PROTOCOLES À DIVULGATION NULLE

Table des matières

Patricia : Si je t'assure, je suis Batwoman.
Victor : Je ne te crois pas.
Patricia : Tu as tors car je ne connais la cachette secrète de Batman et nous sommes les seuls à la connaître.
Victor : Prouve-le !
Patricia : Très bien, je vais te dire où elle se trouve…
Victor : Ha, ha ! Vous êtes faits, je vais tout raconter au Jocker !
Patricia : Malheur !

Le moyen le plus simple pour Patricia de prouver à Victor qu'elle connaît un secret est de le partager avec Victor. Malheureusement, une fois ceci fait, ce n'est plus un secret et Victor peut le dévoiler à tout le monde. L'idéal serait de pouvoir prouver la connaissance du secret sans le dévoiler. C'est ce que proposent de faire les protocoles de preuve à divulgation nulle couramment appelés protocoles zero-knowledge.

Ce type de protocoles est apparu il y a une vingtaine d'années (entre 1983 et 1985) grâce à Goldwasser, Micali et Rackoff. Beaucoup d'études ont été faites dessus. Ainsi, en dehors de leur intérêt pratique, ils ont contribué au développement de certaines zones de la cryptographie et de la théorie de la complexité.

I. Généralités

I.1. Première approche des protocoles zero-knowledge

Avant de définir exactement ce que sont les preuves à divulgation nulle, essayons de comprendre intuitivement et néanmoins assez précisément de quoi il s'agit.

I.1.1. Définition intuitive

Tout d'abord, les protocoles zero-knowledge sont des protocoles cryptographiques :

- Leur mise en œuvre nécessite la présence d'au moins deux parties : un prouveur et un vérifieur que nous appelleront respectivement Patricia (ou plus simplement P) et Victor (ou plus simplement V).
- Ils permettent de réaliser des opérations telles que l'identification, la signature ou l'échange de clefs.

Ils ont de plus quelques propriétés particulières :

- Patricia ne peut tromper Victor en prétendant connaître un secret qu'elle ne possède pas. En réalité, les protocoles zero-knowledge ne permettent pas d'être 100% certain que Patricia ne triche pas, cependant en les itérant plusieurs fois, il est possible de réduire la part d'incertitude autant qu'on peut le souhaiter.
- Victor ne peut rien apprendre d'intéressant de Patricia pendant l'échange même en trichant, c'est à dire en ne suivant pas le protocole. On qualifie par « intéressante » toutes informations que Victor ne peut découvrir tout seul.
- Victor n'ayant rien appris de son échange avec Patricia ne peut pas convaincre une tiers personne qu'il est Patricia même s'il a enregistré l'échange.

Rem : Malgré toutes ces propriétés, les protocoles zero-knowledge (tels que nous allons les définir plus bas) ne sont pas inviolables. Ils ne résistent pas, en effet, aux attaques de types man-in-the-middle où Martin, un être malveillant, agit comme un relais entre Patricia et Victor. Ainsi Patricia et Victor croient se parler l'un à l'autre alors qu'ils parlent en fait à Martin qui pourra commettre les pires crimes en se faisant passer pour Patricia (bien qu'il n'ait rien appris de son secret).
Il n'y a pas de parades « théoriques » à ce type d'attaque. Cependant, en pratique, on peut limiter le temps de réponse de sorte que Martin n'est pas le temps de relayer les questions/réponses entre Patricia et Victor.

I.1.2. Illustration : la caverne d'Ali Baba

L'illustration la plus simple et la plus répandue de la manière dont se déroule un protocole zero-knowledge a été donnée par Jean Jacques Quisquater et Louis Guillou. La caverne d'Ali Baba possède une porte qui ne s'ouvre que lorsqu'on connaît la formule magique. Patricia veut prouver à Victor qu'elle connaît cette formule sans la lui dévoiler.

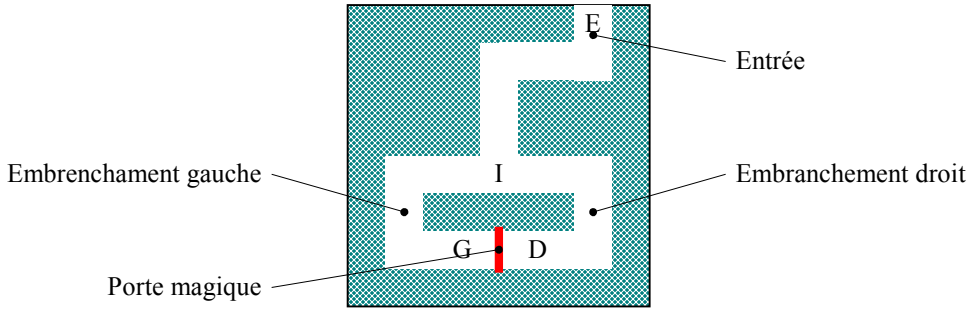


Figure 1: Caverne d'Ali Baba

I.1.2.1. Protocole

Le protocole se déroule en 5 étapes :

