtient g^{ab}, qui est le même. Mais puisqu'il est difficile d'inverser l'exponentiation dans un corps fini, c'est-à-dire de calculer le logarithme discret, une tierce personne, par exemple Eve, ne peut pas découvrir a et b, donc ne peut pas calculer g^{ab}.

4. Notre Proposition

Notre objectif à travers cet article est de donner une solution au problème du produit scalaire distribué appliqué aux calculs des confiances, défini comme suit : Soit $P = \{p_1, p_2, ..., p_N\}$ l'ensemble de tous les participants au réseau. Alice, un élément de P noté A, veut calculer la confiance qu'elle accorde à E (un autre élément de P) en prenant en considération les recommandations de chaque élément B_i (pour $1 \leq i \leq n$) de P. Les données de Alice sont représentées par le vecteur $X = (x_1, ..., x_n) \in z_m^n$. La donnée de chacun des B_i est la valeur y_i. A la fin du protocole, seul le résultat final du produit scalaire $X.Y \mod m$ est connu par Alice et/ou par chacun des autres participants, où $Y = (y_1, ..., y_n) \in z_m^n$ et m sont des paramètres publics.

4.1. Modèle d'attaque

L'attaquant suit le modèle semi honnête défini par Goldreich in [13], aussi appelé le modèle honnête mais curieux. Dans cet article, Goldreich a proposé une définition formelle de ce modèle. Un attaquant suit un modèle semi honnête s'il suit convenablement le protocole à l'exception qu'il peut enregistrer ses résultats intermédiaires dans le but de déduire plus d'informations. Dans le modèle semi honnête, les parties peuvent aussi s'échanger et recouper leurs résultats en formant des collusions.

4.2. Notations

Nous donnons ici quelques notations utilisées dans notre protocole.

$prior(p_i)$	La priorité de chaque participant p_i, nous pouvons prendre ici : $prior(p_i) > prior(p_j)$ si $i > j$
$s_{i,j}$	La clé secrète de m bits partagée par les deux participants p_i et p_j. $s_{i,j} = -s_{i,j}$ si $prior(p_i) > prior(p_j)$
$\|s_{i,j}\|$	Le nombre de bits de $s_{i,j}$
$Rand_{seed}$	GNPA basé sur la graine $seed$
pk	Une clé publique
sk	Une clé privée
Enc_{pk}	La fonction de chiffrement homomorphe utilisant la clé publique pk
Dec_{sk}	La fonction de déchiffrement homomorphe utilisant la clé privée sk

4.3. Description du protocole

Afin de résoudre le problème décrit ci-dessus, nous allons introduire un nouveau schéma beaucoup plus efficace et beaucoup plus sûr que celui proposé par Yao et *al.* [20]. Il est inspiré de l'envoi superposé et basé sur le chiffrement homomorphe, tous les deux décrits un peu plus haut. Le schéma que nous proposons est en particulier optimal en terme de résistance aux collusions. Un résumé du protocole est présenté ci-dessous :

Résumé du protocole

- En utilisant un chiffrement homomorphe (section 3.2), A chiffre une à une les valeurs x_i ($Enc_{p_k}(x_i; r_i)$) et les envoie aux B_i ;
- Chaque B_i calcule l'exponentielle y_i du chiffrement reçu de la part de A comme suit (voir propriétés du chiffrement homomorphe) : ($w_i = Enc_{p_k}(x_i.y_i; r_i^{y_i})$) et masque le résultat en utilisant l'envoi superposé décrit à la (section 3.1) :
  - Le masque $MSK_i = \sum_{j=1, j\neq i}^{n} s_{i,j}$
  - Le résultat du chiffrement est envoyé à A : $w_i^{'} = w_i \ . \ Enc_{p_k}(MSK_i; r_i^{'})$.
- Alice calcule le produit de tous les $w_i^{'}$, et déduit directement le résultat final $X.Y = \sum_{i=1}^{n} x_i y_i$ (Voir le fonctionnement du protocole DC-net et du chiffrement homomorphe).

