Chapitre III

Les schémas d'identification

"L'art de persuader consiste autant en celui d'agréer qu'en celui de convaincre."

Blaise Pascal

Ce chapitre présente les schémas d'identification que nous avons élaborés. Le premier est basé sur le problème G-SD. Deux variantes sont proposées, afin d'améliorer ses paramètres. Le deuxième schéma est basé sur le problème SD. Leur complexité et leur performance seront étudiées dans le chapitre suivant.

DANS de nombreuses applications (courrier électronique, échanges bancaires, ...), il peut s'avérer nécessaire que l'expéditeur s'assure de l'identité du destinataire avant de commencer une transaction. C'est le problème de l'identification.

Dans la terminologie usuelle, les deux intervenants sont respectivement appelés le *vérifieur* et le *prouveur* et sont représentés par Bob et Alice (ou Victor et Peggy).

Un schéma d'identification est un protocole cryptographique permettant à Alice de prouver, en temps polynomial, son identité à Bob, sans que ce dernier puisse se faire passer pour Alice auprès d'une tierce personne.

De nos jours, les supports plastifiés, au format carte de crédit, sont de plus en plus employés comme moyen d'identification pour distinguer clients, membres ou adhérents. C'est pourquoi, il est supposé, lors de l'élaboration d'un schéma d'identification, que le prouveur dispose d'une faible puissance de calcul et d'une quantité limitée de mémoire. Ces contraintes ne s'appliquent pas au vérifieur.

Il est important de noter ici que l'hypothèse faite sur le prouveur est différente de celle utilisée dans la théorie générale développée par S. Goldwasser, S. Micali et C. Rackoff (cf. paragraphe suivant). En effet, ces derniers supposent que le prouveur possède une puissance illimitée. Or, dans le cadre d'une application pratique, le prouveur est souvent associé à un support mobile de type cartes à puces, sa puissance est donc limitée.

De même, en ce qui concerne le vérifieur, dans certains cas (la télévision cryptée par exemple), sa puissance de calcul est restreinte. Nous ne nous sommes pas placés dans cette optique, cependant on peut facilement montrer que pour les schémas que nous proposons, la complexité des calculs effectués est en $\mathcal{O}(k^2)$, k étant la dimension de la matrice utilisée.

Le problème de base concernant les méthodes d'identification actuelles (cartes à puces, mots de passe, numéros d'identification, ...) provient du fait que le prouveur établit son identité en révélant un secret s qu'il détient (pensez aux quatre chiffres de votre carte bleue !). Ainsi, il suffit qu'un adversaire, coopérant avec un vérifieur malhonnête, récupère cette quantité pour se faire passer pour Alice auprès d'un vérifieur honnête.

§1. Preuve à Divulgation Nulle (Zéro-Knowledge)

En 1985, S. Goldwasser, S. Micali et C. Rackoff [GMR] ont introduit une méthode interactive de preuve. Cette méthode est probabiliste, c'est-à-dire qu'un observateur sera convaincu de la preuve d'un théorème $\mathcal{T}$ avec une grande probabilité (par exemple $1 - \frac{1}{2^n}$). Les auteurs étudient quelle est la quantité minimale d'information qu'un individu doit dévoiler pour convaincre son interlocuteur qu'il connaît la preuve du théorème $\mathcal{T}$. Ils s'intéressent particulièrement au cas où cette quantité est nulle.

Le modèle qu'ils ont élaboré permet de prouver l'appartenance d'un élément I à un certain langage L de la classe $\mathcal{NP}$ (proof of membership). Pour cela, ils supposent que le prouveur possède une puissance de calcul illimitée et que son temps d'exécution est exponentiel, celui du vérifieur est polynomial. Considérons le langage

$$L = \{(m,x), x \in \mathbb{Z}_m^\times \text{n'est pas un résidu quadratique}\}$$

et soit $I = (m,x)$. Il n'existe pas d'algorithme efficace pour décider si un nombre est un résidu quadratique modulo m lorsque la factorisation de m est inconnue. S. Goldwasser, S. Micali et C. Rackoff proposent le protocole suivant permettant à un prouveur de démontrer au vérifieur que le couple (m,x) appartient ou non à L, sans lui dévoiler la factorisation de m.

- B choisit n nombres quelconques $r_1, \ldots, r_n$ dans $\mathbb{Z}_m^\times$,
- B choisit un vecteur aléatoire $(e_1, \ldots, e_n) \in \{0,1\}^n$,
- B envoie à A le vecteur $(t_1, \ldots, t_n)$ où $t_i = x^{e_i} r_i^2 \pmod m$,
- A possédant une puissance de calcul illimitée, peut déterminer les t_i qui sont des résidus quadratiques. Il en déduit donc le vecteur $(r_1, \ldots, r_n)$ de la

façon suivante :

$$r_i = \begin{cases} \sqrt{t_i x^{-1}} \pmod{n} & \text{si } t_i \text{ n'est pas un résidu quadratique,} \\ \sqrt{t_i} \pmod{n} & \text{sinon.} \end{cases}$$

et l'envoie à B qui contrôle la validité de la réponse.

Si $(m, x) \in L$, alors A peut toujours déterminer la séquence $r_1, \dots r_n$. Si $(m, x) \notin L$, alors $(t_1, \dots, t_n)$ est une suite aléatoire de résidus quadratiques et la probabilité pour que A reconstitue la suite $r_1, \dots, r_n$ vaut $\frac{1}{2^n}$.

Cependant, il n'est pas très approprié de parler de "preuve à divulgation nulle" pour un tel protocole, puisqu'un bit d'information est révélé à B : l'appartenance de I à L. Or, avant le début du protocole, B ne connaissait pas le statut de (m, x) par rapport à L.

En 1986, A. Fiat et A. Shamir [FiSh] démontrent l'intérêt pratique des systèmes interactifs de preuve à divulgation nulle, en les appliquant au problème de l'identification. Supposons qu'Alice connaisse, pour une instance donnée, la solution s d'un problème "difficile". En utilisant un tel système, Alice peut démontrer à Bob qu'elle détient s, en ne révélant aucune information concernant ce dernier.

En 1987, U. Feige, A. Fiat et A. Shamir [FFS], ont proposé un nouveau cadre formel pour les systèmes interactifs de preuve à divulgation nulle. Cette nouvelle modélisation permet l'élaboration d'applications pratiques, puisqu'il n'est plus supposé que le prouveur dispose d'une puissance de calcul illimitée. Le prouveur ne démontre plus l'appartenance d'une donnée I à un ensemble L, mais il convainc le vérifieur qu'il connaît le statut de I par rapport à L (proof of knowledge), sans dévoiler cet état à B.

U. Feige, A. Fiat et A. Shamir introduisent, dans leur modèle, les notations suivantes :

- $\overline{A}$ représente un prouveur honnête,
- $\tilde{A}$ représente un fraudeur utilisant un algorithme probabiliste polynomial en temps,
- A représente soit $\overline{A}$, soit $\tilde{A}$,
- $\overline{B}$ représente un vérifieur honnête,
- $\tilde{B}$ représente un vérifieur malhonnête utilisant un algorithme, polynomial en temps, afin d'extraire des informations de $\overline{A}$,
- (X, Y) représente l'exécution du processus d'identification où X est le prouveur et Y le vérifieur.

Les conventions prises sont les suivantes :

- A et B sont considérés comme un couple interactif de machines de Turing polynomiales probabilistes. Elles possèdent en commun un ruban de données I, deux rubans de communication CA et CB, deux rubans privés de travail WA et WB, et deux rubans privés d'aléa RA et RB,

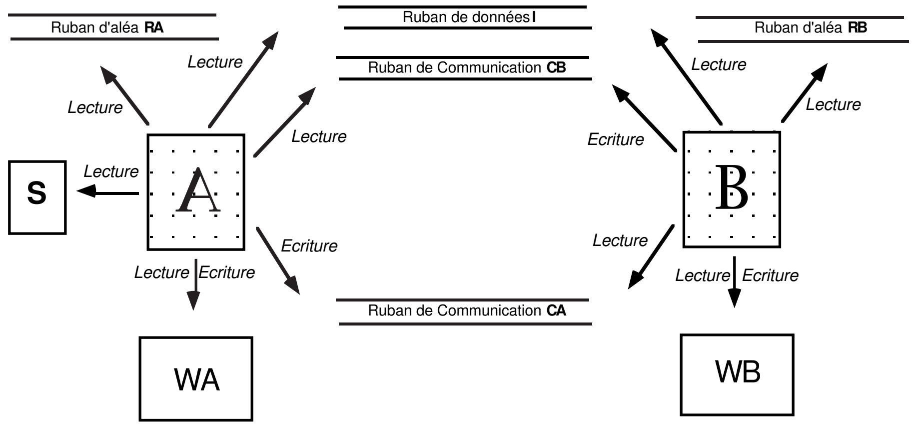

Fig. 1. Un couple interactif de machines de Turing

- $\overline{A}$ a accès à un ruban privé contenant son secret s. Ce secret satisfait un prédicat public, $P(I, s)$, vérifiable en temps polynomial.

  Exemple : Dans le cas du problème G-SD on a :
  . $s = (m, e)$,
  . $I = \{G, x = mG + e, p = \omega(e)\}$,
  . $P(I, s) =$ "s est une solution valide du problème G-SD défini par la donnée I."
- A et B possèdent des états d'exécution et de repos distincts,
- Au début du déroulement du protocole, B est en état de repos et A est en état d'exécution,
- Lorsque A atteint un état de repos, B transite vers un état d'exécution, et vice-versa,
- B possède deux états spéciaux : un état de rejet ("échec") et un état d'acceptation ("succès") à partir desquels aucune transition n'est possible,
- Le protocole s'arrête lorsque B atteint l'un des états mentionnés précédemment,
- Le temps d'exécution de B est polynomialement bornée. C'est-à-dire qu'il existe un polynôme p tel que, si la donnée I est de longueur n, alors B

effectue au plus $p(n)$ transitions avant d'atteindre l'état "succès" (ou l'état "échec"),

- B n'effectue aucune transition lorsqu'il est au repos et que A s'exécute,
- A s'exécute en un temps fini. Il existe une fonction f tel que, si la donnée I est de longueur n, alors, entre un état d'exécution et un état de repos, le temps de calcul de A n'excède pas $f(n)$,
- Il existe un polynôme m tel que A n'écrive jamais plus de $m(n)$ caractères sur le ruban de communication, lorsque la donnée I est de longueur n.

Heuristiquement, un système interactif de preuve à divulgation nulle doit satisfaire trois contraintes :

- Le prouveur ne peut pas tromper le vérifieur, si le prouveur ne connaît pas le secret s, ses chances de convaincre l'autre sont négligeables,
- Le vérifieur ne doit pas obtenir le moindre indice, concernant le secret s, lui permettant de se faire passer pour le prouveur auprès d'une tierce personne,
- Le vérifieur n'apprend rien du prouveur qu'il ne puisse apprendre par lui-même.

Contrairement au modèle de S. Goldwasser, S. Micali et C. Rackoff, pour une même donnée I, $\overline{B}$ peut accepter ou rejeter la preuve de A. Cette décision dépend seulement de l'état de connaissance de A et non pas des propriétés intrinsèques de I. Donc :

- Si A est $\overline{A}$, $\overline{B}$ doit accepter sa preuve avec une forte probabilité, pour toutes les données I satisfaisant le prédicat $P(I, s)$,
- Si A est $\tilde{A}$, $\overline{B}$ doit accepter sa preuve avec une probabilité négligeable, quelle que soit la donnée I.

Si on note $(A, B)[I]$, l'état dans lequel se trouve B après l'exécution de (A, B), lorsque le ruban commun de données est I, on aboutit aux définitions suivantes :

Définition 1.1. *Un couple interactif de machines de Turing polynomiales probabilistes, $\overline{A}$ et $\overline{B}$, est un système interactif de preuve pour le prédicat $P(I, s)$, si pour $0 \leqslant \varepsilon \leqslant \frac{1}{2}$, les conditions suivantes sont satisfaites :*

Complétude : *Pour toute donnée I satisfaisant $P(I, s)$,*

$$Pr\left((\overline{A}, \overline{B})[I] = \text{“succès”}\right) \geqslant 1 - \varepsilon.$$

Consistance : *Pour tout $\tilde{A}$, quelle que soit la donnée I*

$$Pr\left((\tilde{A}, \overline{B})[I] = \text{“succès”}\right) \leqslant \varepsilon.$$

Convention : dans tout ce qui suivra, on appellera *ruban de communication*, la concaténation de toutes les valeurs échangées entre A et B, lors du déroulement d'une preuve. Ces dernières seront considérées comme des variables aléatoires suivant une certaine loi de distribution.

DÉFINITION 1.2. *Un système interactif de preuve est à divulgation nulle si pour tout vérifieur B, il existe une machine de Turing polynomiale, probabiliste, capable de simuler un ruban de communication dont la loi de distribution est parfaitement indiscernable de la loi de distribution d'un ruban de communication provenant d'une véritable identification avec B.*

En d'autres termes, s'il existe un simulateur qui, sans connaître s, peut créer un ruban de communication R_1, indistinguable d'un véritable ruban de communication R_2 alors, étant donné que R_1 ne contient aucune information concernant s, il en est de même pour R_2.

S'il n'est pas possible de construire un tel simulateur, cela signifie qu'une information, dépendante du secret s, est transmise lors du protocole. Dans certains cas, ceci n'affecte en rien la sécurité du protocole car il se peut très bien que cet information soit totalement inexploitable par un attaquant.

Remarque : Le lecteur trouvera une très bonne introduction "informelle" au zéro-knowledge dans [QuGu].

§2. Préliminaires

Lors de l'élaboration d'un schéma d'identification, hormis sa sécurité, trois paramètres essentielles doivent être pris en compte :

- Le volume des quantités échangées (le *débit de transaction*),
- La complexité de calcul du prouveur (le nombre d'opérations nécessaires à un processus d'identification),
- La capacité de stockage nécessaire au prouveur.

