Un nouveau schéma de partage de secrets publiquement vérifiable

Assia Ben Shil
Laboratoire LIP2
Tunis- Tunisie
Email : essia.benshil@gmail.com

Kaouther Blibech
Laboratoire LIP2
Tunis- Tunisie
Email : kaouther.blibech@gmail.com

Riadh Robbana
Laboratoire LIP2
Tunis- Tunisie
Email : riadh.robbana@gmail.com

***Résumé*—Dans un schéma de partage de secrets, un courtier distribue des parts d'un secret S entre un ensemble de n participants de telle manière que seulement quelques sous-ensembles prédéfinis de participants (appelés structures d'accès) peuvent reconstruire le secret, les autres ne pouvant rien déduire concernant le secret. Dans [7], Stadler a introduit la notion de vérifiabilité publique pour les schémas de partage de secrets. Dans un schéma de partage de secrets publiquement vérifiable (PVSS), les participants ayant reçu une part du secret durant le processus de distribution peuvent vérifier que le courtier a correctement calculé et distribué les parts. Les participants, ainsi que toute autre personne, peuvent aussi vérifier les parts les uns des autres. Dès lors, plusieurs schémas PVSS ont été proposés. Dans [15], un schéma PVSS intéressant a été proposé par Behnad et Eghlidos, permettant aux participants détenant une part du secret d'utiliser un protocole "*zero-knowledge*" pour prouver leur appartenance ainsi que la correction de leurs parts. Dans ce papier, nous présentons un nouveau schéma PVSS aussi sécurisé mais plus simple que le schéma cité.**

I. Introduction

Le partage de secrets est une technique cryptographique permettant de diviser un secret entre un ensemble de participants tel que seulement quelques sous-ensembles prédéfinis de participants (appelés structures d'accès) peuvent reconstruire le secret.

En 1979, Shamir [2] et Blakley [1] ont indépendamment introduit l'idée de partage de secrets. Généralement, un schéma de partage de secrets opère en deux phases : une phase de distribution durant laquelle le courtier calcule les parts et les distribue en donnant une part à chaque participant et une phase de reconstruction durant laquelle les membres d'un sous-ensemble qualifié de participants regroupent leurs parts afin de reconstruire le secret.

La plupart des schémas de partage de secrets se basent sur la technique de partage de secrets de Shamir [2]. Le partage de secret de Shamir se base sur l'interpolation polynomiale. L'idée est que la courbe associée à un polynôme donné passe par une infinité de points mais peut être définie de manière unique grâce à un nombre fixe de points.

Malgré son efficacité, le schéma de Shamir présente quelques limites. En effet, dans ce schéma, il faut accorder une confiance absolue au courtier. Ce dernier, peut distribuer quelques parts erronées empêchant ainsi les participants de reconstruire le secret initial partagé. Les schémas de partage de secrets vérifiables (VSS) [3, 5, 6] ont été proposés pour permettre aux participants de vérifier la validité des parts distribuées par le courtier. Cependant, un participant malicieux peut recevoir une part valide et soumettre une part invalide durant le processus de reconstruction. Les schémas de partage de secrets publiquement vérifiables (PVSS) [7-17] ont été proposés pour remédier à ce problème. En effet, les schémas PVSS permettent d'empêcher le courtier et les participants de frauder. Dans un schéma PVSS, la validité des parts distribuées peut être vérifiée par n'importe qui.

Dans [15], Behnad et Eghlidos présentent un schéma PVSS intéressant où les participants peuvent prouver leur appartenance et la validité de leurs parts grâce à un protocole *zero-knowledge* [1]. Ainsi, une partie non autorisée ne peut pas participer à la phase de reconstruction.

Dans ce papier, nous présentons un nouveau schéma PVSS basé sur le schéma PVSS proposé par Behnad et Eghlidos dans [15] et nous montrons que notre schéma PVSS est plus simple que ce dernier tout en gardant le même niveau de sécurité du schéma mentionné.

Ce papier est organisé comme suit : Tout d'abord, nous présentons les schémas PVSS et spécialement celui de Behnad et Eghlidos [15]. Ensuite, nous introduisons notre nouveau schéma PVSS, nous étudions la sécurité de notre schéma PVSS et nous effectuons une comparaison entre notre schéma et le schéma de Behnad et Eghlidos. Nous concluons par quelques remarques utiles.

II. Les schémas PVSS

Les schémas PVSS ont été introduits par Stadler dans [7] afin de permettre à n'importe qui (et non seulement les participants détenant une part du secret) de vérifier que les parts ont été correctement distribuées par le courtier.

Dans ce papier, Stadler a proposé deux schémas PVSS. Le premier est utilisé pour partager un logarithme discret. Il se base sur une hypothèse non standard concernant le "double logarithme discret". Il s'agit des expressions de la forme $y = g^{(h^x)}$ (avec g un générateur du groupe d'ordre p, et h un élément fixe d'ordre élevé dans Z_p^*) de façon qu'étant donné y, il est difficile de trouver x. Sous cette hypothèse, ce schéma

1. Dans un protocole zero-knowledge, on essaie de convaincre un vérificateur, interactivement, qu'on connaît réellement un secret sans le révéler. Le vérificateur ne peut découvrir aucune information concernant le secret même s'il ne respecte pas le protocole zero-knowledge.

est aussi difficile que le problème Diffie-Hellman Décisionnel. Le second schéma est basé sur le problème de la racine RSA. Il est utilisé pour partager une racine n-ième et dépend de l'hypothèse RSA. Les chiffrements se basent sur une variante du protocole d'échange de clefs de Diffie-Hellman. Cependant, la sécurité de ce schéma n'a pas été formellement étudiée. De plus, pour ces deux schémas, la vérification nécessite un échange d'informations entre le vérificateur et le participant. Il s'agit d'une vérification interactive.