La sécurité de notre protocole repose sur celle de DC-net et sur celle du chiffrement homomorphe. Par ailleurs, il est prouvé que si les valeurs s_{ij} sont prises aléatoirement et renouvelées à chaque utilisation, l'envoi superposé est inconditionnellement sûr. Quant au chiffrement homomorphe, il est prouvé qu'il est sémantiquement sûr.

Le maintien de cette sécurité inconditionnelle est difficilement réalisable en pratique. Afin de résoudre ce problème, nous avons fait appel aux systèmes de chiffrement à clés publiques, qui sont des mécanismes facilement réalisables. Le mécanisme à clé publique utilisé dans notre protocole pour l'échange des secrets $s_{i,j}$ est le protocole de Diffie Hellman (voir section 3.4).

Cette deuxième version du protocole engendre des coûts non négligeables en terme de communication et de calcul. Ceci est causé essentiellement par le calcul des exponentielles et aux différentes étapes que comporte un échange de DH. Pour résoudre ce problème, nous avons pensé à l'utilisation des GNPAs. Avec l'aide de ce mécanisme, DH n'est exécuté qu'une seule fois dans le but de générer une graine pour chaque couple (B_i, B_j) noté $seed_{B_i,B_j}$. Ensuite, à chaque round, les participants renouvellent leurs secrets s_{B_i,B_j} au moyen du générateur pseudo aléatoire.

D'autres améliorations peuvent être remarquées dans la version donnée ci-dessous et dans [5].

Description du protocole

Soit $P = \{p_1, p_2, ..., p_N\}$. p_i peut jouer le rôle de A, E ou B_i.

$G = Z_p^*$ représente un groupe multiplicatif, fini et cyclique d'ordre p. Soit le générateur g et considérons $|G| > 2^{2t}$.

k est un paramètre choisi proportionnellement à la sécurité que nous voulons atteindre.

A L'initialisation du protocole (round r=1) : pour tout (A, E et $B = \{B_1, B_2, ..., B_n\}$) faire

Chaque couple (B_i, B_j) se met d'accord sur (G, g).

Pour tout $B_i \in B$ **faire**

- B_i sélectionne aléatoirement un secret : $e_i \in Z_p$.
- B_i choisi aléatoirement parmi les $n-1$ participants, un ensemble de k amis différents $\{f_1, f_2, ..., f_k\}$.
- **Pour** j **de** 1 **à** k **faire**
  - B_i envoie à f_j la partie publique g^{e_i} et une preuve de connaissance de e_i.
  - Quand f_j reçoit g^{e_i}, il :
    - calcule $(g^{e_i})^{e_{f_j}}$.
    - répond à B_i en lui envoyant à son tour sa partie publique $g^{e_{f_j}}$ et une preuve de connaissance de e_{f_j}.
  - B_i calcule $(g^{e_{f_j}})^{e_i}$.
  - Enfin, B_i et f_j hachent $g^{e_i e_{f_j}}$ en une chaîne de t-bit : $seed_{i,f_j} = h(g^{e_i e_{f_j}})$ qui sera la graine secrète entre B_i et f_j.

A chaque round $r \geq 1$ **:**

Considérons $n \geq 2$ et $s^{r=0}_{B_i,B_j} = seed_{B_i,B_j}$.