Tous les schémas proposés seront évalués, dans le prochain chapitre, à l'aide de ces trois paramètres. C'est ce que nous appellerons les *performances du schéma*.

Depuis les travaux de S. Goldwasser, S. Micali et C. Rackoff, de nombreux schémas d'identification, dont la sécurité dépend d'un problème difficile de la théorie des nombres, ont été élaborés [FiSh, GuQu, Schnn, BrMc, Oka]. Nous les appellerons les *schémas classiques*. Pour ces derniers, un processus d'identification nécessite l'exécution de calculs complexes sur des grands entiers (coûteux en temps et puissance machine) ce qui ne facilite pas leur implémentation. De plus la difficulté des problèmes utilisés (logarithme discret, factorisation) n'est pas prouvée. En contrepartie, le volume mémoire et le débit de transaction de ces schémas sont généralement assez faibles.

La théorie des codes, conjointement avec la théorie de la complexité, ont ouvert une nouvelle voie d'étude pour la mise au point de protocoles cryptographiques n'utilisant que des opérations très simples. C'est S. Harari [Har] qui, en 1988, propose, le premier, un schéma d'identification utilisant les codes correcteurs d'erreurs dont la sécurité dépend d'un problème $\mathcal{NP}$-complet : le problème SD. Depuis, d'autres schémas, basés sur ce même problème, ont été proposés par M. Girault [Gir] et J. Stern [Ste_2, Ste_3]. Dorénavant, nous appellerons ces schémas, les *schémas SD*.

Parmi eux, le deuxième protocole de J. Stern est le plus performant. Il obtient les meilleurs résultats en ce qui concerne les paramètres mentionnés au début du paragraphe. De plus le débit de transaction et la complexité des calculs sont assez faibles pour pouvoir envisager une implémentation sur carte à puces. Le seul inconvénient du schéma est dû à la taille des données devant être stockées par le prouveur.

Il est tout de même précisé par l'auteur que cet inconvénient peut être contourné de la façon suivante : la matrice utilisée étant totalement aléatoire, il est possible de la reconstruire à l'aide d'un générateur aléatoire et d'un germe de t bits. Il suffit donc de stocker le germe au lieu de la matrice. Cependant, ceci nécessite l'intégration d'un tel générateur dans la carte à puces. De plus, pour chaque opération, utilisant la matrice, il faut d'abord regénérer cette dernière, ce qui augmente le nombre de calculs effectués par la carte.

Dans tout ce qui va suivre, toutes les comparaisons entre les différents schémas seront faites en considérant la taille réelle de la matrice utilisée, étant donné que l'alternative consistant à utiliser un générateur aléatoire peut aussi s'appliquer aux solutions que nous proposons.

Dans ce chapitre, différentes alternatives au schéma de J. Stern sont proposées, afin d'en améliorer les performances. Nous nous sommes particulièrement intéressés au problème de la taille des données utilisées afin d'obtenir un schéma dont les caractéristiques soient proches de celles des schémas classiques.

Tous les schémas SD utilisent une matrice binaire $H(k, n)$, quelconque, de rang k. Cette dernière peut être considérée comme la matrice de contrôle d'un code binaire quelconque. Les auteurs suggèrent de choisir $n = 2k$.

Le premier schéma, que nous proposons, utilise une matrice génératrice

$G(k, n)$ d'un code binaire quelconque. Sa sécurité repose sur le problème G-SD. C'est en quelque sorte la version duale du schéma de J. Stern. En considérant les paramètres minimaux de tous les schémas SD, et en respectant la contrainte $n = 2k$, cette nouvelle description permet d'obtenir, à seuil de sécurité équivalent, un débit de transaction inférieur à tous ceux des schémas SD.

De nouveaux paramètres (optimaux) sont calculés afin de réduire, d'une part le nombre d'opérations effectuées par le prouveur, et d'autre part le débit de transaction. La taille des données devant être stockées par le prouveur est aussi diminuée.

Finalement, deux variantes et un autre schéma sont proposées, afin de réduire de manière significative la place mémoire nécessaire au prouveur, ceci afin d'éviter l'alternative consistant à utiliser un générateur aléatoire.

Dans la première variante, nous proposons une représentation différente de la matrice à l'aide de deux éléments d'un corps fini, ce qui permet de la stocker entièrement sans aucune contrainte (en pratique sa taille est équivalente à celle de la donnée publique du schéma de Fiat-Shamir).

Dans la deuxième variante, et le dernier schéma proposé, tous les calculs effectués par le prouveur n'utilisent pas la matrice G, il devient donc inutile de la stocker sur la carte.

Pour ces trois alternatives, la complexité des calculs est elle aussi améliorée. Cependant, certaines structures sont introduites dans la matrice G, ce qui ouvre la voie à de nouvelles stratégies d'attaque.

Les performances de tous ces schémas seront étudiées dans le chapitre suivant.

Le principe général des protocoles que nous allons décrire est le suivant : Alice détient la solution s d'un problème $\mathcal{NP}$-complet pour une instance donnée. Le vérifieur (Bob) pose au prouveur (qui prétend être Alice) une série de questions. Si le prouveur connaît effectivement la quantité s, il peut répondre correctement à toutes ces questions. Dans le cas contraire, la probabilité pour qu'un intrus réponde correctement vaut q $(0 < q < 1)$; cette valeur dépendant essentiellement du nombre de questions que peut poser le vérifieur. Si ce protocole est répété r fois, la probabilité pour que le prouveur soit réellement Alice sera de $1 - q^r$. Ainsi, il est possible de fixer, en fonction de r, un seuil de sécurité au delà duquel Bob sera intimement convaincu de la bonne foi du prouveur. De plus, Bob ne doit pas obtenir, à l'aide des questions et des réponses qui sont échangés au cours d'un processus d'identification, le moindre indice sur le secret d'Alice, excepté le fait qu'elle le détient.

§3. La Mise en Gage

Les protocoles, qui sont proposés, utilisent la notion de mise en gage. Voici un exemple illustrant ce concept :

Afin de s'assurer une renommée internationale, Alice, une magicienne encore peu connue, annonce en public qu'elle peut prédire le tirage du prochain loto. Bob, n'y voyant que son propre intérêt, la met au défi et lui demande de dévoiler les numéros gagnants afin de pouvoir vérifier la véracité de ses propos. Si Alice possède réellement des dons de voyance alors Bob sera rapidement riche. Dans le cas contraire, il ne perdra que sa mise.

Alice qui n'est pas dupe, refuse de dévoiler les bons numéros et propose la solution suivante : elle écrit la série de nombres sur un papier qu'elle enferme dans un coffre-fort dont elle garde la clé. Le coffre-fort est confié à la surveillance de Bob jusqu'au jour du tirage du loto. Ce jour là, Alice remet la clé du coffre à Bob, qui peut alors comparer les nombres inscrits par Alice avec la série gagnante.

Ainsi, la mise en gage, est un protocole où Alice ne dévoile à Bob le contenu d'une quantité secrète qu'au bout d'un certain temps, sans pouvoir, entre-temps, changer la valeur de cette dernière.

Il existe différentes techniques permettant d'élaborer une mise en gage, l'une d'entre elles consiste à utiliser une fonction publique f à sens unique :

- Alice envoie $y = f(x)$ à Bob,
- Alice et Bob exécutent un protocole,
- Alice dévoile x à Bob. Ce dernier vérifie la validité de y.

La fonction f utilisée dans les prochains paragraphes sera une fonction de hachage notée $< . >$ (ceci permet de réduire le nombre de bits transmis). A Crypto'94, M. Girault et J. Stern [GiSt], ont donné les spécifications devant être respectées par une fonction de hachage utilisée dans le cadre des schémas d'identification. Notamment, dans notre cas, il suffira de supposer que la fonction est résistante aux collisions. Nous pouvons alors mieux préciser le principe général des protocoles proposés :

- A transmet à B une série de mises en gage, $c_i =< x_i >$, où une partie des x_i dépendent de son secret s,
- Bob pose une question b_j à Alice,
- Alice dévoile une partie des x_i : $x_{i_1}, \ldots, x_{i_k}$,
- Bob vérifie certaines propriétés des x_{i_j} et contrôle que $< x_{i_j} >= c_{i_j}$.

Il sera démontré que seul un prouveur honnête peut engendrer des quantités c_i toutes valides.

§4. Le modèle de l'Oracle Aléatoire

Cette notion, initialement abordée dans [FiSh], a été largement approfondie par M. Bellare et P. Rogaway [BeRo], en 1993, dans le but de valider l'utilisation des fonctions de hachage dans les protocoles d'identification. L'idée est la suivante :

- Dans un premier temps, le protocole doit être élaboré en supposant que tous les intervenants ont accès à un oracle aléatoire,
- On démontre alors que, sous ces hypothèses, le protocole est sûr,
- Dans un cadre pratique, on remplace les accès à l'oracle aléatoire par le calcul d'une fonction h "bien choisie" (i.e. qui reflète au mieux les comportements de l'oracle).

M. Bellare et P. Rogaway suggèrent d'employer comme fonction h :

- une fonction de hachage dont la sortie est tronquée,
- une fonction de hachage dont la taille de l'entrée est restreinte,
- une fonction de hachage utilisée de façon non standard (ex. $h(x) = MD5(xx)$ [Riv]),
- toute combinaison des trois précédentes suggestions.

Un oracle aléatoire est une fonction $R : \{0,1\}^* \to \{0,1\}^\infty$ où chaque bit de $R(x)$ est choisi uniformément et de façon indépendante.

Le problème qui se pose, dans les protocoles que nous proposons, est dû à l'utilisation des mises en gage. En effet, ces dernières ne seront pas indépendantes les unes des autres.

Pour démontrer qu'une preuve interactive est à divulgation nulle, il faut construire un simulateur M capable de créer un ruban de communication indistinguable d'un ruban provenant d'une réelle exécution d'un processus d'identification. Si le prouveur envoie, dans un premier temps, t mises en gage, M doit pouvoir simuler cet envoi. Or, il sera montré qu'un individu, ne connaissant pas le secret s, ne peut engendrer qu'un nombre limité, ℓ, de mises en gage valides. Pour les $t - \ell$ valeurs restantes, M ne peut engendrer que des valeurs quelconques. Un simulateur M est donc dans l'impossibilité de recréer l'indépendance des t mises en gage.

Dans le modèle de l'Oracle Aléatoire, les mises en gage correspondent à la sortie d'un oracle R. Ce sont donc, par hypothèse, des variables aléatoires, indépendantes, uniformément distribuées. Ainsi, la simulation de ces quantités n'est plus un problème.

§5. Notations, Conventions

En plus des définitions introduites au chapitre i, §5, et des notations du §1, dans les prochains paragraphes, les notations suivantes seront utilisées :

- Soient σ une permutation de l'ensemble $\{1,\ldots,n\}$ et y un vecteur de longueur n *(resp. G une matrice $k \times n$)*, alors $y\sigma$ *(resp. $G\sigma$)* est défini comme étant le vecteur z *(resp. la matrice G')* vérifiant $z_j = y_{\sigma(j)}$ pour $1 \leqslant j \leqslant n$, *(resp. $G^{'}_{i,j} = G_{i,\sigma(j)}$)*
- Soit x un vecteur de longueur n, $\omega(x)$ représente le poids de x, i.e. le nombre d'éléments non nul de x,
- $< x >$ représente l'image par une fonction de hachage de la quantité x,
- $x \square y$ représente la concaténation des chaînes binaires x et y,
- Tous les vecteurs seront considérés comme des vecteurs lignes,
- Soient M une matrice et y un vecteur, alors M^{t} *(resp. y^{t})* est la transposée de la matrice M *(resp. du vecteur y)*,
- $T(RA)$ représente l'arbre d'exécution d'un protocole d'identification, lorsque le prouveur possède un ruban d'aléa égal à RA. Il se construit de la façon suivante : le sommet de l'arbre correspond à la première étape du protocole. Le vérifieur pose une question. Si la réponse du prouveur est valide, on crée un fils. Ensuite le prouveur est réinitialisé [GMR], autant de fois que le nombre de questions pouvant être posées, par le vérifieur, lors d'une étape. Pour chaque réponse valide, un fils est créé. On recommence le procédé, autant de fois qu'il y a d'étapes dans le protocole, et ceci pour chaque nouveau sommet. Cet arbre est un arbre de simulation théorique, il ne reflète pas le déroulement d'un vrai processus d'identification. Cependant, il permet d'établir la preuve de consistance d'un protocole d'identification.

§6. Le schéma G-SD

Les données générales du schéma sont les suivantes :

Quantités publiques	: $G(k,n)$ une matrice binaire quelconque de rang k, une fonction de hachage notée $< . >$.
Quantités secrètes de chaque prouveur	: m un vecteur binaire de longueur k, e un vecteur binaire de longueur n et mG.
Quantités publiques de chaque prouveur	: $x = mG + e$ et $p = \omega(e)$.

(x, p) est appelée *l'identification publique* du prouveur. La quantité mG est stockée par le prouveur afin d'éviter à ce dernier d'avoir à la recalculer à chaque processus d'identification.

6.1. Le protocole

Après le succès international de son film, *"Les six parcs du Jura"*, le célèbre producteur français, Stéphane Chpillebergue (Bob pour les intimes), las des bains de foule parisiens, s'en est allé se réfugier dans sa superbe villa. Sa femme, la tendre et douce Alice, doit le rejoindre dans quelques jours. Afin de déjouer les stratagèmes des journalistes vénaux qui, affublés de perruques ridicules, assaillent son interphone, en se faisant passer pour Alice, afin d'obtenir une interview exclusive, Bob a élaboré un protocole d'identification avec sa femme. Il se compose de r étapes, chacune d'entre elles se déroulant de la façon suivante :

- A calcule aléatoirement :
  - un vecteur binaire u de longueur k,
  - une permutation σ de l'ensemble $\{1, \ldots, n\}$.

  et envoie à B :

$$c_1 =< \sigma >, c_2 =< (u + m)G\sigma >, c_3 =< (uG + x)\sigma >$$

- B envoie b, un élément quelconque de $\{0, 1, 2\}$.
- Si b vaut 0,
  - A dévoile $u + m$ et σ,
  - B vérifie la validité de c_1 et c_2.
- Si b vaut 1,
  - A dévoile $(u + m)G\sigma$ et $e\sigma$,
  - B vérifie la validité de c_2 et c_3 et vérifie que $\omega(e\sigma) = p$.
- Si b vaut 2,
  - A dévoile σ et u,
  - B vérifie la validité de c_1 et c_3.