Dans [8], Fujisaki et Okamoto ont défini la non-interactivité pour un schéma PVSS comme étant le fait que la vérification d'une part peut être faite sans aucune communication avec le courtier ou avec n'importe quel participant. Le schéma qu'ils ont proposé dans [8] dépend de "l'hypothèse RSA modifiée" qui suppose qu'en inversant la fonction RSA, elle reste difficile. Cette hypothèse RSA modifiée permet une découverte partielle.

Notons que les schémas présentés dans [7, 8] dépendent de quelques hypothèses non standards. Dans [11], Schoenmakers fournit un schéma PVSS plus avancé en ajoutant le fait qu'en soumettant sa part, le participant doit fournir une preuve de sa correction. Son schéma PVSS est plus simple que les schémas précédents. Il utilise des techniques fonctionnant sur n'importe quel groupe dans lequel le problème du logarithme discret est difficile. Ce schéma est aussi difficile que le problème Diffie-Hellman Décisionnel.

Dans [12], Young et Yung ont proposé une amélioration du schéma PVSS de Schoenmakers. Le schéma qu'ils ont proposé pour partager le logarithme discret est aussi difficile que le problème du logarithme discret. Ils ont prouvé dans [7] que leur schéma est calculatoirement zero-knowledge. De plus, dans les schémas PVSS, des hypothèses de chiffrement sécurisé sont employées. Mais, dans leur schéma, Young et Yung peuvent utiliser n'importe quel chiffrement probabiliste.

Dans [9], Boudot et Traore ont proposé de nouveaux schémas PVSS permettant aux participants de déchiffrer leurs parts rapidement (découverte rapide ou *fast recovery*) ou bien après un nombre prédéfini d'opérations (découverte différée ou *delayed recovery*). En effet, ils fournissent un schéma PVSS pour partager un logarithme discret avec une découverte rapide et un schéma PVSS pour partager une factorisation avec une découverte rapide. Ils présentent aussi un schéma PVSS pour partager un logarithme discret avec une découverte différée et un schéma PVSS pour partager une factorisation avec une découverte différée.

Dans la plupart des schémas PVSS existants, la phase de vérification est interactive. Ceci est dû à l'utilisation du protocole *zero-knowledge* de Fiat-Shamir [4]. Dans [13], Ruiz et Villar ont proposé un schéma PVSS avec une vérification non-interactive. C'est le premier schéma PVSS efficace n'utilisant pas la technique de Fiat-Shamir. Il est basé sur les propriétés homomorphiques du schéma de chiffrement de Paillier [10]. C'est le premier schéma PVSS connu basé sur l'Hypothèse Décisionnelle de Résiduosité Composite (HDRC). Le processus de vérification de ce schéma est plus simple que celui des autres schémas connus.

Récemment, d'autres schémas PVSS ont été proposés. Dans [16], Jhanwar a proposé un autre schéma PVSS non-interactif basé sur le pairing. Dans [14], Yu & all ont proposé un schéma PVSS avec la possibilité d'ajouter de nouveaux participants au schéma. Dans [17], Wu & all ont proposé un schéma PVSS basé sur le pairing et réduisant la complexité de calcul en gardant le même niveau de sécurité des systèmes à clefs publiques existants.

Behnad et Eghlidos ont proposé dans [15] un schéma PVSS avec une vérification non-interactive et possédant deux caractéristiques. D'abord, après la distribution des parts et en cas de plainte de la part de n'importe quel participant, une tierce partie peut initier un protocole de résolution de conflits afin d'identifier celui qui ment. Cette tierce partie peut déterminer qui du courtier ou du participant est en train de mentir. De plus, Behnad et Eghlidos ajoutent un processus de preuve d'appartenance avant la reconstruction du secret. Dans ce processus, un participant doit prouver son appartenance et la validité de sa part en même temps.

Le schéma de Behnad et Eghlidos est le premier schéma PVSS dans lequel le problème d'appartenance est traité de manière explicite et qui a ajouté un processus de gestion de conflits en cas de plainte. C'est pourquoi nous trouvons ce schéma intéressant et nous nous concentrerons sur ce dernier dans la section suivante. Dans ce papier, nous introduisons un nouveau schéma PVSS avec une vérification non-interactive, un processus de résolution de conflits et un processus de preuve d'appartenance mais qui est plus simple que ce dernier.

III. Le schéma PVSS de Behnad et Eghlidos

Comme les autres schémas PVSS, la phase de partage consiste en un processus de distribution dans lequel les parts sont calculées et envoyées aux participants et un processus de vérification dans lequel la validité des parts est vérifiée. Behnad et Eghlidos ont ajouté un troisième processus appelé résolution de conflits. Dans ce processus, le participant présente une plainte contre le courtier si la part reçue n'est pas valide.

Dans le schéma PVSS de Behnad et Eghlidos, tous les calculs sont effectués dans deux ensembles, Z_p et Z_q (p et q sont deux grands entiers premiers tels que $q|p-1$[2]). De plus, les notations suivantes sont utilisées :

- G_q est un sous groupe de Z_p^* d'ordre q , tel que le calcul du logarithme discret dans ce groupe n'est pas faisable.
- $g \in G_q$ est un générateur de ce groupe.