Pour i **de** 1 **à** n **faire**

Soit l'ensemble $BF_{B_i} = \{B_j / s^r_{B_i B_j} existe\}$

- Alice génère une paire de clés privée et publique (s_k, p_k) et envoie p_k à B_i.
- Alice chiffre l'élément x_i de son vecteur X avec sa clé publique de façon homomorphe. Le résultat de ce chiffrement $c_i = Enc_{p_k}(x_i; r_i)$ est envoyé à B_i où r_i est une chaîne de bits aléatoires.
- Comme le chiffrement est homomorphe, B_i n'est pas obligé de connaître la clé privée de Alice pour chiffrer $x_i y_i$. B_i calcule w_i comme suit :
  - $w_i = c_i^{y_i} \bmod m$, on obtient : $w_i = Enc_{p_k}(x_i.y_i; r_i^{y_i})$.
- **Pour** j **de** 1 **à** $|BF_{B_i}|$ **faire**
  - B_i Calcule le masque : $MSK_{B_i} = \sum_{B_j \in BF_{B_i}} s^r_{B_i B_j}$ où, $s^r_{B_i,B_j} = Rand_{seed_{B_i,B_j}}(s^{r-1}_{B_i,B_j})$
- Dans le but de cacher y_i à Alice, B_i calcule :
  - Le chiffrement $w_i^{'} = w_i \ . \ Enc_{p_k}(MSK_{B_i}; r_i^{'})$.
- Esuite B_i envoie $w_i^{'}$ à Alice.

Quand Alice reçoit tous les $w_i^{'}$, elle calcule :
$\prod_{i=1}^n w_i^{'} \bmod m =$
$\prod_{i=1}^n Enc_{p_k}(x_i.y_i; r_i^{y_i}).Enc_{p_k}(MSK_{B_i}; r_i^{'}) = \prod_{i=1}^n Enc_{p_k}(x_i.y_i + MSK_{B_i}; r_i^{y_i}.r_i^{'}) =$
$Enc_{p_k}(\sum_{i=1}^n x_i y_i + \sum_{i=1}^n MSK_{B_i}; \prod_{i=1}^n r_i^{y_i}.r_i^{'}) = Enc_{p_k}(X.Y + \sum_{i=1}^n MSK_{B_i}; \prod_{i=1}^n r_i^{y_i}.r_i^{'})$ $(*)$

Alice utilise la fonction Dec et sa clé privée s_k pour déchiffrer $(*)$. Comme chaque s_{B_i,B_j} apparaît sous deux formes (une fois inséré par B_i (s_{B_i,B_j}) et une fois inséré par B_j ($-s_{B_i,B_j}$)), la somme $\sum_{i=1}^n MSK_{B_i}$ sera égale à zéro, et par conséquent Alice obtient immédiatement le résultat final $X.Y$.

Nous avons prouvé dans [5] que la sécurité de notre protocole dépend du paramètre k. Effectivement, La résistance aux collusions augmente proportionnellement à la valeur de k. Par exemple, pour $k = (n-1)/2$, on obtient approximativement pour chaque B_i une résistance aux collusions constituées de $(n-2)B_j$, ce qui signifie que dans ce cas, notre protocole est résistant à une collusion de tous les participants moins un.
En ce qui concerne les performances de notre protocole. Les traitements génèrent des coûts de calcul équivalents à ceux du protocole de Yao et *al.*. Quant aux coûts de communication, nous réduisons de 50% les communications par rapport au protocole de Yao et *al.*

5. Conclusion et Perspectives

Dans cet article, nous avons tout d'abord discuté de la problématique associée aux calculs distribués et tout particulièrement du problème lié à la sécurité du produit scalaire appliqué aux modèles de confiance. Ensuite, Nous avons proposé un nouveau protocole beaucoup plus sûr et plus efficace notamment en terme de coûts de communication que ses prédécesseurs. Effectivement, notre protocole permet une haute résistance aux collusions dans le sens où il suffit qu'un seul participant soit honnête pour que le protocole assure la confidentialité des données privées de Alice et des B_i. Concernant les performances de notre protocole, il réduit de 50% les communications par rapport au protocole de Yao et *al.*