6.2. Propriétés du schéma

Par rapport au cadre formel défini au §1, B transite vers l'état "échec" lorsque, pour une question b, les quantités c_i dévoilées, ne sont pas valides. Si au bout de r étapes, toutes les quantités dévoilées sont valides alors, B transite vers l'état "succès".

Chaque prouveur $\overline{A}$ est caractérisé par la quantité x $(= mG + e)$, et un entier p $(= \omega(e))$. On a :

. $I = \{G, x, p\}$
. $P(I, s)$ = "s est un couple (m, e) vérifiant $x = mG + e$, $\omega(e) = p$; $G, x, p \in I$."

Remarque : Etant donné que chaque prouveur possède un couple (m, e) distinct (dans le cas contraire, au moins deux prouveurs posséderaient la même identification publique), alors il existe, pour chacun d'entre eux, un unique couple (I, s) tel que $P(I, s)$ soit vrai.

THÉORÈME 6.2.1. *Ce protocole est un système interactif de preuve pour $P(I, s)$.*

Preuve.
Complétude : Tout prouveur connaissant un couple (m, e) valide, pour la donnée I, peut répondre à n'importe quelle question du vérifieur, donc

$$Pr((\overline{A}, \overline{B})[I] = \text{“succès”}) = 1.$$

Consistance :

LEMME 6.2.2. *Si $Pr((\tilde{A}, \overline{B})[I] = \text{“succès”}) \geqslant (\frac{2}{3})^r + \varepsilon$ alors, il existe une machine de Turing M, polynomiale probabiliste qui extrait, avec une probabilité arbitrairement proche de 1, soit des collisions pour la fonction de hachage, soit un couple (m, e) associé aux données publiques.*

Preuve. Soit $T(RA)$ l'arbre d'exécution de $(\tilde{A}, \overline{B})$ correspondant à toutes les questions du vérifieur lorsque $\tilde{A}$ possède un ruban d'aléa RA. A chaque étape, $\overline{B}$ a le choix entre 3 questions. En premier lieu, nous allons montrer qu'à partir d'un sommet possédant trois fils, on peut exhiber soit un couple valide (m, e), soit une collision pour la fonction de hachage. Nous montrerons alors, q'une machine de Turing M, polynomiale probabiliste, peut trouver, avec une forte probabilité, un tel sommet dans $T(RA)$.

Soit V un sommet contenant 3 fils. Ceci correspond à une situation où trois quantités c_1, c_2, c_3 ont été calculées et où $\tilde{A}$ a pu répondre aux trois questions du vérifieur. Soit $u_0 + m_0$ et σ_0 les réponses à la question $b = 0$, y_1 et e_1 les réponses à la question $b = 1$, et u_2, σ_2 les réponses à la question $b = 2$. On a

$$\begin{array}{lll} < \sigma_0 > & = c_1 = & < \sigma_2 > \\ \omega(e_1) & = p & \\ < (u_0 + m_0) G \sigma_0 > & = c_2 = & < y_1 > \\ < y_1 + e_1 > & = c_3 = & < (u_2 G + x) \sigma_2 > \end{array}$$

Ainsi, à moins d'avoir trouvé une collision pour la fonction de hachage, on a :

$$x = (u_0 + m_0 + u_2)G + e_1{\sigma_0}^{-1},$$

où $e_1{\sigma_0}^{-1}$ est un mot de longueur n et de poids p. Donc le couple $(u_0 + m_0 + u_2, e_1{\sigma_0}^{-1})$ est un couple secret valide et peut être utilisé afin d'usurper l'identité de $\overline{A}$.

Finalement, l'hypothèse du lemme, implique que

$$Pr(T(RA)\text{possède un sommet ayant 3 fils}) \geqslant \varepsilon.$$

En effet, considérons RA comme un ensemble de μ éléments, où $\tilde{A}$ choisit aléatoirement les quantités qu'il transmet, et notons Q l'ensemble $\{0, 1, 2\}$. Ces deux ensembles sont considérés comme des espaces probabilisés munis de la distribution uniforme.

Un couple $(c, b) \in (RA \times Q)^r$ représente donc les mises en gage et les questions et réponses échangées, lors d'un processus d'identification, entre $\tilde{A}$ et $\overline{B}$. On qualifiera (c, b) du terme "valide", si l'exécution de $(\tilde{A}, \overline{B})$ aboutit à un succès.

Soit V, le sous ensemble de $(RA\times Q)^r$ composé de tous les couples valides. L'hypothèse du lemme signifie que

$$\frac{\text{card}(V)}{\text{card}((RA \times Q)^r)} \geqslant (\frac{2}{3})^r + \varepsilon.$$

Considérons Ω_r, le sous ensemble de RA^r, vérifiant

- Si $c \in \Omega_r$, alors $2^r + 1 \leqslant \text{card}\{b, (c, b) \text{ soit valide }\} \leqslant 3^r$,
- Si $c \in RA^r \setminus \Omega_r$, alors $0 \leqslant \text{card}\{b, (c, b) \text{ soit valide }\} \leqslant 2^r$.

Alors, $V = \{(c, b) \text{ valides}, c \in \Omega_r\} \cup \{(c, b) \text{ valides}, c \in RA^r \setminus \Omega_r\}$, donc

$$\text{card}(V) \leqslant \text{card}(\Omega_r)3^r + (\mu^r - \text{card}(\Omega_r))2^r.$$

D'où

$$\begin{aligned}\frac{\text{card}(V)}{\text{card}((RA \times Q)^r)} &\leqslant (\frac{\text{card}(\Omega_r)}{\text{card}(RA^r)} + 2^r(3^{-r} - \frac{\text{card}(\Omega_r)}{\text{card}((RA \times Q)^r)}) \\ &\leqslant \frac{\text{card}(\Omega_r)}{\text{card}(RA^r)} + \left(\frac{2}{3}\right)^r.\end{aligned}$$

On en déduit que

$$\frac{\text{card}(\Omega_r)}{\text{card}(RA^r)} \geqslant \varepsilon.$$

Ce qui signifie que la probabilité pour que les quantités, choisis aléatoirement par un fraudeur, satisfassent au moins $2^r + 1$ questions du vérifieur, est supérieure à ε. Or, si plus de $2^r + 1$ questions sont satisfaites, ceci signifie que $T(RA)$ possède au moins $2^r + 1$ feuilles, i.e. qu'il possède au moins un sommet avec trois fils. En effet si aucun sommet de $T(RA)$ ne possède trois fils, alors ce dernier ne peut avoir, au plus, que 2^r feuilles. Donc,

$$Pr(T(RA)\text{possède un sommet ayant 3 fils}) \geqslant \varepsilon.$$

En réinitialisant $\frac{1}{\epsilon}$ fois $\tilde{A}$, on peut trouver, avec une probabilité constante, un arbre $T(RA)$ dont un sommet possède trois fils. En répétant suffisamment ce procédé, cette probabilité peut être rendue arbitrairement proche de 1. □

La première conclusion de ce lemme implique que $< \; . \; >$ n'est pas résistante aux collisions. La dernière conclusion contredit l'hypothèse de non-solvabilité, en temps polynomial, du problème G-SD. On en déduit que pour tout I,

$$Pr((\tilde{A}, \overline{B})[I] = \text{“succès”}) \leqslant \left(\frac{2}{3}\right)^r.$$

□

THÉORÈME 6.2.3. *Dans le modèle de l'oracle aléatoire, ce protocole est un système interactif de preuve, à divulgation nulle, pour $P(I, s)$.*

Preuve. Afin de reproduire le comportement d'un vérifieur malhonnête, il faut supposer que ce dernier adoptera une certaine stratégie en fonction des mises en gage que le prouveur lui dévoile. Même si, en pratique, $\tilde{B}$ ne peut, à priori, rien déduire de ces valeurs (si la fonction de hachage a de bonnes propriétés), il est nécessaire, dans un cadre théorique de prendre cette éventualité en compte afin de prouver la sécurité du protocole. On notera $St(c_1, c_2, c_3)$ la stratégie d'un vérifieur quelconque. On a $St(c_1, c_2, c_3) \in \{0, 1, 2\}$. Notons

$$\begin{array}{cccc} \phi_m\colon & \mathbb{F}_2^k & \longrightarrow & \mathbb{F}_2^k \\ & u & \longmapsto & u + m \end{array}$$

$$\begin{array}{cccc} \psi\ : & \mathbb{F}_2^k & \longrightarrow & \mathbb{F}_2^n \\ & u & \longmapsto & uG. \end{array}$$

ϕ_m est un automorphisme de $\mathbb{F}_2^k$ et si C est le code engendré par la matrice G, alors ψ est un isomorphisme de $\mathbb{F}_2^k$ sur C (c'est l'isomorphisme d'encodage).

Voici la machine de Turing, polynomiale probabiliste, M, qui construit un ruban de communication indistinguable d'un ruban provenant d'une réelle identification :

1. M choisit une question, b, quelconque de $\{0, 1, 2\}$,

- Si $b = 0$, M choisit :
  . y un élément quelconque de $\mathbb{F}_{2^k}$,
  . σ une permutation quelconque de $\{1, \ldots, n\}$,

  et calcule $c_1 =< \sigma >$ et $c_2 =< yG\sigma >$ et remplace c_3 par une chaîne aléatoire. Posons $H = c_1 \square c_2 \square c_3$ et $Rep = y \square \sigma$. Il est clair que y et $u + m$ suivent la même loi de distribution. En effet, soit z un élément quelconque de $\mathbb{F}_2^k$,puisque u est un élément quelconque de $\mathbb{F}_2^k$, on a

$$\begin{aligned} Pr(u + m = z) &= Pr(u = \phi_m^{-1}(z)) \\ &= \frac{1}{2^k} \\ &= Pr(y = z). \end{aligned}$$

- Si $b = 1$, M choisit :
  . y un élément quelconque du code C,
  . e' un élément quelconque de $\mathbb{F}_2^n$ de poids p,
  . σ une permutation quelconque de $\{1, \ldots, n\}$,

  et calcule $c_2 =< y\sigma >$ et $c_3 =< (y + e')\sigma >$ et remplace c_1 par une chaîne aléatoire. Posons $H = c_1 \square c_2 \square c_3$ et $Rep = y\sigma \square e'\sigma$. $e'\sigma$ a la même distribution que $e\sigma$, de plus soit z un mot du code C, alors

$$\begin{aligned} Pr((u + m)G = z) &= Pr(u = \phi_m^{-1}(\psi^{-1}(z))) \\ &= 2^{k-n} \\ &= Pr(y = z), \end{aligned}$$

  donc $(u + m)G$ est distribué uniformément parmi les mots du code C et peut être remplacé par y.
- Si $b = 2$, M choisit :
  . y un élément quelconque de $\mathbb{F}_2^k$,
  . σ une permutation quelconque de $\{1, \ldots, n\}$,

  et calcule $c_1 =< \sigma >$ et $c_3 =< (yG + x)\sigma >$ et remplace c_2 par une chaîne aléatoire. Posons $H = c_1 \square c_2 \square c_3$ et $Rep = y \square \sigma$.

2. M fait calculer $b' = St(H)$,
3. Si $b' = b$, alors M écrit sur le ruban H, b et Rep, sinon M retourne en 1.

Alors M simule un ruban de communication, indiscernable d'un ruban issu d'un processus d'identification constitué de r tours, en un nombre moyen de $3r$ tours. □

6.3. Sécurité du schéma

La sécurité du schéma dépend des paramètres n, k, p et r. Connaissant les données publiques relatives à $\overline{A}$, un intrus doit, pour se faire passer pour ce dernier, trouver le couple (m, e) vérifiant $x = mG + e$. Ce problème de recherche est lié au problème $\mathcal{NP}$-complet G-SD. D'après le chapitre ii, les paramètres minimaux pour lesquels, actuellement, une telle recherche s'avère impossible sont

$$n = 512, k = 256, p = 56.$$

Cependant, si seul un prouveur honnête est capable de répondre simultanément aux trois questions de B, d'après le paragraphe précédent, la probabilité de succès d'un fraudeur est bornée par $(\frac{2}{3})^r$. En effet, sans connaître le couple (m, e), il est possible de répondre correctement à deux questions sur trois de la façon suivante :

• Choisir aléatoirement un vecteur binaire u de longueur k, une permutation σ de l'ensemble $\{1, \ldots, n\}$ et remplacer la quantité m par un vecteur arbitraire m'. Envoyer

$$c_1 =< \sigma >, c_2 =< (u + m')G\sigma >, c_3 =< (uG + x)\sigma >$$

Si b vaut 0, alors $u + m'$ et σ sont des réponses correctes. Si b vaut 2, alors σ et u sont des réponses correctes. Si b vaut 1, le fraudeur doit dévoiler les quantités $(u + m')G\sigma$ et e', où e' est un mot binaire de longueur n et de poids p. Si sa réponse est valide, alors $(uG + x)\sigma = (u + m')G\sigma + e'$, i.e $x = m'G + e'\sigma^{-1}$. Ce qui implique que le fraudeur possède une solution du problème G-SD, ce qui est exclu.