Les calculs de la forme g^α sont effectués dans Z_p. Les exponentiations sont effectuées dans Z_q. Tous les autres calculs sont effectués dans Z_q. [15]

Le courtier fixe le polynôme $F(x) = F_0 + F_1 x + ... + F_{k-1}x^{k-1}$, où $F_1, \ldots, F_{k-1} \in_R Zq$ et F_0 est le secret à partager. Il publie $C_i = g^{F_i}$ pour $i = 0, \ldots, k-1$. De plus, le courtier choisit sa clef privée $d \in_R Zq$ et publie sa clef publique g^d. De même, chaque participant Pr_j pour $1 \leq j \leq$

2. q divise p-1.

n choisit sa clef privée $a_j \in_R Zq$ et publie sa clef publique g^{a_j}. Ensuite, le courtier calcule les parts $s_j = F(j)$ pour $j = 1, \ldots, n$. Pour chaque part s_j, il calcule la part chiffrée $E_j = s_j * (g^{a_j})^d$ et l'envoie au participant Pr_j.[15]

Chaque participant Pr_j calcule $s_j = E_j * [(g^d)^{a_j}]^{-1}$ et vérifie que $g^{s_j} = \prod_{i=0}^{k-1} C_i^{j^i}$ [3]. Si ceci est vrai, alors, la part est valide, sinon, le participant présente une plainte contre le courtier. [15]

Dans le cas d'une plainte, une tierce partie R peut voter contre le courtier D, ou contre le participant A, grâce au processus de résolution de conflits. Le courtier et le participant essaient de prouver leur honnêteté à R. Pour ce faire, A et D partagent une clef secrète et publient un "engagement [4]" pour cette clef. Le courtier utilise la clef secrète pour chiffrer et envoyer la part du participant A à R de façon que ce dernier puisse vérifier, en utilisant la clef publique, que cette part peut être calculée à partir des coefficients du polynôme de partage.

Au début de la phase de reconstruction, Behnad et Eghlidos ont ajouté un nouveau processus appelé "preuve d'appartenance". Dans ce processus, chaque participant doit prouver son appartenance et la validité de sa part aux autres participants. De plus, ce processus peut être initié par n'importe quelle tierce partie. Une fois l'appartenance des participants vérifiée, le secret peut être reconstruit à partir des parts soumises, comme suit : $s = \sum_{i=1}^{k} w_i s_i$ où $w_i = \sum_{i \neq j} i/(j-1)$ (Coefficient de Lagrange) [15].

IV. Un nouveau schéma PVSS

L'idée de notre schéma est de publier des engagements pour les parts au lieu de publier les coefficients du polynôme de partage. En fait, à partir des parts on peut retrouver les coefficients du polynôme de partage et vice versa. Cependant, la publication des engagements pour les parts est beaucoup plus appropriée pour les applications où la structure d'accès est dynamique et où les utilisateurs doivent avoir accès à la trace des modifications de cette dernière.

Dans notre schéma, nous effectuons les calculs dans les mêmes ensembles que le schéma de Behnad et Eghlidos et nous utilisons les mêmes notations.

A. Phase de partage

1) Processus de distribution: Dans le processus de distribution, le courtier fixe le polynôme $F(x) = F_0 + F_1 x + ... + F_{k-1} x^{k-1}$, où $F_1, \ldots, F_{k-1} \in_R Zq$ et F_0 est le secret à partager. De plus :

1) Chaque participant choisit une clef privée a_i et publie g^{a_i} comme sa clef publique.
2) Le courtier D calcule les parts $s_i = F(i)$, pour $1 \leq i \leq n$.
3) Il calcule un engagement g^{s_i} pour chaque part s_i, pour $1 \leq i \leq n$.
4) Il publie g^{s_i}, pour $1 \leq i \leq n$. Nous notons L l'ensemble des valeurs g^{s_i} publiées.
5) Il envoie une part chiffrée $E_i = s_i * (g^{a_i})^{s_i}$ au participant Pr_i, pour $1 \leq i \leq n$.

2) Processus de vérification: Chaque participant Pr_i, calcule $s_i = E_i * [(g^{s_i})^{a_i}]^{-1}$ et vérifie ensuite que $g^{s_i} \in L$, sinon, le participant présente une plainte contre le courtier.

3) Processus de résolution de conflits: Dans le cas de plainte, la tierce partie, R, peut voter contre le courtier D ou contre le participant A. Dans notre schéma, le courtier D et le participant A essaient de prouver leur honnêteté à R en utilisant le protocole suivant :

1) A choisit $a \in_R Zq$ et publie g^a comme sa clef publique.
2) R choisit $r \in_R Zq$ et publie g^r comme sa clef publique.
3) D calcule $\lambda_1 = s_A * [g^{(a+r)}]^{s_A}$ et $\lambda_2 = s_A * (g^a)^{s_A}$ et publie ces valeurs. Nous notons la première équation Eq_1 et la seconde Eq_2.
4) R calcule $\alpha = \lambda_1/\lambda_2$ et vérifie que $(\alpha^{1/r} = g^{s_A}) \in L$. Si ceci n'est pas vrai, R conclut que le courtier D ment et le processus de résolution de conflits est arrêté.
5) A calcule $s_A = \lambda_2/[(g^{s_A})^a]$ et vérifie que $g^{s_A} \in L$. Si ceci est vrai, il envoie une confirmation à R et le processus de résolution de conflits est arrêté, sinon, il envoie a à R.
6) R vérifie que g^a est la clef publique de A. Si ceci n'est pas vrai, il conclut que A ment et le processus de résolution de conflits est arrêté.
7) R calcule $(g^{s_A})^a$ et $e = \lambda_2/(g^{s_A})^a$ ensuite il vérifie si $e = s_A$ et si Eq_1 est vraie pour s_A. Si elle est vraie, A ment, sinon D ment.