Bibliographie

1. Mikhail J. Atallah et Wenliang Du. Secure multi-party computational geometry. pages 165–179. Springer-Verlag, 2001.
2. E. Barker et J. Kelsey. Recommendation for random number generation using deterministic random bit generators. SP-800-90, U.S. DoC/National Institute of Standards and Technology, 2006. http ://csrc.nist.gov/publications/nistpubs/.
3. Thomas Beth, Malte Borcherding, et Birgit Klein. Valuation of trust in open networks. In *ESORICS '94 : Proceedings of the Third European Symposium on Research in Computer Security*, pages 3–18, London, UK, 1994. Springer-Verlag.
4. Bogdan Carbunar et Radu Sion. Uncheatable reputation for distributed computation markets. In *Financial Cryptography*, volume 4107 sur *Lecture Notes in Computer Science*, pages 96–110. Springer, 2006.
5. Boussad Ait-salem Carlos Aguilar Melchor et Philippe Gaborit. A collusion-resistant distributed scalar product protocol with application to privacy-preserving computation of trust. *Network Computing and Applications, IEEE International Symposium on*, 2009.
6. David Chaum. The dining cryptographers problem : Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1 :65–75, 1988.
7. David Chaum, Claude Crépeau, et Ivan Damgard. Multiparty unconditionally secure protocols. In *STOC '88 : Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 11–19, New York, NY, USA, 1988. ACM.
8. Ivan Damgard et Mats Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *PKC '01 : Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography*, pages 119–136, London, UK, 2001. Springer-Verlag.
9. Whitfield Diffie et Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6) :644–654, 1976.
10. Laurent Eschenauer, Virgil D. Gligor, et John Baras. On trust establishment in mobile ad-hoc networks. In *In Proceedings of the Security Protocols Workshop*, pages 47–66. Springer-Verlag, 2002.
11. Bart Goethals, Sven Laur, Helger Lipmaa, et Taneli Mielikäinen. On private scalar product computation for privacy-preserving data mining. In Choonsik Park et Seongtaek Chee, editors, *Information Security and Cryptology - ICISC 2004, 7th International Conference, Seoul, Korea, December 2-3, 2004, Revised Selected Papers*, volume 3506 sur *Lecture Notes in Computer Science*, pages 104–120. Springer, 2004.
12. O. Goldreich, S. Micali, et A. Wigderson. How to play any mental game. In *STOC '87 : Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 218–229, New York, NY, USA, 1987. ACM.
13. Oded Goldreich. Secure multi-party computation. Final Draft, 2002.
14. Oded Goldreich, Shafi Goldwasser, et Silvio Micali. How to construct random functions. *J. ACM*, 33(4) :792–807, 1986.
15. Reto Kohlas et Ueli M. Maurer. Confidence valuation in a public-key infrastructure based on uncertain evidence. In *PKC '00 : Proceedings of the Third International Workshop on Practice and Theory in Public Key Cryptography*, pages 93–112, London, UK, 2000. Springer-Verlag.
16. Alfred J. Menezes, Paul C. van Oorschot, et Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 2001.
17. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. pages 223–238. 1999.
18. Huu Tran, Michael Hitchens, Vijay Varadharajan, et Paul Watters. A trust based access control framework for p2p file-sharing systems. In *HICSS '05 : Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, page 302.3, Washington, DC, USA, 2005. IEEE Computer Society.
19. A. Yao. Protocols for secure computations. In *the twenty-third annual IEEE Symposium on Foundations of Computer Science*, pages 160–164. IEEE Computer Society, 1982.
20. Danfeng Yao, Roberto Tamassia, et Seth Proctor. Private distributed scalar product protocol with application to privacy-preserving computation of trust. In *IFIPTM 2007 – Joint iTrust and PST Conferences on Privacy, Trust Management and Security*, July 2007.
21. C. Zouridaki, B. L. Mark, M. Hejmo, et R. K. Thomas. A quantitative trust establishment framework for reliable data packet delivery in manets. In *SASN '05 : Proceedings of the 3rd ACM workshop on Security of ad hoc and sensor networks*, pages 1–10, New York, NY, USA, 2005. ACM.