• Choisir aléatoirement un vecteur binaire u de longueur k, une permutation σ de l'ensemble $\{1, \ldots, n\}$ et remplacer les quantités m et e par deux vecteurs arbitraires m' et e'. Calculer $x' = m'G + e'$ et envoyer

$$c_1 =< \sigma >, c_2 =< (u + m')G\sigma >, c_3 =< (uG + x')\sigma >$$

Si b vaut 0, alors $u + m'$ et σ sont des réponses correctes. Si b vaut 1, alors $(u + m')G\sigma$ et $e'\sigma$ sont des réponses correctes. Si b vaut 2, le fraudeur doit dévoiler les quantités σ et u. Si sa réponse est valide, alors $(uG + x)\sigma = (uG + x')\sigma$, i.e $(m, e) = (m', e')$. Ce qui implique que le fraudeur possède une solution du problème G-SD, ce qui est exclu.

• Choisir aléatoirement une permutation σ de l'ensemble $\{1, \ldots, n\}$, un vecteur binaire u de longueur k et calculer $uG + x$. Choisir deux vecteurs y' et e' tels que $uG + x = y' + e'$ et $\omega(e') = p$. Envoyer

$$c_1 =< \sigma >, c_2 =< y'\sigma >, c_3 =< (uG + x)\sigma >$$

Si b vaut 1, alors $y'\sigma$ et $e'\sigma$ sont des réponses correctes. Si b vaut 2, alors σ et u sont des réponses correctes. Si b vaut 0, le fraudeur doit dévoiler les quantités σ et $u + m'$. Si sa réponse est valide, alors $(u + m')G\sigma = y'\sigma$. Or

$y' = uG + x + e'$, l'égalité précédente aboutit alors à $x = m'G + e'$. Ce qui implique que le fraudeur possède une solution du problème G-SD, ce qui est exclu.

Les paramètres minimaux assurant la sécurité du système sont donc :

$$n = 512, k = 256, p = 56, r = 35.$$

La complexité des différentes attaques est alors, au moins, de l'ordre de 2^{70}. La probabilité de succès des différentes fraudes est bornée par 10^{-6}, i.e un fraudeur n'a qu'une chance sur un million de satisfaire le protocole.

§7. Paramètres optimaux de G-SD (G-SD opt.)

Afin de démontrer l'efficacité du schéma G-SD, nous l'avons comparé aux autres schémas SD, en respectant la contrainte $n = 2k$ (cf. chapitre suivant). Les paramètres utilisés, sont les paramètres minimaux suggérés par J. Stern pour son schéma. Le débit de transaction de ce dernier dépend essentiellement du nombre de colonnes de la matrice de contrôle H utilisée.

Dans le schéma G-SD, la taille de deux des réponses fournies par A dépend du nombre de lignes de la matrice G. Ainsi, tout en tenant compte des performances des attaques présentées au chapitre ii, il suffit de diminuer la dimension de G, pour réduire le débit de transaction de notre schéma. Notons $\Omega(n, k, p)$, le facteur de travail correspondant à la recherche d'un mot de poids p, dans un code linéaire binaire (n, k), nous avons donc cherché, à n fixé ($n = 512$) :

$$\min\{k, \Omega(n, k, p) > 2^{60}\},$$

en choisissant pour valeur de p, la distance minimale théorique d'un code linéaire $[n, k]$. Cette étude aboutit aux nouveaux paramètres suivants :

$$n = 512, k = 120, p = 114.$$

Par rapport aux paramètres initiaux, ceci permet de diminuer d'environ 35% le nombre de bits à envoyer lorsque $b = 0$ où $b = 2$.

De plus (cf. chapitre suivant), par rapport au schéma G-SD initial, ces nouveaux paramètres impliquent pour le prouveur :

- une réduction de 30% de sa complexité de calcul,
- une diminution de 30% de la taille de la matrice G à stocker.

Le schéma, ainsi obtenu est, parmi tous les schémas SD, le plus performant (pour un seuil de sécurité supérieur à 2^{60}).

Dans le schéma de J. Stern, il n'est pas possible de réduire significativement le nombre de colonnes de H, afin de diminuer le débit de transaction. En effet, pour obtenir un gain de 35%, il faudrait utiliser une matrice H, de dimension supérieure à 60, comportant au plus 350 colonnes. Or, pour de tels paramètres, les attaques probabilistes sont assez efficaces. A titre d'exemple, pour $n = 350$ et $k = 175$, p vaut environ 38 et le facteur de travail d'une attaque est de l'ordre de 2^{49}.

En résumé, dans un cadre pratique, l'avantage du problème G-SD, par rapport au problème SD est le suivant : Pour n et k fixés, les quantités échangées lors de l'exécution du schéma de J. Stern *(resp. du schéma G-SD)* dépendent de n *(resp. de k)*. Or, on a $k < n$, la taille des données sera donc plus ou moins importante selon le schéma utilisé. De plus à n fixé, on peut chercher le k minimal qui, d'une part, assure la sécurité du schéma G-SD et, d'autre part, offre le meilleur débit de transaction.

Un avantage indéniable des schémas SD, par rapport aux schémas classiques provient du fait que les calculs effectués n'utilisent que des opérations binaires de base ("ou"-logique, "et"-logique).

Un des inconvénients majeur est la taille de la donnée publique (en l'occurrence la matrice de contrôle H ou la matrice génératrice G) qui doit être stockée par le prouveur. Dans le cadre d'une application pratique, afin de garantir une complexité d'attaque supérieur à 2^{60}, ceci représente environ 66000 bits (47040 pour G-SD opt.).

Ce volume est élevé, si le support envisagé est une carte à puces, ou tout autre périphérique possédant une faible capacité de stockage, à moins d'utiliser un générateur aléatoire pour engendrer la matrice. Or, afin d'obtenir des schémas compétitifs, par rapport aux schémas classiques, il est nécessaire de trouver une alternative à ce problème de stockage. A titre d'exemple, dans le schéma de Fiat-Shamir [FiSh], la donnée publique consiste en un entier n qui est le produit de deux premiers p et q. La taille de n, suggérée par les auteurs, est de 512 bits.

Le but des deux prochaines variantes du schéma G-SD, que nous proposons, est d'apporter une solution à ce problème. Dans la première, la matrice G est représentée sous une forme plus "compacte". Dans la deuxième, une partie du protocole est modifiée, afin que le prouveur n'effectue plus aucun calcul utilisant la matrice G ; ce qui a pour conséquence qu'il n'est plus nécessaire qu'il la stocke. Cette dernière sera donc placée dans un lieu accessible par le vérifieur.

§8. Le schéma MR G-SD

Ce schéma n'est valable que pour $n = 2k$. Dans le schéma G-SD, l'opération nécessitant le plus de calculs, pour le prouveur, est le calcul de uG, où u est un mot binaire quelconque de longueur k. La complexité de cette opération est en $\mathcal{O}(k^2)$. Ainsi, pour les paramètres initiaux suggérés ($n = 512, k = 256$), environ 2^{17} opérations binaires sont nécessaires pour évaluer uG. De plus, pour effectuer ce calcul, il est nécessaire que le prouveur puisse stocker G ; ou du moins une partie de celle-ci, si elle est mise sous forme systématique (i.e. $G = (I_k \mid M)$). Dans ce cas, le prouveur doit pouvoir stocker au moins k^2 bits.

La variante proposée, permet de représenter la matrice G, sous une forme plus "compacte", en utilisant deux éléments d'un corps fini, d'où les initiales "MR" signifiant "Matrice Réduite". Nous montrons alors, qu'en utilisant la multiplication de Berlekamp (cf. Chap. i, §5.2), le prouveur peut effectuer des calculs avec G sans stocker cette dernière. Il suffira donc que la matrice ne soit conservée que par le vérifieur. Si, de plus, la représentation de la matrice G est bien choisie, il est possible de réduire, d'un facteur 3, le nombre d'opérations nécessaires au calcul de uG, ceci pour un seuil de sécurité équivalent à celui de G-SD.

Pour cela on utilise les propositions suivantes :

Convention : Soit $\beta = \{\beta_1, \ldots, \beta_k\}$ une base de $\mathbb{F}_{2^k}$, dans tout ce qui va suivre nous ferons volontairement la confusion entre l'élément $x = \sum_{i=1}^{k} x_i\beta_i \in \mathbb{F}_{2^k}$ et le vecteur binaire $(x_1, \ldots, x_k) \in \mathbb{F}_2^k$.

Définition 8.1. *Soit $\beta = \{\beta_1, \ldots, \beta_k\}$ une base de $\mathbb{F}_{2^k}$. Soit γ un élément quelconque de $\mathbb{F}_{2^k}$. La β-matrice produit de γ, $[\gamma]_\beta$, est définie comme étant la matrice*

$$[\gamma]_\beta = \begin{pmatrix} \gamma\beta_1 \\ \vdots \\ \gamma\beta_k \end{pmatrix},$$

où $\gamma\beta_i$ est considéré comme un vecteur binaire de longueur k. C'est donc une matrice binaire $k \times k$ inversible.

Exemple : Considérons le corps $\mathbb{F}_{2^3}$ engendré par $p(x) = x^3 + x + 1$. Soit α une racine de $p(x)$ et soit $\beta = \{1, \alpha, \alpha^2\}$, la base de $\mathbb{F}_{2^3}$, on a

$0=(000)$	$\alpha=(010)$	$\alpha^3=(110)$	$\alpha^5=(111)$
$1=(100)$	$\alpha^2=(001)$	$\alpha^4=(011)$	$\alpha^6=(101)$

Choisissons $\gamma = \alpha^4$, alors

$$[\gamma]_\beta = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}.$$

PROPOSITION 8.2. *Soient $\beta = \{\beta_1, \ldots, \beta_k\}$ une base de $\mathbb{F}_{2^k}$ et γ un élément fixe de $\mathbb{F}_{2^k}$. Soit $\rho = \sum_{i=1}^{k} \rho_i \beta_i$ un élément quelconque de $\mathbb{F}_{2^k}$ alors,*

$$\rho\gamma = (\rho_1, \ldots, \rho_k)[\gamma]_\beta.$$

Exemple : Avec les données de l'exemple précédent, pour $\rho = \alpha^5$, on a bien

$$\begin{aligned} \rho\gamma &= \alpha^2 \\ &= \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} \end{aligned}$$

L'idée est alors de remplacer la matrice G par la matrice $([\gamma_1]_{\beta^*}, [\gamma_2]_{\beta^*})$, où :

- γ_1 et γ_2 sont deux éléments du corps fini $\mathbb{F}_{2^k}$,
- β^* est la base trace duale associée à la base β.

Le produit d'un vecteur binaire u de longueur k par la matrice G revient alors à calculer, la décomposition, dans la base β^*, des produits $\tilde{u}\gamma_1$ et $\tilde{u}\gamma_2$, où $\tilde{u} = \sum_{i=1}^{k} u_i \beta_i^*$.

Etant donné que γ_1 et γ_2 sont des constantes du système, il semble naturel d'utiliser la méthode de Berlekamp pour effectuer ces calculs.

Si le polynôme irréductible, choisi pour engendrer $\mathbb{F}_{2^k}$, est un trinôme irréductible, et si la base β est de la forme $\{1, \alpha, \ldots, \alpha^{k-1}\}$ alors, d'après le chapitre i, cor. 5.2.5, le calcul de $\tilde{u}\gamma_i$ dépend essentiellement du β-poids de γ_i. Si $\omega_\beta(\gamma_i)$ est petit par rapport à k, un tel calcul nécessite moins de k^2 opérations binaires. De plus, ces produits peuvent facilement être effectués, en parallèle, à l'aide de deux registres à décalage de longueur k. Il n'est donc plus nécessaire que le prouveur stocke la matrice G, et étant donné que les produits $u\gamma_i$ sont effectués en utilisant la méthode de Berlekamp, il est inutile, lors d'une implémentation sur circuits, de stocker en ROM, les quantités γ_1 et γ_2. Il suffit en effet de câbler les cellules du registre à décalage, effectuant la multiplication dans $\mathbb{F}_{2^k}$, selon les valeurs de ces deux éléments (cf. p. 16). La matrice G sera donc une quantité publique accessible au vérifieur.

Ainsi, si ces deux paramètres sont bien choisis le calcul de uG nécessitera moins de k^2 opérations (à la condition qu'il existe un trinôme irréductible de degré k [Zie]) et les données devant être stockées par le prouveur se réduiront au couple secret (m, e) et à la quantité mG (cf. §6). Pour $k = 255$, il existe des polynômes irréductibles de degré k.

Si le β-poids de γ_i doit être petit par rapport à k, afin de minimiser les calculs du prouveur, il est plus prudent de le prendre suffisamment grand afin que la classe des matrices G de la forme $([\gamma_1]_{\beta^*}, [\gamma_2]_{\beta^*})$ ne forme pas une sous-classe trop restreinte du problème G-SD. Si $\omega_\beta(\gamma_1) = \omega_\beta(\gamma_2) = t$, alors pour $k = 255$ et $t = 80$, cette classe contient environ 2^{450} éléments.

Plus généralement cette dernière contient $\binom{k}{t}^2$ éléments, alors que le nombre de matrices $(k, 2k)$ de rang k est [LiNi, p. 455]

$$2^{\frac{k^2-k}{2}} \prod_{i=0}^{k-1} (2^{k-i} - 1)(2^{2k-i} - 1)(2^{i+1} - 1)^{-1}.$$

Pour $k = 255$ cette quantité vaut environ 2^{130050}.

Nous suggérons donc d'utiliser comme paramètres minimaux :

$$n = 510, k = 255, p = 56, \omega_\beta(\gamma_1) = \omega_\beta(\gamma_2) = 80.$$

Le calcul de uG nécessite alors, environ $2^{15.3}$ opérations et le seuil de sécurité est équivalent à celui du schéma G-SD (si aucune autre attaque, propre aux spécificités de ce schéma, n'est plus efficace que les algorithmes décrits au chapitre ii).