B. Phase de reconstruction

1) Processus de preuve d'appartenance: Si un vérificateur veut vérifier que Pr_i est un participant détenant une part, ce dernier doit prouver son appartenance au vérificateur. Notre preuve d'appartenance est la suivante :

1) Le vérificateur choisit $a \in_R Zq$ et envoie g^a au participant.
2) Le participant envoie $R_P = (g^a)^{s_i}$ au vérificateur.
3) Le vérificateur calcule $R_V = (g^{s_i})^a$ (g^{s_i} est une valeur publique).
4) Si $R_V = R_P$, le participant ayant envoyé R_P est le participant qui possède la part s_i.

2) Regroupement des parts: Le secret est reconstruit à partir des parts soumises, comme suit : $s = \sum_{i=1}^{k} w_i s_i$ où $w_i = \sum_{i \neq j} i/(j-1)$ (Coefficient de Lagrange).

V. Sécurité

Nous commençons, ici, par prouver que les propriétés de sécurité du schéma PVSS de Behnad et Eghlidos restent valables pour notre schéma. En effet, nous prouvons que, dans notre schéma, le courtier D ne peut pas tricher en envoyant une part invalide au participant A. D'ailleurs, puisque les

3. Sachant que $C_i = g^{F_i}$, nous avons :

$$\prod_{i=0}^{k-1} C_i^{j^i} = \prod_{i=0}^{k-1} (g^{F_i})^{j^i} = \prod_{i=0}^{k-1} g^{F_i * j^i} = g^{\sum_{i=0}^{k-1} F_i * j^i} = g^{F(j)} = g^{s_j}$$

,(car $s_j = F(j)$).

4. Nous appelons engagement pour une valeur v, la valeur g^v.

engagements pour les parts (g^{s_i}) sont publiques, le participant A va rapidement détecter que la part qu'il a reçue n'appartient pas à la liste L des valeurs publiques. Nous montrons, donc, que A peut prouver ce fait à la tierce partie R durant le processus de résolution de conflits. Ceci revient à prouver le lemme suivant :

Lemme 1: "Le courtier D ne peut pas envoyer une part invalide au participant A". [15]

Preuve : Durant le processus de vérification, un courtier honnête doit calculer $\lambda_1 = s_A * [g^{(a+r)}]^{s_A}$ et $\lambda_2 = s_A * (g^a)^{s_A}$. Cependant, un courtier malicieux peut avoir un autre comportement. En effet, il peut calculer λ_1 et λ_2 en utilisant une part invalide $s\prime_A$.

Pour un courtier malicieux, $\lambda_1 = \alpha_1 * [g^{(a+r)}]^{\beta_1}$ et $\lambda_2 = \alpha_2 * (g^a)^{\beta_2}$ où α_1, α_2, β_1 et β_2 peuvent être remplacés par s_A . Tandis que pour un courtier honnête $\alpha_1 = \alpha_2 = \beta_1 = \beta_2 = s_A$, un courtier malicieux peut remplacer chacune des valeurs α_1, α_2, β_1 et β_2 par s_A ou par $s\prime_A$. Il peut ainsi utiliser 15 combinaisons possibles pour le couple (λ_1, λ_2). Nous regroupons toutes ces combinaisons en huit cas et nous prouvons que le courtier ne peut pas frauder. Tous ces cas sont étudiés dans l'annexe A. ■

Nous prouvons aussi que dans notre schéma, un comportement malicieux d'un participant A qui reçoit une part valide et réclame avoir reçu une part invalide sera détecté. Nous montrons ici que durant le processus de résolution de conflits, le courtier D peut prouver à une tierce partie R que A a fraudé. Nous prouvons donc le lemme suivant :

Lemme 2: "Le participant A ne peut pas affirmer que la part qu'il a reçue n'est pas valide alors qu'elle l'est". [15]

Preuve : Supposons que le participant A a reçu une part valide s_A mais affirme qu'il a reçu une part invalide. Durant le processus de résolution de conflits, la seule information envoyée par A est sa clef a. Si A envoie a' à R au lieu de a, R calcule $g^{a'}$ et vérifie qu'il ne s'agit pas de la clef publique de A. S'il envoie a à R, ceci veut dire que R peut découvrir sa part correcte s_A. Ainsi, si A a reçu une part valide et affirme qu'il a reçu une part invalide, son comportement sera détecté grâce à l'étape 5 du processus de résolution de conflits. De plus, il sera amené à donner sa clef secrète et sa part à R. D'autre part, si le courtier D ment, R découvrira à l'étape 4 ou plus tard à l'étape 7 du processus de résolution de conflits qu'il s'agit d'une part invalide qui n'appartient pas à la liste L. ■

De plus, nous prouvons que :

Lemme 3: "Sous l'hypothèse Diffie-Hellman Calculatoire, une partie non autorisée ne peut pas passer pour un participant légitime".