8.1. Sécurité du schéma

Les structures introduites dans G ouvre la voie à de nouvelles stratégies de cryptanalyse. Voici une attaque possible : Soient e_1 et e_2 deux vecteurs binaires de longueur k tel que $e = (e_1, e_2)$. On a $x = mG + e = (m\gamma_1 + e_1, m\gamma_2 + e_2)$. Notons $\mathbb{F}_{2^\ell}$ le plus gros sous-corps de $\mathbb{F}_{2^k}$. Si $m \in \mathbb{F}_{2^\ell}$ et $\mathrm{Tr}_{\mathbb{F}_{2^k}:\mathbb{F}_{2^\ell}}(e_1) = 0$, le schéma n'est pas sûr. En effet, il suffit de remarquer que

$$\mathrm{Tr}_{\mathbb{F}_{2^k}:\mathbb{F}_{2^\ell}}(m\gamma_1 + e_1) = m\mathrm{Tr}_{\mathbb{F}_{2^k}:\mathbb{F}_{2^\ell}}(\gamma_1).$$

Or γ_1 est une quantité publique puisque la première ligne de $[\gamma_1]_{\beta^*}$ est la décomposition de ce dernier dans la base β^*. Il est donc possible de retrouver, à partir de l'égalité précédente, le vecteur m, et $e = x + mG$. La même attaque est efficace si $m \in \mathbb{F}_{2^\ell}$ et $\mathrm{Tr}_{\mathbb{F}_{2^k}:\mathbb{F}_{2^\ell}}(e_2) = 0$.

Pour $k = 255$, on a $\ell = 85$. Afin de déjouer cette attaque, il suffit de choisir $m \in \mathbb{F}_{2^k} \setminus \mathbb{F}_{2^\ell}$, ce qui laisse encore $2^{255} - 2^{85}$ choix possibles pour m. Une autre alternative consiste à choisir k premier.

Une autre voie possible de cryptanalyse nous a été communiqué récemment par J. Stern et fera l'objet d'un futur travail. L'idée est de modéliser différemment le problème sur lequel repose le schéma MR G-SD.

Les données y_1, y_2 et x sont publics et on a $x = (my_1 + e_1, my_2 + e_2)$. On en déduit que

$$y_2(my_1 + e_1) + y_1(my_2 + e_2) = y_1 e_2 + y_2 e_1.$$

Le problème revient donc à résoudre le système suivant :

$$\begin{cases} y_1 e_2 + y_2 e_1 &= z \\ \omega(e_1) + \omega(e_2) &= p \end{cases}, \tag{8.1.1}$$

où y_1, y_2 et z sont des quantités connues et e_1 et e_2 sont les inconnues. On peut de plus “factoriser” ce problème. En effet, la base de $\mathbb{F}_{2^k}$ étant de la forme $\{1, \alpha, \ldots, \alpha^{k-1}\}$, on peut essayer de résoudre simultanément les systèmes

$$\begin{cases} y_1 e_2' + y_2 e_1' &= z/\alpha^j \\ \omega(e_1') + \omega(e_2') &= p \end{cases}. \tag{8.1.2}$$

Si (e_1', e_2') est une solution de (8.1.2), alors, étant donné que la multiplication par α^j consiste en un décalage et en, au plus, j additions entre deux bits (puisque c'est un trinôme qui engendre le corps), dans certains cas $(\alpha^j e_1', \alpha^j e_2')$ sera une solution de (8.1.1).

Il reste donc à déterminer s'il est possible de développer des algorithmes efficaces permettant de résoudre le système (8.1.2).

Nous n'avons pas, à ce jour, trouvé d'autres voies possibles de cryptanalyse. Toute suggestion est la bienvenue !

§9. Le schéma CR G-SD

L'avantage du schéma précédent est qu'il permet d'effectuer des calculs avec la matrice G sans vraiment stocker cette dernière, puisqu'elle correspond en fait à un câblage de certaines cellules d'un registre à décalage. Une faiblesse cryptographique potentielle provient du fait que la matrice utilisée possède une forte structure algébrique.

Afin de trouver une alternative à cette contrainte, nous nous sommes posés la question suivante : Est-il possible d'élaborer un schéma où :

- la matrice G soit moins structurée,
- tous les calculs effectués par le prouveur n'utilisent pas cette matrice.

La solution, proposée, est une variante du schéma G-SD, appelée CR G-SD, utilisant une matrice G moins structurée. Le protocole est modifié, afin que le prouveur n'effectue plus aucun calcul utilisant cette matrice. Seule son identification publique dépendra de cette dernière. Ceci permet de diminuer la complexité de calcul du prouveur, d'où les initiales "CR" signifiant "Calculs Réduits". Etant donné que le prouveur n'utilise plus la matrice G, son stockage par ce dernier devient donc inutile, d'où un gain non négligeable en place mémoire, par rapport au schéma G-SD. Cette dernière sera stockée uniquement par le vérifieur.

Pour cela, une étape du protocole se décomposera en deux parties.

Première partie : Alice utilise la puissance de calcul de Bob pour lui faire calculer certaines quantités dont elle a besoin pour construire c_1, c_2 et c_3. C'est le principe d'utilisation d'un serveur non sécurisé pour effectuer des calculs complexes. En plus des paramètres habituels, assurant la sécurité du schéma, de nouveaux paramètres seront fixés afin que le vérifieur ne puisse extraire aucune information des calculs qu'il effectue pour le prouveur.

Deuxième partie : Le protocole G-SD est exécuté.

Le problème qui se pose est le suivant : dans le schéma initial, le prouveur envoie les quantités $c_2 =< (u + m)G\sigma >$ et $c_3 =< (uG + x)\sigma >$. Ce qui suppose qu'il a, auparavant, calculé la quantité uG. Sans altérer la sécurité du schéma, notre but est d'essayer de faire calculer cette valeur par le vérifieur sans dévoiler u. En effet, dans le cas contraire, lorsque A dévoile $u + m$, alors B en déduit immédiatement m, et $e = x + mG$.

L'idée de base est la suivante : supposons que le prouveur détienne une autre quantité secrète $u \neq 0$ tel que $uG = 0$ (ceci implique forcément que la matrice G utilisée n'est pas de rang k). Soit v un vecteur binaire quelconque, posons $w = u + v$ alors, $vG = wG$. Supposons, qu'au début du protocole, A envoie v à B, et que B lui renvoie $y = vG$, alors A connaît la quantité wG sans avoir dévoilé w. Il peut ensuite envoyer $c_2 =< y\sigma + mG\sigma >$ et $c_3 =< y\sigma + x\sigma >$ et dévoiler, lors des questions de B, la quantité w.

Cette méthode n'est pas fiable. Voici brièvement, comment deux étapes, du protocole, suffisent à exhiber une clef (m, e).

Etape 1 :

- A calcule un vecteur binaire quelconque $v^{(1)}$, et $w^{(1)} = u + v^{(1)}$. A envoie $v^{(1)}$ à B,
- B conserve $v^{(1)}$ et envoie $y = v^{(1)}G$,
- A envoie $c_1 =< \sigma >$, $c_2 =< y\sigma + mG\sigma >$, $c_3 =< y\sigma + x\sigma >$,
- B pose la question $b = 2$,
- A dévoile $w^{(1)}$.
- B peut en déduire $u = v^{(1)} + w^{(1)}$.

Etape 2 :

- A calcule un vecteur binaire quelconque $v^{(2)}$, et $w^{(2)} = u + v^{(2)}$. A envoie $v^{(2)}$ à B,
- B en déduit $w^{(2)} = u + v^{(2)}$ et envoie $y = v^{(2)}G$,
- A envoie $c_1 =< \sigma >$, $c_2 =< y\sigma + mG\sigma >$, $c_3 =< y\sigma + x\sigma >$,
- B pose la question $b = 0$,
- A dévoile $w^{(2)} + m$.
- B en déduit m et $e = x + mG$.

Une première solution évidente est de choisir un élément u différent à chaque étape. Ceci suppose que le prouveur :

- connaisse une matrice génératrice, G', de l'orthogonal de l'espace engendré par G^{t},
- calcule à chaque étape $u^{\mathrm{t}} = xG'$, où x est un vecteur quelconque de longueur $n - k$.

Recourir à cette technique suppose que le prouveur doit stocker une matrice G' de taille équivalente à celle de G, et qu'il effectue un calcul aussi complexe que le calcul de uG. Ce qui est le contraire du but recherché. Voici la solution que nous proposons :

Représentons tout élément x de longueur $2k$, par le couple (x_1, x_2) où les x_i sont des vecteurs de longueur k, que l'on considérera, si nécessaire, comme des éléments du corps $\mathbb{F}_{2^k}$. L'alternative, que nous proposons, consiste à envoyer, à la place du vecteur v, la quantité $(\eta v_1, \eta v_2)$, où η est un élément de $\mathbb{F}_{2^k}$ engendré aléatoirement à chaque étape du protocole. Ceci permet d'éviter l'attaque précédente. En effet, η étant un élément engendré à chaque étape du protocole, lors de la première étape, B récupère les quantités $(\eta^{(1)}v_1^{(1)}, \eta^{(1)}v_2^{(1)})$ et $(\eta^{(1)}w_1^{(1)}, \eta^{(1)}w_2^{(1)})$ à partir desquelles il déduit

$$(\eta^{(1)}u_1, \eta^{(1)}u_2) = (\eta^{(1)}v_1^{(1)}, \eta^{(1)}v_2^{(1)}) + (\eta^{(1)}w_1^{(1)}, \eta^{(1)}w_2^{(1)}).$$

Néanmoins, au cours de la deuxième étape, B ne peut rien déduire de la quantité $(\eta^{(2)}v_1^{(2)}, \eta^{(2)}v_2^{(2)})$, car

$$(\eta^{(2)}v_1^{(2)}, \eta^{(2)}v_2^{(2)}) + (\eta^{(1)}u_1, \eta^{(1)}u_2) \neq (\eta^{(2)}w_1^{(2)}, \eta^{(2)}w_2^{(2)}).$$

Cependant si u satisfait $uG = 0$, et si v et w vérifient $u = v + w$, pour avoir l'égalité $(\eta w_1, \eta w_2)G = (\eta v_1, \eta v_2)G$, il faudrait que

$$uG = 0 \Longrightarrow \forall \eta \in \mathbb{F}_{2^k}, \qquad (\eta u_1, \eta u_2)G = 0. \tag{9.1}$$

Voici une solution possible pour ce problème. Soient γ_1 et γ_2 deux éléments du corps fini $\mathbb{F}_{2^k}$ et soit

$$G' = \begin{pmatrix} [\gamma_1]_\beta \\ [\gamma_2]_\beta \end{pmatrix}.$$

Cette matrice, de rang k, comporte $2k$ lignes et k colonnes. Il existe donc $u = (u_1, u_2) \neq 0$, tel que $uG' = 0$. De plus $uG' = u_1\gamma_1 + u_2\gamma_2$, le membre gauche de cette égalité se trouvant dans $\mathbb{F}_2^k$ et celui de droite dans $\mathbb{F}_{2^k}$ (identifié à $\mathbb{F}_2^k$).

Pour ce type de matrice l'équation (9.1) est vérifiée. Cependant, G' possède une structure assez importante. Le schéma utilisera la matrice $G = G'S$ où $S(k, 2k)$ est une matrice binaire quelconque de rang k, servant à brouiller la structure de G'. Soit u tel que $uG' = 0$, alors $\forall \eta \in \mathbb{F}_{2^k} \quad (\eta u)G = 0$. En effet,

$$\begin{aligned}(\eta u_1, \eta u_2)G &= (\eta u_1, \eta u_2)G'S \\ &= ((\eta u_1, \eta u_2)G')S \\ &= 0.\end{aligned}$$

Remarquons que

$$G'S = \begin{pmatrix} [\gamma_1]_\beta S \\ [\gamma_2]_\beta S \end{pmatrix},$$

où S est une matrice binaire quelconque. Donc G est la juxtaposition de deux matrices $(k, 2k)$ quelconques. Cette construction ne permet pas d'engendrer toutes les matrices $(2k, 2k)$ de rang k, cependant, par rapport au schéma MR G-SD, cette matrice est beaucoup moins structurée.

9.1. Préliminaires

Un centre de certification, ayant la confiance de tous les utilisateurs, calcule :

- une base $\beta_1 = \{1, \alpha, \ldots, \alpha^{k-1}\}$ du corps $\mathbb{F}_{2^k}$,
- une base $\beta_2 = \{1, \xi, \ldots, \xi^{2k-1}\}$ du corps $\mathbb{F}_{2^{2k}}$, et la base trace duale β_2^* associée,
- Deux éléments quelconques, γ_1 et γ_2, de $\mathbb{F}_{2^k}$,
- $G' = \begin{pmatrix} [\gamma_1]_{\beta_1} \\ [\gamma_2]_{\beta_1} \end{pmatrix}$, une matrice binaire $2k \times k$,
- $S(k, 2k)$ une matrice binaire quelconque de rang k,
- $G = G'S$.

et attribue à chaque utilisateur un vecteur u binaire, non nul, de longueur $2k$ vérifiant $uG' = 0$. Puis, il calcule, pour chaque prouveur, la matrice $\hat{G} = [\rho]_{\beta_2^*}(G \mid Q)\pi_2$, où ρ est un élément de $\mathbb{F}_{2^{2k}}$, π_2 est une permutation quelconque de l'ensemble $\{1, \ldots, 4k\}$, et $Q(2k, 2k)$ est une matrice quelconque.

Les données générales du schéma sont les suivantes :

Quantités publiques : La matrice $G(2k, 2k)$ de rang k, une fonction de hachage notée $< . >$.

Quantités secrètes de chaque prouveur : π_2^{-1}, α^k, m un vecteur binaire de longueur $2k$, e un vecteur binaire de longueur $2k$, mG et u.

Quantités publiques de chaque prouveur : $\hat{G}$, $x = mG + e$ et $p = \omega(e)$.

Les matrices S, G' et Q ne sont plus utiles et restent inconnues de tous les utilisateurs.

Remarque : l'élément α^k est stocké afin de permettre au prouveur d'effectuer une multiplication classique dans $\mathbb{F}_{2^k}$.

9.2. Le protocole

En l'an 995, au royaume des Numides, le prince Brian et la jeune Audrey avaient été promis, l'un à l'autre, dès leur naissance. Malheureusement, l'année de leur mariage, une guerre sanglante obligea le jeune prince à partir au combat afin de défendre son royaume. Des années plus tard, après moults batailles, il revint au pays, afin d'être consacré roi et épouser la belle Audrey. Sa vue ayant été très touchée, lors d'une féroce bataille, Brian se sait dans l'impossibilité de reconnaître sa bien aimée. La nouvelle se répand aussitôt dans tout le royaume, et des centaines de courtisanes viennent assaillir le château du futur roi, espérant bien pouvoir prendre place à ses côtés sur le trône. Heureusement, Audrey et Brian, étant très amoureux l'un de l'autre, avaient mis au point un schéma d'identification avant le départ du prince.

Le protocole inclut r étapes, chacune d'entre elles se déroulant de la façon suivante :

- A engendre aléatoirement :
  - un élément, η de $\mathbb{F}_{2^k}$,
  - v un vecteur binaire de longueur $2k$.

Posons $w = u + v$, A calcule $y = (\eta v_1, \eta v_2)$ et envoie à B, la décomposition de $y\rho^{-1}$ dans la base β_2^* (y étant considéré comme un élément du corps $\mathbb{F}_{2^{2k}}$, exprimé dans la base β_2^*),

Remarque : D'après le chapitre i, prop. 5.2.5, ce calcul ne nécessite pas le stockage de ρ^{-1}, ni de l'élément $\xi^{2^{2k}}$. En effet ρ^{-1} étant une constante du système, la décomposition de $y\rho^{-1}$, dans la base β_2^*, sera calculée à l'aide de la multiplication de Berlekamp .

- B renvoie $z = (y\rho^{-1})\hat{G}$,
- A engendre une permutation quelconque σ de $\{1,\dots,2k\}$ et calcule $z\pi_2^{-1} = (yG \mid yQ)$. Or

$$yG = (\eta v_1, \eta v_2)\begin{pmatrix}[y_1]_{\beta_1}\\ [y_2]_{\beta_1}\end{pmatrix} S\,,$$

  de plus

$$\begin{aligned}(\eta v_1, \eta v_2)\begin{pmatrix}[y_1]_{\beta_1}\\ [y_2]_{\beta_1}\end{pmatrix} &= \eta v_1 y_1 + \eta v_2 y_2\\ &= \eta(w_1 + u_1)y_1 + \eta(w_2 + u_2)y_2\\ &= \eta w_1 y_1 + \eta w_2 y_2 + \eta\,\underbrace{(u_1 y_1 + u_2 y_2)}_{=uG'=0}\\ &= (\eta w_1, \eta w2)G'.\end{aligned}$$

  Posons $\tau = (\eta w_1, \eta w_2)$ et $v = \tau G$, alors les $2k$ premiers bits de $z\pi_2^{-1}$ sont égaux à v. A envoie à B

$$c_1 =< \sigma >, c_2 =< (mG + v)\sigma >, c_3 =< (v + x)\sigma >$$

- B envoie un élément quelconque b de $\{0,1,2\}$,
- Si b vaut 0,
  - A dévoile $\tau + m$ et σ,
  - B vérifie la validité de c_1 et c_2.
- Si b vaut 1,
  - A dévoile $(\tau + m)G\sigma$ et $e\sigma$,
  - B vérifie que $\omega(e\sigma) = p$, et contrôle la validité de c_2 et c_3.
- Si b vaut 2,
  - A dévoile τ et σ,
  - B vérifie la validité de c_1 et c_3.

Remarque : Si, au premier abord, le schéma paraît assez compliqué, remarquons que les opérations effectuées par le prouveur se réduisent à des additions et à des multiplications sur un corps fini. La première opération ne pose aucun problème au niveau de l'implémentation. Il en est de même pour la multiplication qui peut être réalisée à l'aide de deux registres à décalage. De plus, le calcul de ηv_1 et ηv_2, peut s'effectuer en parallèle.

La sécurité du schéma dépend des paramètres n, k, p et r. Pour des raisons de performances, le paramètre $\omega_{\beta_2}(\rho^{-1})$ devra être pris en compte. Avant de discuter la valeur de ces paramètres, voici quelques précisions concernant le rôle de $\hat{G}$. Cette matrice permet au prouveur de connaître la quantité $y = \tau G$, sans la dévoiler au vérifieur. En effet, si ce dernier connaissait cette valeur, alors il pourrait calculer, dans le cas où $b = 0$, $e = x + (\tau + m)G + \tau G$.

9.3. Propriétés du schéma

Chaque prouveur $\overline{A}$ est caractérisé par la quantité x $(= mG + e)$, et un entier p $(= \omega(e))$. On a :

. $I = \{G, x, p\}$
. $P(I, s)$ = "s est un couple (m, e) vérifiant $x = mG + e$, $\omega(e) = p$; $G, x, p \in I$."

THÉORÈME 9.3.1. *Ce protocole est un système interactif de preuve pour $P(I, s)$.*

Preuve. La première partie du protocole consiste à faire calculer la quantité $(\eta w_1, \eta w_2)G$ par le vérifieur. Ensuite, c'est le protocole G-SD qui est utilisé. La démonstration est donc identique à celle du théorème 6.2.2. □

THÉORÈME 9.3.2. *Ce protocole n'est pas un système interactif de preuves, à divulgation nulle, pour $P(I, s)$.*

Preuve. Lors de la première phase du protocole, $\overline{A}$ dévoile $y' = y\rho^{-1} = (\eta v_1, \eta v_2)\rho^{-1}$, où v est un vecteur binaire quelconque. Puis, si $b = 2$, $\overline{A}$ dévoile $(\eta w_1, \eta w_2)$, et on a la relation

$$(\eta w_1, \eta w_2) = (\eta u_1, \eta u_2) + \rho y'. \tag{9.3.2}$$

Cette égalité dépend essentiellement des quantités secrètes ρ et u. Ainsi, lors d'une simulation, si y' peut être remplacé par un vecteur quelconque de longueur $2k$ (puisque v est quelconque), la quantité simulant $(\eta w_1, \eta w_2)$ doit vérifier la relation (9.3.2). Ceci est impossible, à moins de connaître les quantités secrètes u et ρ. Cependant, cette relation de dépendance entre y et la quantité dévoilée, lorsque b vaut 2, semble inexploitable. □

9.4. Sécurité du schéma

Deux types d'attaque sont envisageables. La première consiste à retrouver le couple (m, e), à partir de l'identification publique d'un prouveur $\overline{A}$. D'après le chapitre ii, pour

$$k = 255, p = 56.$$

une telle recherche est impossible. De plus, pour $k = 255$, il existe des trinômes irréductibles de degré k et $2k$. Ceci permettra de simplifier les opérations de multiplication effectuées par le prouveur.

Afin de minimiser les calculs effectués par le prouveur (cf. chap. iv), ρ sera choisi tel que le β_2-poids de ρ^{-1} soit petit. Cependant, ce poids doit être suffisamment important afin d'éviter une recherche exhaustive de ρ^{-1}. En effet, la connaissance de ce paramètre permet, lorsque $b = 0$, de retrouver e. Pour un poids t donné, le nombre d'éléments ρ^{-1} possibles est $\binom{2k}{t}$, ce qui, pour $k = 255$ et $t = 15$, donne environ 2^{94} alternatives.

Une dernière attaque, suggérée par J. Stern, montre que le système "fuit". Malgré le fait que la matrice G' soit masquée par la matrice S, certaines informations concernant γ_1 et γ_2 peuvent être obtenues. Cependant ces dernières s'avèrent être inutilisables. L'attaque se déroule de la façon suivante :

- Pour chaque étape i, du protocole, où la question $b = 2$ est posée, conserver les quantités $y'^{(i)}(= y^{(i)}\rho^{-1})$ et $\tau^{(i)} = (\eta^{(i)}w_1^{(i)}, \eta^{(i)}w_2^{(i)})$.
- Répéter ce procédé jusqu'à ce qu'il soit possible de trouver des éléments $\theta_i \in \mathbb{F}_{2^k}$ vérifiants
$$\sum_i \theta_i y'^{(i)} = 0.$$
- Poser $W_1 = \sum_i \theta_i \eta^{(i)} w_1^{(i)}$ et $W_2 = \sum_i \theta_i \eta^{(i)} w_2^{(i)}$
Par définition de w_1 et de w_2, on a
$$(\eta^{(i)}w_1^{(i)}, \eta^{(i)}w_2^{(i)}) = (\eta^{(i)}u_1, \eta^{(i)}u_2) + \rho y'^{(i)},$$

d'où
$$(W_1, W_2) = (\sum_i \theta_i \eta^{(i)} u_1, \sum_i \theta_i \eta^{(i)} u_2).$$

Donc $u_2W_1 + u_1W_2 = 0$. De plus on sait que $uG = 0$, on a donc le système d'équations suivant
$$\begin{cases} u_2W_1 + u_1W_2 = 0 \\ uG \qquad\qquad = 0 \end{cases}, \tag{9.4.3}$$
où u est l'inconnue. Or ce système ne suffit pas à déterminer la valeur de u. En effet, il ne possède pas une solution unique. Il suffit de remarquer que
$$\forall v \in \mathbb{F}_{2^k} \quad u' = (vu_1, vu_2) \text{ est une solution de (9.4.3).}$$

Donc un tel système possède $2^k - 1$ solutions non nulles. Cependant ceci laisse fuir deux informations : les rapports u_1/u_2 et γ_1/γ_2.

Néanmoins, sans connaître la clé secrète (m, e), un fraudeur peut répondre correctement à deux questions sur trois en utilisant les stratégies décrites au paragraphe 6.3. Sa probabilité de succès après r étapes est donc bornée par $(\frac{2}{3})^r$. Ainsi, les paramètres minimaux assurant la sécurité du schéma sont
$$k = 255, p = 56, r = 35, t = 15.$$

Le facteur de travail, de la cryptanalyse du problème G-SD, est alors supérieur à 2^{70} et une tentative de fraude n'a qu'une chance sur un million d'aboutir.

Remarque : Dans ce schéma, chaque prouveur est caractérisé par un vecteur x de longueur $2k$, un entier p et une matrice $\hat{G}(2k, 4k)$. Ce qui suppose que le vérifieur dispose d'une importante capacité de stockage pour conserver ces données. Ceci n'est pas un inconvénient majeur dans la mesure où, il n'est pas irréaliste de supposer que, dans un cadre pratique, le vérifieur est un organisme suffisamment important (une banque, par exemple) pouvant disposer d'une capacité de stockage conséquente. De plus, de nos jours, le prix des mémoires de masse est en baisse constante, et les disques de plusieurs gigas octets sont très répandus. Enfin, ces matrices peuvent être stockées en utilisant une méthode de compression. Lorsqu'un prouveur commence une session d'identification, la matrice qui lui correspond est alors momentanément décompressée.

Dans le prochain schéma, nous nous proposons d'éliminer cette contrainte pour le vérifieur.

§10. Le schéma T-SD

En résumé, par rapport au schéma G-SD, le schéma MR G-SD permet de réduire le volume mémoire et la complexité de calcul du prouveur, mais utilise une matrice très structurée.

Pour les mêmes optimisations, le schéma CR G-SD utilise une matrice plus aléatoire, mais oblige le vérifieur à disposer d'une capacité de stockage très importante.

Dans ce schéma, notre but est d'essayer de définir un protocole où le prouveur n'utilise pas la matrice publique et effectue peu de calculs, sans pour autant pénaliser le vérifieur au niveau de sa capacité de stockage. La solution proposée utilise une instance particulière du problème SD et possède un débit de transaction élevé. Le schéma T-SD propose donc un compromis entre la complexité des calculs, la capacité de stockage du prouveur et du vérifieur, et le volume des informations échangées. Pour cela, nous reformulons le problème SD en utilisant l'opérateur trace défini sur une extension quadratique de $\mathbb{F}_{2^k}$. La méthode n'est donc valable que pour $n = 2k$. C'est ce qui sera supposé tout au long de ce paragraphe. De plus, elle ne sera vraiment efficace que si $\mathbb{F}_{2^n}$ a été engendré par un trinôme irréductible de degré n.

10.1. L'opérateur Trace et le problème SD

Soient $\beta = \{\beta_1, \dots, \beta_n\}$, une base de $\mathbb{F}_{2^n}$, et $x = \sum_{i=1}^n x_i\beta_i$ un élément quelconque de $\mathbb{F}_{2^n}$. Dorénavant, l'élément x sera représenté par le vecteur colonne $(x_1, \dots, x_n)^{\mathrm{t}}$.

$\mathbb{F}_{2^n}$ et $\mathbb{F}_2^n$ étant deux corps finis isomorphes et puisqu'il existe un algorithme polynomial permettant de passer de l'un à l'autre, le problème SD peut être reformulé de la façon suivante :

Nom : SD
Entrée : $\beta = \{\beta_1, \dots, \beta_n\}$ une base de $\mathbb{F}_{2^n}$ sur $\mathbb{F}_2$,
f une application $\mathbb{F}_2$-linéaire de $\mathbb{F}_{2^n}$ dans $\mathbb{F}_{2^k}$, i un élément de $\mathbb{F}_{2^k}$ et p un entier.
Question : Existe-t-il un élément e de $\mathbb{F}_{2^n}$ tel que $f(e) = i$ et $\omega_\beta(e) \leqslant p$?