Preuve : Avant la reconstruction du secret, un processus de preuve d'appartenance est obligatoire. Durant ce processus, pour passer pour un participant légitime possédant la part s_i, la partie non autorisée doit être capable de calculer $(g^a)^{s_i}$ à partir des valeurs publiques g^a et g^{s_i}. Cependant, sous l'hypothèse Diffie-Hellman Calculatoire, ceci n'est pas faisable. ■

D'autre part, nous prouvons le lemme suivant :

Lemme 4: "Sous l'hypothèse Diffie-Hellman Calculatoire, il est impossible de briser le chiffrement des parts".

Preuve : Briser le chiffrement des parts revient à calculer la part $s_\mathbf{i}$ à partir de sa valeur chiffrée $E_\mathbf{i} = s_\mathbf{i} * (g^\mathbf{a})^{\mathbf{s_i}}$. Pour ce faire, il faut calculer $s_\mathbf{i} = E_\mathbf{i} * [(g^a)^{s_i}]^{-\mathbf{1}}$ à partir des valeurs $E_\mathbf{i}$, $g^\mathbf{a}$, $g^{\mathbf{s_i}}$. Ceci revient à calculer $g^{\mathbf{as_i}}$ à partir des valeurs publiques $g^\mathbf{a}$ et $g^{\mathbf{s_i}}$. Rappelons que d'après l'hypothèse Diffie-Hellman Calculatoire, il est infaisable de calculer $g^{\mathbf{as_i}}$ à partir des valeurs publiques $g^\mathbf{a}$ et $g^{\mathbf{s_i}}$. Ainsi, l'adversaire ou la partie non autorisée n'est pas capable de calculer $s_\mathbf{i}$.

D'autre part, pour briser le chiffrement d'une part $s_\mathbf{i}$, l'adversaire doit être capable de calculer $s_\mathbf{i}$ à partir de la valeur publique $g^{\mathbf{s_i}}$. Ceci revient à résoudre le problème du logarithme discret. Vu que calculer le logarithme discret n'est pas faisable dans G_q, l'adversaire est incapable de calculer $s_\mathbf{i}$ à partir de $g^{\mathbf{s_i}}$. ■

Enfin, nous prouvons le lemme suivant :

Lemme 5: "Sous l'hypothèse Diffie-Hellman Calculatoire, une partie non autorisée ne peut pas extraire la part s_i à partir des valeurs publiques λ_1 et λ_2 durant le processus de résolution de conflits".

Preuve : Pour extraire la part s_A, l'adversaire doit calculer s_A à partir des valeurs publiques $\lambda_1 = s_A * [g^{(a+r)}]^{s_A}$ et $\lambda_2 = s_A * g^{a*s_A}$. Ceci revient à calculer $s_A = \lambda_1 * g^{[(a+r)*s_A]^{-1}} = \lambda_1 * [g^{as_A} * g^{rs_A}]^{-1}$ à partir des valeurs publiques λ_1, g^a, g^r et g^{s_A} , ou bien à calculer $s_A = \lambda_2 * [g^{as_A}]^{-1}$ à partir des valeurs λ_2, g^a et g^{s_A}. Dans le premier cas, l'adversaire doit être capable de calculer la valeur g^{a*s_A} à partir des valeurs publiques g^a et g^{sA} ainsi que la valeur g^{r*s_A} à partir des valeurs publiques g^r et g^{s_A}. Dans le second cas, l'adversaire doit être capable de calculer la valeur g^{a*s_A} à partir des valeurs publiques g^a et g^{s_A}. Dans les deux cas, l'adversaire n'est pas capable de calculer s_A sous l'hypothèse Diffie-Hellman Calculatoire. ■

VI. Comparaison

Notre nouveau schéma PVSS est plus simple que le schéma PVSS de Behnad et Eghlidos. En effet, dans notre schéma il est plus facile de vérifier la validité d'une part par le participant qui la détient, par un autre participant ou même par une tierce partie.

Comparé au schéma PVSS de Behnad et Eghlidos, notre schéma nécessite moins d'opérations de calcul dans toutes ses phases et tous ses processus. En effet, durant le processus de distribution, le courtier D n'est pas amené à calculer et publier une clef publique $g^\mathbf{d}$ comme dans le schéma de Behnad et Eghlidos. Ceci revient au fait que D ne peut plus frauder après avoir publié les valeurs $g^{\mathbf{s_i}}$. De même, le processus de preuve d'appartenance est plus simple dans notre schéma. En effet, dans le schéma de Behnad et Eghlidos, le participant doit choisir une clef b, calculer $g^\mathbf{b}$ et l'envoyer au vérificateur. Ces étapes ont été enlevées de notre schéma.

De plus, le fait de publier les engagements pour les parts permet d'éliminer beaucoup de calcul durant les processus de

vérification, de résolution de conflits et de preuve d'appartenance. En effet, les valeurs g^{s_i} étant publiques, elles peuvent être directement utilisées ce qui évitera de les calculer à chaque fois à partir des coefficients du polynôme de partage comme dans le schéma de Behnad et Eghlidos. Nous réduisons ainsi le nombre d'opérations utilisant l'exponentiation modulaire.

Notons que le schéma PVSS de Behnad et Eghlidos permet d'ajouter de nouveaux membres et leurs parts seront calculées et distribuées sans avoir à publier de nouvelles valeurs. Cette propriété n'est pas vérifiée par notre schéma. En effet, dans notre schéma, on peut connaitre à tout moment le nombre de participants et surveiller le comportement du courtier puisque ce dernier ne peut pas distribuer de nouvelles parts valides sans publier de nouveaux engagements.