L'opérateur $\mathrm{Tr}_{\mathbb{F}_{2^n}:\mathbb{F}_{2^k}}(x) = x^{2^k} + x$ est une application $\mathbb{F}_2$-linéaire particulière, nous proposons donc de choisir $f(x) = \mathrm{Tr}_{\mathbb{F}_{2^n}:\mathbb{F}_{2^k}}(\mu x)$, où μ est un élément quelconque de $\mathbb{F}_{2^n} \setminus \mathbb{F}_{2^k}$. Puisque $f(x) \in \mathbb{F}_{2^k}$, on peut associer à la fonction f, la matrice binaire (k, n), de rang k, suivante :

$$H_f = (f(\beta_1), \dots, f(\beta_n)),$$

où chacun des $f(\beta_i)$ est considéré comme une colonne de k éléments sur $\mathbb{F}_2$. Avec les conventions prises,

$$\forall x = \sum_{i=1}^{n} x_i\beta_i, \quad f(x) = H_f(x_1, \dots, x_n)^{\mathrm{t}}.$$

Afin de brouiller la structure de H_f, on utilisera la matrice $H = SH_f$, où $S(k, k)$ est une matrice binaire quelconque de rang k. Remarquons que ceci ne change pas l'espace engendré par les lignes de H_f (i.e. le code dont la matrice génératrice est H_f). Cependant, pour le problème SD, ce sont les propriétés des colonnes de H qui sont importantes, or la multiplication de H_f par S modifie la structure de ces dernières.

D'après [LiNi, p. 455], le nombre de matrices pouvant être construites de cette façon est

$$(2^{2k} - 2^k)2^{\frac{k^2-k}{2}} \prod_{i=0}^{k-1} (2^{k-i} - 1)^2 (2^{i+1} - 1)^{-1}.$$

Pour $k = 255$ cette quantité vaut environ 2^{65533}, alors que le nombre total de matrices $(k, 2k)$ de rang k est environ de 2^{130050}.

L'opérateur $\mathrm{Tr}_{\mathbb{F}_{2^n}:\mathbb{F}_{2^k}}(x)$ est $\mathbb{F}_{2^k}$-linéaire. C'est essentiellement cette propriété qui a permis la mise au point du schéma T-SD.

Dans notre protocole, lors d'une certaine étape, le prouveur devra calculer un élément de $\mathbb{F}_{2^k}$. Voici comment réaliser cette opération afin de minimiser les calculs de ce dernier.

10.2. Calcul interactif dans $\mathbb{F}_{2^n}$ d'un élément de $\mathbb{F}_{2^k}$ (n=2k)

Soit $\beta = \{\beta_1, \ldots, \beta_n\}$ une base de $\mathbb{F}_{2^n}$. D'après la proposition 4.7 du chapitre i, pour calculer un élément de $\mathbb{F}_{2^k}$, il suffit de calculer la quantité $\varepsilon^{2^k} + \varepsilon$, où ε est un élément quelconque de $\mathbb{F}_{2^n}$.

Cette opération nécessite

$$n(\sum_{j=1}^{k} \omega_\beta(\varepsilon^{2^j}) - 1) + \sum_{j=1}^{k} I_\beta(\varepsilon^{2^j}) + n$$

additions binaires.

Pour tout entier j, on a,

$$\omega_\beta(\varepsilon^{2^j}) \geqslant 1 \text{ et } I_\beta(\varepsilon^{2^j}) \geqslant 1,$$

donc le calcul de $\varepsilon^{2^k} + \varepsilon$ nécessite au moins $\frac{n^2+n}{2}$ additions binaires.

Cette estimation n'est pas très bonne, puisque pour de nombreuses valeurs de j, $\omega_\beta(\varepsilon^{2^j})$ et $I_\beta(\varepsilon^{2^j})$ sont strictement plus grands que 1.

Si on considère que $\omega_\beta(\varepsilon^{2^j}) \simeq \frac{n}{2}$, et si on majore $I_\beta(\varepsilon^{2^j})$ par $n-1$, on obtient un nombre moyen d'opérations binaires de l'ordre de

$$\frac{n^3}{4} + \frac{n^2 - n}{2}.$$

Voici comment un protocole interactif peut diminuer le coût de ce calcul. Supposons qu'Alice connaisse un élément η tel que $\omega_\beta(\eta^{-2^k})$ soit petit par rapport à n. Posons $t = \omega_\beta(\eta^{-2^k})$. Le protocole est le suivant :

- Alice choisit ε, un élément quelconque de $\mathbb{F}_{2^n}$ tel que $\omega_\beta(\varepsilon) = t$, et envoie $\varepsilon\eta$ à Bob,
- Bob renvoie $z = (\varepsilon\eta)^{2^k}$,
- Alice calcule $y = \eta^{-2^k} z + \varepsilon = \varepsilon^{2^k} + \varepsilon \in \mathbb{F}_{2^k}$.

Dans le cas où $\mathbb{F}_{2^n}$ est engendré par un trinôme irréductible, alors le nombre d'opérations effectuées par Alice, lors de ce protocole, est : $I_\beta(\varepsilon) + 2n(t-1) + n + I_\beta(\eta^{-2^k})$.

A n fixé, cette valeur croit en fonction de t. On a donc intérêt à choisir t petit pour minimiser cette quantité. Cependant pour des raisons de sécurité (cf. ci-dessous), t ne pourra pas être inférieur à une certaine limite.

Pour $n = 510$, $\omega_\beta(\varepsilon) = \omega_\beta(\eta^{-2^k}) = 20$, et en majorant $I_\beta(\varepsilon)$ et $I_\beta(\eta^{-2^k})$ par $n-1$, Alice effectue environ $2^{14.3}$ additions binaires, alors que le calcul systématique de $\varepsilon^{2^k} + \varepsilon$ nécessite en moyenne 2^{25} opérations.

Cette méthode permet donc à un individu possédant de faibles moyens de calcul, d'utiliser un serveur auxiliaire afin d'effectuer certaines opérations. Dans le cadre d'un schéma d'identification, c'est le vérifieur qui tient ce rôle. On doit donc supposer que l'on travaille avec un serveur non sécurisé ([MKI]) puisque Bob peut essayer d'obtenir des informations sur le résultat du calcul qu'Alice lui soumet. Par exemple, Bob pourrait essayer de retrouver η, à partir de $\eta\varepsilon$, sachant que $\omega_\beta(\varepsilon)$ est petit. Pour $n = 510$ et $\omega_\beta(\varepsilon) = 20$, il y a, à peu près, 2^{118} choix possibles pour ε, ce qui rend cette attaque totalement inefficace.

10.3. Principe

L'idée de base est la même que dans le schéma CR G-SD. Soit $H(k,n)$ une matrice binaire de rang k, Alice désire faire calculer à Bob, la quantité Hy^{t}, sans dévoiler à Bob le vecteur y de longueur n. Si la matrice H est de la forme

$$H = (\mathrm{Tr}_{\mathbb{F}_{2^n}:\mathbb{F}_{2^k}}(\beta_1) \cdots \mathrm{Tr}_{\mathbb{F}_{2^n}:\mathbb{F}_{2^k}}(\beta_n)),$$

avec $\{\beta_1, \dots, \beta_n\}$ une base de $\mathbb{F}_{2^n}$, alors $Hy^{\mathrm{t}} = \mathrm{Tr}_{\mathbb{F}_{2^n}:\mathbb{F}_{2^k}}(y)$. Supposons qu'Alice détienne une quantité secrète $u \in \mathbb{F}_{2^k}$. Soit γ un élément de $\mathbb{F}_{2^k}$, et soit u_1 un vecteur binaire de poids t. Posons $u_2 = u + u_1$, alors $\mathrm{Tr}_{\mathbb{F}_{2^n}:\mathbb{F}_{2^k}}(u_1) = \mathrm{Tr}_{\mathbb{F}_{2^n}:\mathbb{F}_{2^k}}(u_2)$. Donc, si Alice envoie γu_2 à Bob, ce dernier peut calculer $\mathrm{Tr}_{\mathbb{F}_{2^n}:\mathbb{F}_{2^k}}(\gamma u_1)$, sans qu'Alice dévoile γu_1.

Remarque : Le principe exposé est encore valable pour $H' = SH$, où $S(k,k)$ est une matrice binaire de rang k et $H = (\mathrm{Tr}_{\mathbb{F}_{2^n}:\mathbb{F}_{2^k}}(\mu\beta_1) \cdots \mathrm{Tr}_{\mathbb{F}_{2^n}:\mathbb{F}_{2^k}}(\mu\beta_n))$, $\mu \in \mathbb{F}_{2^n} \setminus \mathbb{F}_{2^k}$. Il faut alors choisir u tel que $u\mu \in \mathbb{F}_{2^k}$.

Afin de minimiser les calculs d'Alice, il faut que :
- le calcul de $\gamma \in \mathbb{F}_{2^k}$ ne soit pas complexe,
- le nombre d'opérations nécessaires pour le calcul de γu_i soit minime.

Le protocole décrit au paragraphe précédent apporte une solution à la première contrainte. Il reste à satisfaire la deuxième.

Si on choisit u_1 de poids petit, alors le calcul de γu_1 est réalisable par le prouveur, mais rien ne nous assure alors que u_2 soit de poids petit, ce qui pose un problème pour le calcul de γu_2.

L'alternative consiste à écrire γu_2 sous la forme $(\gamma\rho_1).(\rho_2\rho_1)^{-1}(\rho_2 u_2)$ où ρ_1 et ρ_2 sont des éléments de poids t (t petit). Ces trois produits, nécessitent peu de calculs, puisque chacun d'entre eux comporte au moins un facteur de poids t. Ils peuvent donc être calculés par le prouveur. L'inversion, ainsi que le produit final seront réalisés par le vérifieur.

10.4. Préliminaires

Notations :

- Soit μ un élément de $\mathbb{F}_{2^n} \setminus \mathbb{F}_{2^k}$, $T_\mu(x)$ représente l'opérateur $\mathrm{Tr}_{\mathbb{F}_{2^n}:\mathbb{F}_{2^k}}(\mu x)$,
- Soit $\{\beta_1, \dots, \beta_n\}$ une base de $\mathbb{F}_{2^n}$, à toute application $\mathbb{F}_2$-linéaire f, de $\mathbb{F}_{2^n}$ dans $\mathbb{F}_{2^k}$, on associe la matrice binaire $k \times n$

$$H_f(f(\beta_1) \cdots f(\beta_n)),$$

- Comme cela a été supposé depuis le début, t représentera un entier petit, dont la valeur conditionnera la complexité des calculs du prouveur.

Un centre de certification ayant la confiance de tous les utilisateurs calcule :

- une base $\beta = \{1, \alpha, \dots, \alpha^{n-1}\}$ de $\mathbb{F}_{2^n}$,
- une matrice binaire quelconque $S(k,k)$ de rang k,
- un élément $\mu \in \mathbb{F}_{2^n} \setminus \mathbb{F}_{2^k}$,
- la matrice $H = SH_{T_\mu}$,
- p un entier,

et attribue à chaque utilisateur :

- un élément quelconque $\eta \in \mathbb{F}_{2^n}$ tel que $\omega_\beta(\eta^{-2^k}) = t$,
- un vecteur binaire s de longueur n et de poids p,
- un élément $u \in \mathbb{F}_{2^n}$ vérifiant $T_\mu(u) = 0$.

Les données générales du schéma sont les suivantes :

Quantités publiques	: La matrice $H(k,n)$ de rang k, une fonction de hachage notée $< . >$, et la base β.
Quantités secrètes de chaque prouveur	: s un vecteur binaire de longueur n et de poids p, les éléments η^{-2^k}, u et η.
Quantités publiques de chaque prouveur	: $i = Hs^{\mathrm{t}}$ et $p = \omega(s)$.

Remarque : la base β est rendue publique, afin que le prouveur et le vérifieur puisse effectuer certains calculs dans $\mathbb{F}_{2^n}$. Toutes les multiplications opéreront sur des éléments différents à chaque étape. Ces calculs seront donc effectués en utilisant la méthode classique, ce qui implique que le prouveur devra stocker l'élément α^n. Les quantités μ et S ne sont plus utiles et restent inconnues de tous les utilisateurs.

10.5. Le protocole

1er Juin de l'an 2069, l'univers tout entier attend avec impatience la retransmission de la finale intergalactique du tournoi de SuperBall, opposant les Olympiens de Mars aux Mutants de la planète Valencius. Dans le plus grand secret, Bob Pietax, le manager des Olympiens de Mars, contacte, quelques jours avant la finale, Alice, le médecin personnel de l'équipe des Mutants, afin que cette dernière drogue légèrement toute l'équipe le jour de la rencontre. Alice accepte à la condition que Bob lui envoie, sur un compte anonyme, 12 millions de térops (monnaie interplanétaire de l'époque). La transaction doit s'effectuer à partir d'un terminal intergalactique du Crédit Lunaire. Bob met au point un schéma d'identification avec Alice, afin qu'un intrus ne puisse pas détourner la transaction.

Le protocole comprend r étapes, chacune d'entre elles se déroulant de la façon suivante :

- A calcule ε un élément quelconque de $\mathbb{F}_{2^n}$ tel que $\omega_\beta(\varepsilon) = t$. Puisque A possède un élément η vérifiant $\omega_\beta(\eta^{-2^k}) = t$, il peut utiliser la méthode décrite au paragraphe 10.2, afin de calculer, à l'aide de B, l'élément $\gamma = \varepsilon^{2^k} + \varepsilon \in \mathbb{F}_{2^k}$,
- A calcule aléatoirement $\rho_1, \rho_2, u_1 \in \mathbb{F}_{2^n}$ (tels que $\omega_\beta(\rho_1) = \omega_\beta(\rho_2) = \omega_\beta(u_1) = t$) et $u_2 = u + u_1$, et envoie à B : $\rho_2\rho_1, \gamma\rho_1, \rho_2 u_2$,
- B calcule $x = H(\gamma\rho_1(\rho_1\rho_2)^{-1}\rho_2 u_2)^{\mathrm{t}}$
  $= S(H_{T_\mu}(\gamma u_2)^{\mathrm{t}}) = S(\gamma T_\mu(u_2)^{\mathrm{t}}) = S(H_{T_\mu}(\gamma u_1)^{\mathrm{t}}) = H(\gamma u_1)^{\mathrm{t}}$,
- A engendre une permutation quelconque σ de $\{1, \ldots, n\}$ et envoie à B

$$c_1 =< (\gamma u_1 + s)\sigma >, c_2 =< \sigma >, c_3 =< (\gamma u_1)\sigma >$$

- B envoie un élément quelconque b de $\{0, 1, 2\}$,
- Si b vaut 0,
  - A dévoile $\gamma u_1 + s$ et σ,
  - B vérifie que $H(\gamma u_1 + s)^{\mathrm{t}} = x + i$ et contrôle la validité de c_1 et c_2.
- Si b vaut 1,
  - A dévoile $(\gamma u_1)\sigma$ et $s\sigma$,
  - B vérifie que $\omega(s\sigma) = p$ et contrôle la validité de c_1 et c_3.
- Si b vaut 2,
  - A dévoile γu_1 et σ,
  - B vérifie que $H(\gamma u_1)^{\mathrm{t}} = x$ et contrôle la validité de c_2 et c_3.