VII. Conclusion

Tout comme le schéma PVSS de Behnad de Eghlidos, notre nouveau schéma PVSS comprend un processus de résolution de conflits initié en cas de plainte contre le courtier et une preuve d'appartenance explicite qui doit être fournie au début de la phase de reconstruction du secret partagé. Grâce à ces deux processus, personne ne peut frauder. Ainsi, le nouveau schéma PVSS, quoiqu'il soit plus simple, reste aussi sécurisé que le schéma PVSS de Behnad et Eghlidos. De plus, dans notre schéma la publication des engagements pour les parts permet de suivre les traces des changements de la structure d'accès. En effet, l'ajout de nouveaux membres se fait en publiant les engagements pour les nouvelles parts. L'ajout des membres est alors directement visible pour tout le monde.

De ce fait, notre schéma est plus approprié pour les applications où l'on doit suivre les traces des changements de la structure d'accès. Mais il peut être utilisé même si cette dernière est figée.

Dans nos travaux futurs, nous comptons proposer un nouveau schéma PVSS utilisant une fonction de chiffrement moins couteuse.

Aknowledgments

Les auteurs présentent leurs remerciements à Monsieur Hassan Magtouf et Mademoiselle Wafa Neji pour les discussions intéressantes qui ont enrichi ce travail.

Références

[1] Blakley, G. R. : Safeguarding cryptographic keys. In Proc. AFIPS 1979 National Computer Conference, pp. 313-317, (1979).

[2] Shamir, A. : How to share a secret. Communications of the ACM, 22, pp. 612-614, (1979).

[3] Chor, B., Goldwasser, S., Micali, S. and Awerbuch, B : Verifiable secret sharing and achieving simultaneity in presence of faults. Proc. of the 26th Annual IEEE Symp. on the Foundations of Computer Science (FOCS), pp. 383-395, (1985).

[4] Fiat, A. and Shamir, A. : How to prove yourself : Practical solutions to identification and signature problems. In Crypto, pp. 186-194, (1986).

[5] Feldman, P. : A practical scheme for non-interactive verifiable secret sharing. Proc. of the 28th IEEE Symp. On the Foundations of Computer Science, pp. 427-437, (1987).

[6] Pedersen, T. : Non-interactive and information theoretic secure verifiable secret sharing. Crypto'91, LNCS 576, pp. 129-140, (1992).

[7] Stadler, M. : Public verifiable secret sharing. Advances in Cryptology-Eurocrypt'96, U. Maurer ed, 190—199, (1996).

[8] Fujisaki, E. and Okamoto, T. : A practical and provably secure scheme for publicly verifiable secret sharing and its applications. Advances in Cryptology-Eurocrypt'98, pp. 32-47, (1998).

[9] Boudot, F. and Traore, J. : Efficient Publicly Verifiable Secret Sharing Schemes with Fast or Delayed Recovery. 2nd International Conference on Information and Communication Security, pp. 88-102, (1999).

[10] Paillier, P. : Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In Advances in Cryptology - Eurocrypt'99, LNCS 1952, pp. 223-238, (1999).

[11] Schoenmakers, B. : A simple Publicly Verifiable Secret Sharing Scheme and its Application to Electronic Voting. Advances in Cryptology-Crypto'99, Wiener ed, pp. 148-164, (1999).

[12] Young, A. and Yung. M : A PVSS as Hard as Discrete Log and Shareholder Separability. Advances in 4th International Workshop on Practice and Theory in Public Key Cryptosystems, pp. 287-299, (2001).

[13] Ruiz, A and Villar, J. L. : Publicly verifiable secret sharing from paillier's cryptosystem. In WEWoRC, pp. 98-108, (2005).

[14] Yu, J., Kong, F. Y. and Hao, R. : Publicly Verifiable Secret Sharing with Enrollment Ability. In the 8th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, pp. 194-199, (2007).

[15] Behnad, A. and Eghlidos, T. : A New Publicly Verifiable Secret Sharing Scheme. Journal of Science and Technology, Scientia Iranica, Transaction on Computer Science and Electrical Engineering, Vol. 15, No. 2, pp. 246-251, (2008).

[16] Jhanwar, M. P. : A Practical (Non-interactive) Publicly Verifiable Secret Sharing Scheme. ISPEC, pp. 273-287, (2011).

[17] Wu, T. Y. and Tseng, Y. M. : A pairing-based publicly verifiable secret sharing scheme. Journal of Systems Science and Complexity, ISSN 1559-7067, vol. 24, no1, pp. 186-194, (2011).

Annexe A

Cas 1 : Supposons que le courtier D envoie ($\lambda_1 = s\prime_A * [g^{(a+r)}]^{s\prime_A}$, $\lambda_2 = s\prime_A * (g^a)^{s\prime_A}$) ou ($\lambda_1 = s_A * [g^{(a+r)}]^{s\prime_A}$, $\lambda_2 = s_A * (g^a)^{s\prime_A}$). Ces deux couples sont similaires puisque R peut calculer $\alpha = \lambda_1/\lambda_2 = g^{r\ s\prime_A}$ et vérifier que $\alpha^{1/r} = g^{s\prime_A}$ n'est pas une valeur publique. Donc, R va savoir que D ment.

Cas 2 : Supposons que D envoie ($\lambda_1 = s\prime_A * [g^{(a+r)}]^{s_A}$, $\lambda_2 = s\prime_A * (g^a)^{s_A}$). R calcule $\alpha = \lambda_1/\lambda_2 = g^{r\ s_A}$ et vérifie que $\alpha^{1/r} = g^{s_A}$ est une valeur publique. Donc, A calcule $\upsilon = \lambda_2/[(g^{s_A})^a]$ et vérifie que $g^{\upsilon} \in L$. Puisque ceci n'est pas vrai, il envoie a à R. Ce dernier calcule $\upsilon = \lambda_2/[(g^{s_A})^a]$ et vérifie que $\lambda_1 \neq \upsilon * [g^{(a+r)}]^{\upsilon}$. Donc, R conclut que D ment.

Cas 3 : Supposons que D envoie ($\lambda_1 = s_A * [g^{(a+r)}]^{s_A}$, $\lambda_2 = s_A * (g^a)^{s\prime_A}$) ou ($\lambda_1 = s\prime_A * [g^{(a+r)}]^{s_A}$, $\lambda_2 = s\prime_A * (g^a)^{s\prime_A}$). R calcule $\alpha = \lambda_1/\lambda_2 = [(g^a)^{s_A}(g^r)^{s_A}]/(g^a)^{s\prime_A} = g^{(a.s_A - a.s\prime_A + r.s_A)}$ et vérifie si $\alpha^{1/r} = \left(g^{(a.s_A - a.s\prime_A + r.s_A)}\right)^{-r}$ est une valeur publique. Même si cette valeur est égale à la valeur publique g^{s_A}, R va savoir que D ment. En effet, puisque A doit calculer $\upsilon = \lambda_2/[(g^{s_A})^a]$, il va savoir qu'il a reçu une part invalide ($g^{\upsilon} \notin L$). Il va donc envoyer a à R. Ce dernier calcule $\upsilon = \lambda_2/[(g^{s_A})^a]$ et vérifie que cette valeur ne vérifie pas l'équation $\lambda_1 = \upsilon * [g^{(a+r)}]^{\upsilon}$. Donc, R va conclure que D ment.

Cas 4 : Supposons que D envoie ($\lambda_1 = s_A * [g^{(a+r)}]^{s\prime_A}$, $\lambda_2 = s_A * (g^a)^{s_A}$) ou ($\lambda_1 = s\prime_A * [g^{(a+r)}]^{s\prime_A}$, $\lambda_2 = s\prime_A * (g^a)^{s_A}$). R calcule $\alpha = \lambda_1/\lambda_2 =$

$[(g^a)^{s\prime_A}(g^r)^{s\prime_A}]/(g^a)^{s_A} = g^{(a.s\prime_A - a.s_A + r.s\prime_A)}$ et vérifie si $\alpha^{1/r} = (g^{(a.s\prime_A - a.s_A + r.s\prime_A)})^{-r}$ est une valeur publique. Même si cette valeur est égale à la valeur publique g^{s_A}, R va savoir que D ment. En effet, puisque A doit calculer $\upsilon = \lambda_2/[(g^{s_A})^a]$, il va savoir qu'il a reçu une part invalide ($g^\upsilon \notin L$). Il va donc envoyer a à R. Ce dernier calcule $\upsilon = \lambda_2/[(g^{s_A})^a]$ et vérifie que cette valeur ne vérifie pas l'équation $\lambda_1 = \upsilon * [g^{(a+r)}]^\upsilon$. Donc, R va conclure que D ment.

Cas 5 : Supposons que D envoie ($\lambda_1 = s_A * [g^{(a+r)}]^{s_A}$, $\lambda_2 = s\prime_A * (g^a)^{s_A}$). R calcule $\alpha = \lambda_1/\lambda_2 = (s_A/s\prime_A) * (g^r)^{s_A}$ et vérifie si $\alpha^{1/r} = (s_A/s\prime_A)^{1/r} * g^{s_A}$ est une valeur publique. Même si cette valeur est égale à la valeur publique g^{s_A}, R va savoir que D ment. En effet, puisque A doit calculer $\upsilon = \lambda_2/[(g^{s_A})^a]$, il va savoir qu'il a reçu une part invalide ($g^\upsilon \notin L$). Il va donc envoyer a à R. Ce dernier calcule $\upsilon = \lambda_2/[(g^{s_A})^a]$ et vérifie que cette valeur ne vérifie pas l'équation $\lambda_1 = \upsilon * [g^{(a+r)}]^\upsilon$. Donc, R va conclure que D ment.

Cas 6 : Supposons que D envoie ($\lambda_1 = s\prime_A * [g^{(a+r)}]^{s_A}$, $\lambda_2 = s_A * (g^a)^{s_A}$). R calcule $\alpha = \lambda_1/\lambda_2 = (s\prime_A/s_A) * (g^r)^{s_A}$ et vérifie si $\alpha^{1/r} = (s\prime_A/s_A)^{1/r} * g^{s_A}$ est une valeur publique. Même si cette valeur est égale à la valeur publique g^{s_A}, R va savoir que D ment. En effet, puisque A doit calculer $\upsilon = \lambda_2/[(g^{s_A})^a]$, il va savoir qu'il a reçu une part invalide ($g^\upsilon \notin L$). Il va donc envoyer a à R. Ce dernier calcule $\upsilon = \lambda_2/[(g^{s_A})^a]$ et vérifie que cette valeur ne vérifie pas l'équation $\lambda_1 = \upsilon * [g^{(a+r)}]^\upsilon$. Donc, R va conclure que D ment.