10.6. Propriétés du schéma

Chaque prouveur $\overline{A}$ est caractérisé par la quantité i, et un entier p. On a :

. $I = \{H, i, p\}$

. $P(I, s)$ = "s est un vecteur vérifiant $Hs^{t} = i$, $\omega(s) = p$; $H, i, p \in I$."

Remarque : Etant donné que chaque prouveur possède un secret s distinct (dans le cas contraire, au moins deux prouveurs posséderaient la même identification publique), alors il existe, pour chacun d'entre eux, un unique couple (I, s) tel que $P(I, s)$ soit vrai.

Sous l'hypothèse que le problème SD reste difficile à résoudre pour la matrice H utilisée, alors

THÉORÈME 10.6.1. *Ce protocole est un système interactif de preuve pour $P(I, s)$.*

Preuve.

Complétude : Tout prouveur connaissant un vecteur s valide, pour la donnée I, peut répondre à n'importe quelle question du vérifieur, donc

$$Pr((\overline{A}, \overline{B})[I] = \text{"succès"}) = 1.$$

Consistance :

LEMME 10.6.2. *Si $Pr((\tilde{A}, \overline{B})[I] = \text{"succès"}) \geqslant (\frac{2}{3})^r + \varepsilon$ alors, il existe une machine de Turing M, polynomiale probabiliste qui extrait, avec une probabilité arbitrairement proche de 1, soit une collision pour la fonction de hachage, soit un vecteur s associé aux données publiques.*

Preuve. Soit $T(RA)$ l'arbre d'exécution de $(\tilde{A}, \overline{B})$ correspondant à toutes les questions du vérifieur lorsque $\tilde{A}$ possède un ruban d'aléa égal à RA. A chaque étape, $\overline{B}$ a le choix entre 3 questions. En premier lieu, nous allons montrer qu'à partir d'un sommet possédant trois fils, on peut exhiber soit un vecteur s valide, soit une collision pour la fonction de hachage. Nous montrerons alors, q'une machine de Turing M, polynomiale probabiliste, peut trouver, avec une forte probabilité, un tel sommet dans $T(RA)$.

Soit V un sommet contenant 3 fils. Ceci correspond à une situation où trois quantités c_1, c_2, c_3 ont été calculées et où $\tilde{A}$ a pu répondre aux trois questions du vérifieur. Soit $y_0 + s_0$ et σ_0 les réponses à la question $b = 0$, y_1 et e_1 les réponses à la question $b = 1$, et y_2, σ_2 les réponses à la question

$b = 2$. On a

$$\begin{array}{lll} <(y_0+s_0)\sigma_0> & = c_1 = & <y_1+s_1> \\ H(y_0+s_0)^{\mathrm{t}} & = x+i & \\ <\sigma_0> & = c_2 = & <\sigma_2> \\ \omega(s_1) & = p & \\ <y_1> & = c_3 = & <y_2\sigma_2> \\ Hy_2{}^{\mathrm{t}} & = x & \end{array}$$

Ainsi, à moins d'avoir trouvé une collision pour la fonction de hachage, on a :

$$H(s_1\sigma_0^{-1}) = i,$$

où $s_1\sigma_0^{-1}$ est un vecteur de longueur n et de poids p. Il constitue donc une clé secrète valide qui peut être utilisée afin d'usurper l'identité de $\overline{A}$.

Finalement, l'hypothèse du lemme, implique que (cf. Lemme 6.2.2)

$$Pr(T(RA)\text{possède un sommet ayant 3 fils}) \geqslant \varepsilon.$$

Donc, en réinitialisant $\frac{1}{\epsilon}$ fois $\tilde{A}$, on peut trouver, avec une probabilité constante, un arbre $T(RA)$ dont un sommet possède trois fils. En répétant ce procédé, cette probabilité peut être rendue arbitrairement proche de 1. □

La première conclusion de ce lemme implique que $<\ .\ >$ n'est pas résistante aux collisions. La dernière conclusion contredit l'hypothèse de non-solvabilité, en temps polynomial, du problème SD. On en déduit que pour tout I,

$$Pr((\tilde{A},\overline{B})[I] = \text{“succès”}) \leqslant \left(\frac{2}{3}\right)^r.$$

□

THÉORÈME 10.6.3. *Dans le modèle de l'oracle aléatoire, ce protocole n'est pas un système interactif de preuve, à divulgation nulle, pour $P(I,s)$.*

Preuve. Lors de l'exécution d'une étape du protocole, la quantité γu_2 est connue. De plus si $b = 2$, alors l'élément γu_1 est dévoilé. Or ces deux valeurs sont liées par la relation suivante : $\gamma u_1 + \gamma u_2 = \gamma u$, où u est une quantité secrète du prouveur. Donc, lors d'une simulation, il faudrait arriver à produire deux éléments x_1 et x_2 tel que

$$x_1 + x_2 \in \{\gamma u, \gamma = \mathrm{Tr}_{\mathbb{F}_{2^n}:\mathbb{F}_{2^k}}(\varepsilon), \omega_\beta(\varepsilon) = t\}.$$

Ce qui est impossible, car cet ensemble dépend de la quantité u. Cependant, le fait de pouvoir calculer γu, n'apporte rien au vérifieur, car il ne peut en déduire la quantité u, étant donné que γ est la trace d'un élément quelconque, ε, de poids t. □

10.7. Sécurité du schéma

La sécurité du schéma dépend des paramètres n, k, p, r et t. Sa solidité repose sur la difficulté de résolution du problème SD. En effet, afin d'usurper l'identité de $\overline{A}$, il faut résoudre le système $Hs^{\mathrm{t}} = i, \omega(s) = p$. Pour

$$n = 510, k = 255, p = 56,$$

cette attaque ne peut aboutir. De plus, étant donné que, pour $n = 510$, il existe des trinômes irréductibles de degré n, ceci permettra de minimiser le coût des multiplications effectuées par le prouveur.

Une autre stratégie d'attaque est envisageable. Un assaillant peut essayer de retrouver l'un des éléments suivants : $\varepsilon, \eta^{-2^k}, u_1, \rho_1, \rho_2$ puisque le poids de ces éléments respecte une certaine contrainte. La connaissance d'un seul de ces éléments permet de retrouver tous les autres. Dans ce cas, puisque B connaît γu_2, il en déduit, quand b vaut 2, l'élément u. Finalement, lorsque b vaut 0, le secret s est retrouvé puisque $s = \gamma u_1 + s + \gamma u_2 + \gamma u$. Le β-poids de $\varepsilon, \eta^{-2^k}, u_1, \rho_1$ et ρ_2 est égal à t. Ce qui donne $\binom{n}{t}$ choix possibles pour chacune des quantités. Ce paramètre conditionne d'une part, la sécurité du schéma, et d'autre part le nombre d'opérations effectuées par le prouveur. Pour $n = 510$, la valeur $t = 20$ réalise un bon compromis. Ceci donne environ 2^{118} alternatives pour choisir chacun des éléments de poids t et une multiplication par l'un de ces éléments nécessitent environ $2^{13.3}$ opérations binaires.

Pour terminer, nous décrivons une attaque suggérée par J. Stern et nous montrons pourquoi elle ne peut aboutir :

A chaque étape i du protocole, le vérifieur calcule la quantité $\gamma^{(i)} u_2$. De plus, si $b = 2$, le vérifieur dévoile $\gamma^{(i)} u_1$. Le vecteur secret $u = u_1 + u_2$ vérifie alors

$$\begin{cases} Hu^{\mathrm{t}} = 0 & \text{par définition de u,} \\ u_2 \gamma^{(i)} u_1 + u_1 \gamma^{(i)} u_2 = 0. \end{cases} \tag{10.7.1}$$

Cependant ce système d'équations ne suffit pas à déterminer u. En effet, étant donné que l'opérateur T_μ est $\mathbb{F}_{2^k}$-linéaire, alors

$$\forall \beta \in \mathbb{F}_{2^k} \quad \beta u \text{ est une solution de (10.7.1).}$$

Le système possède donc $2^k - 1$ solutions non nulles.

Cependant, sans connaître le secret s, un intrus peut, à chaque étape, définir une stratégie afin de répondre à deux questions sur trois. Avant de définir ces stratégies de fraude, remarquons que tout individu $\tilde{A}$ ne connaissant pas le secret s, peut interagir avec B afin de calculer un élément

$y \in \mathbb{F}_{2^k}$, et peut construire deux éléments yu_1 et yu_2 tel que $H(yu_1)^t = H(yu_2)^t$. Pour cela :

- $\tilde{A}$ choisit un élément ε quelconque de $\mathbb{F}_{2^n}$ et l'envoie à B,
- B renvoie ε^{2^k},
- $\tilde{A}$ en déduit $y = \varepsilon^{2^k} + \varepsilon$,
- $\tilde{A}$ choisit un vecteur u vérifiant $Hu^t = 0$ (ceci implique que $T_\mu(u) = 0$), et décompose ce vecteur sous la forme $u_1 + u_2$, où u_1 a été choisi aléatoirement,
- $\tilde{A}$ envoie alors à B, les quantités $y\rho_1, \rho_1\rho_2, \rho_2 u_2$, où ρ_1 et ρ_2 ont été choisis aléatoirement,
- B en déduit $x = H(yu_2)^t$ et $\tilde{A}$ peut se servir de la quantité yu_1 pour élaborer les stratégies de fraude suivante :

- Choisir σ une permutation quelconque de l'ensemble $\{1, \ldots, n\}$ et remplacer s par un vecteur arbitraire s' vérifiant $Hs'^t = i$. Envoyer

$$c_1 =< (yu_1 + s')\sigma >, c_2 =< \sigma >, c_3 =< (yu_1)\sigma >$$

  Si b vaut 0, alors $yu_1 + s'$ et σ sont des réponses correctes. Si b vaut 2, alors yu_1 et σ sont des réponses correctes. Si b vaut 1, le fraudeur doit dévoiler deux quantités y et z, où $\omega(z) = p$. Si sa réponse est valide, alors $H(z\sigma^{-1}) = i$, or $z\sigma^{-1}$ et z ont même poids, et donc le fraudeur possède une solution du problème SD défini avec la matrice H.

- Choisir σ une permutation quelconque de l'ensemble $\{1, \ldots, n\}$ et remplacer s par un vecteur arbitraire s' vérifiant $\omega(s') = p$. Envoyer

$$c_1 =< (yu_1 + s')\sigma >, c_2 =< \sigma >, c_3 =< (yu_1)\sigma >$$

  Si b vaut 1, alors $(yu_1)\sigma$ et $s'\sigma$ sont des réponses correctes. Si b vaut 2, alors yu_1 et σ sont des réponses correctes. Si b vaut 0, le fraudeur doit dévoiler un vecteur de longueur n, z, et une permutation π. Si sa réponse est valide, alors $\pi = \sigma$, $Hz^t = x + i$, $z\pi = (yu_1)\sigma + s'\sigma$, et $H(yu_1)^t = x$. Ceci implique que $Hs'^t = i$, ce qui signifie que le fraudeur connaît la solution du problème SD.

- Calculer un élément y vérifiant $Hy^t = x + i$. Choisir y' et s' tel que $y = y' + s'$ et $\omega(s') = p$. Remplacer yu_1 par y' et s par s'. Choisir aléatoirement une permutation σ de l'ensemble $\{1, \ldots, n\}$ et envoyer

$$c_1 =< (y' + s')\sigma >, c_2 =< \sigma >, c_3 =< y'\sigma >$$

  Si b vaut 0, $y' + s'$ et σ sont des réponses correctes. Si b vaut 1, $y'\sigma$ et $s'\sigma$ sont des réponses correctes. Si b vaut 2, le fraudeur doit dévoiler y'

et σ. Si sa réponse est valide ceci signifie que $Hy'^{t} = x$, d'où $Hs'^{t} = i$. Or, par hypothèse, $\omega(s') = p$, donc le fraudeur possède une solution du problème SD.

La probabilité de succès d'un fraudeur étant bornée, après r étapes, par $(\frac{2}{3})^r$, les paramètres minimaux assurant la sécurité du système sont :

$$n = 510, k = 255, p = 56, t = 20, r = 35.$$

La complexité des différentes attaques est alors, au moins, de l'ordre de 2^{70}. La probabilité de succès des différentes fraudes est bornée par 10^{-6}, i.e un fraudeur n'a qu'une chance sur un million de satisfaire le protocole.

Etant donné que la matrice utilisée n'est pas quelconque, d'autres stratégies d'attaque sont sûrement envisageables. Contrairement au schéma MR G-SD, nous n'avons pas réussi à caractériser des clés faibles pour ce schéma. Cependant, des études expérimentales semblent montrer que, pour la matrice utilisée dans ce schéma, les performances des algorithmes du chapitre ii sont les mêmes que celles obtenues pour des matrices quelconques. De plus, si initialement la matrice de départ possède une forte structure, puisqu'elle correspond à un certain mot de longueur n sur $\mathbb{F}_{2^k}$, l'étape de démultiplication sur $\mathbb{F}_2$, et de multiplication par une matrice inversible brouille totalement le mot initial. C'est en quelque sorte le même principe que le système Mc Eliece, où toutes les données sont définies sur un corps d'extension et où les quantités publiques sont obtenues par démultiplication sur le corps de base.