Cas 7 : Supposons que D envoie ($\lambda_1 = s_A * [g^{(a+r)}]^{s\prime_A}$, $\lambda_2 = s\prime_A * (g^a)^{s\prime_A}$) ou ($\lambda_1 = s\prime_A * [g^{(a+r)}]^{s\prime_A}$, $\lambda_2 = s_A * (g^a)^{s\prime_A}$). Pour le premier couple de valeurs, R calcule $\alpha = \lambda_1/\lambda_2 = (s_A/s\prime_A) * (g^r)^{s\prime_A}$ et vérifie si $\alpha^{1/r} = (s_A/s\prime_A)^{1/r} * g^{s\prime_A}$ est une valeur publique. Pour le second couple de valeurs, R calcule $\alpha = \lambda_1/\lambda_2 = (s\prime_A/s_A) * (g^r)^{s\prime_A}$ et vérifie si $\alpha^{1/r} = (s\prime_A/s_A)^{1/r} * g^{s\prime_A}$ est une valeur publique. Dans les deux cas, même si cette valeur est égale à la valeur publique g^{s_A}, R va savoir que D ment. En effet, puisque A doit calculer $\upsilon = \lambda_2/[(g^{s_A})^a]$, il va savoir qu'il a reçu une part invalide ($g^\upsilon \notin L$). Il va donc envoyer a à R. Ce dernier calcule $\upsilon = \lambda_2/[(g^{s_A})^a]$ et vérifie que cette valeur ne vérifie pas l'equation $\lambda_1 = \upsilon * [g^{(a+r)}]^\upsilon$. Donc, R va conclure que D ment.

Cas 8 : Supposons que D envoie ($\lambda_1 = s_A * [g^{(a+r)}]^{s_A}$,$\lambda_2 = s\prime_A * (g^a)^{s\prime_A}$) ou ($\lambda_1 = s\prime_A * [g^{(a+r)}]^{s\prime_A}$,$\lambda_2 = s_A * (g^a)^{s_A}$) ou ($\lambda_1 = s_A * [g^{(a+r)}]^{s\prime_A}$, $\lambda_2 = s\prime_A * (g^a)^{s_A}$) ou ($\lambda_1 = s\prime_A * [g^{(a+r)}]^{s_A}$,$\lambda_2 = s_A * (g^a)^{s\prime_A}$).

Pour le premier couple de valeurs, R calcule $\alpha = \lambda_1/\lambda_2 = (s_A/s\prime_A) * [(g^a)^{s_A}(g^r)^{s_A}]/(g^a)^{s\prime_A} = (s_A/s\prime_A) * g^{(a.s_A - a.s\prime_A + r.s_A)}$.

Pour le second couple de valeurs, R calcule $\alpha = \lambda_1/\lambda_2 = (s\prime_A/s_A) * [(g^a)^{s\prime_A}(g^r)^{s\prime_A}]/(g^a)^{s_A} = (s\prime_A/s_A) * g^{(a.s\prime_A - a.s_A + r.s\prime_A)}$.

Pour le troisième couple de valeurs, R calcule $\alpha = \lambda_1/\lambda_2 = (s_A/s\prime_A) * [(g^a)^{s\prime_A}(g^r)^{s\prime_A}]/(g^a)^{s_A} = (s_A/s\prime_A) * g^{(a.s\prime_A - a.s_A + r.s\prime_A)}$.

Pour le quatrième couple de valeurs, R calcule $\alpha = \lambda_1/\lambda_2 = (s\prime_A/s_A) * [(g^a)^{s_A}(g^r)^{s_A}]/(g^a)^{s\prime_A} = (s\prime_A/s_A) * g^{(a.s_A - a.s\prime_A + r.s_A)}$.

Dans les quatre types de couples, même si $\alpha^{1/r}$ est égale à la valeur publique g^{s_A}, R va savoir que D ment. En effet, puisque A doit calculer $\upsilon = \lambda_2/[(g^{s_A})^a]$, il va savoir qu'il a reçu une part invalide ($g^\upsilon \notin L$). Il va donc envoyer a à R. Ce dernier calcule $\upsilon = \lambda_2/[(g^{s_A})^a]$ et vérifie que cette valeur ne vérifie pas l'équation $\lambda_1 = \upsilon * [g^{(a+r)}]^\upsilon$. Donc, R va conclure que D ment.

Ainsi, nous avons présenté une étude exhaustive de tous les cas où un courtier malicieux peut frauder en remplaçant dans Eq_1 et/ou Eq_2 une ou les deux occurrences de la part correcte s_A par une part erronée $s\prime_A$. Nous ajoutons un nouveau cas où D calcule λ_1 et λ_2 en jouant non seulement sur la valeur de s_A mais aussi sur la clef publique du participant A. Ceci revient à choisir une valeur spécifique de $s\prime_A$ comme l'explique le neuvième cas :

Cas 9 : Supposons que D envoie $s\prime_A = s_A/(g^{a's_A - a\ s_A})$ au lieu de s_A. Ainsi, $s\prime_A = s_A * g^{s_A(a-a')} = s_A * g^{a\ s_A} * g^{-a\prime s_A} = \lambda_2 * g^{-a\prime s_A}$. Donc, $\lambda_2 = s\prime_A * (g^{a\prime})^{s_A}$. Même si D calcule λ_1 comme suit : $\lambda_1 = s\prime_A * [g^{(a\prime+r)}]^{s_A}$ tel que $(\lambda_1/\lambda_2)^{1/r} = g^{s_A} \in L$, cette fraude sera détectée par le participant A quand il vérifie que $\lambda_2/[(g^{s_A})^a] = s_A$ à l'étape 5 du processus de résolution des conflits. R va à son tour détecter cette fraude du courtier à l'étape 7 du même processus quand il découvrira que Eq_2 n'est pas vérifiée.