1. Victor et Patricia vont à l'entrée de la caverne (E).
2. Patricia rentre et va au fond de la caverne en choisissant aléatoirement soit la branche gauche (G) soit la droite (D).
3. Victor entre et attend à l'intersection des deux branches (I)
4. Victor choisit aléatoirement une des branches (celle de gauche ou celle de droite) et demande à Patricia de ressortir de la caverne par cette branche.
5. Patricia s'exécute en utilisant éventuellement la formule magique.

A la fin de l'étape 5, si Patricia ressort par la mauvaise branche, Victor est sûr que Patricia est une imposteuse. Dans le cas contraire, il commence à croire Patricia. Il peut alors se déclarer satisfait et s'arrêter ou continuer pour renforcer sa foi en Patricia.

I.1.2.2. Preuve à divulgation nulle

Ce protocole est bien une preuve à divulgation nulle.

Voyons comment Patricia pourrait tricher. Si Patricia ne connaît pas la formule magique, il faut qu'elle aille au fond de la caverne par le même embranchement que celui choisit par Victor à l'étape 4. Cependant comme elle ne peut connaître ce choix à l'avance, elle doit compter sur sa chance. Ce faisant, elle n'a qu'une chance sur deux de ressortir par la bonne branche à l'étape 5. Donc si Victor recommence n fois les étapes 1 à 5, Patricia n'a plus qu'une chance sur 2^n de le tromper. Ainsi, il lui suffit de prendre un n assez grand pour s'estimer convaincu par Patricia.

Dans cet exemple simpliste, il est évident que Victor pourra répéter autant de fois le protocole qu'il le veut et tricher comme il lui plaira (sauf à mettre un micro près de la porte), il ne pourra jamais connaître le secret de Patricia. Il ne pourra pas non plus convaincre une tiers personne ni qu'il connaît le secret de Patricia, ni même que Patricia le connaît (bien que lui en soit sûr maintenant).

I.2. Notion de preuve

Après cette approche intuitive des protocoles zero-knowledge et avant de poursuivre plus loin la définition des preuves à divulgation nulle, essayons simplement de préciser ce que l'on entend par « preuve ».

I.2.1. Statiques ou interactives

On peut dire que selon les points de vue, il y a deux types de preuves.

En mathématiques une preuve est plutôt vue comme un enchaînement fixé d'états reliés par des règles. Les états initiaux sont des axiomes, les états suivants sont des théorèmes et les règles sont des règles de dérivation. Le tout est donc un objet statique et généralement, la preuve est au moins aussi importante que ce qu'elle démontre.

Ce n'est cependant pas la seule façon de définir une preuve. En effet, dans la vie courante, une preuve est plus un processus (qui dénote une certaine interactivité) qu'un objet fixé. De plus ce processus est généralement sans importance par rapport à sa conclusion.

Dans les protocoles cryptographiques et en particulier zero-knowledge, c'est la deuxième interprétation de « preuve » qui est utilisée puisqu'ils se basent toujours sur un jeu de question(s)/réponse(s).

I.2.2. Le rôle des intervenants

Le rôle des intervenants (c'est à dire le(s) prouveur(s) et le(s) vérifieur(s)) dans une preuve, aussi bien en mathématiques que dans la vie courante, n'est pas du tout équilibré. Etablir une preuve est toujours beaucoup plus compliqué que la vérifier.

On peut dire en quelque sorte que c'est le prouveur qui fait tout le travail. Cependant le vérifieur n'est pas non plus facultatif car c'est lui qui va valider la preuve. D'ailleurs, il n'est pas inutile de rappeler qu'il doit toujours se montrer méfiant et mettre en doute tout ce que lui dit le prouveur. En effet, si le vérifieur se mettait à faire confiance au prouveur, il n'y aurait tout simplement pas besoin de preuve.

I.2.3. Deux propriétés fondamentales

Il n'est pas évident de mettre en place un système de preuves. En effet, celui-ci doit vérifier deux propriétés fondamentales :

- La validité qui garantit que le vérifieur ne peut pas être trompé par une fausse preuve.
- La complétude qui assure au prouveur que le vérifieur acceptera toutes les preuves valides.

On peut remarquer que ces deux propriétés (nécessaires) sont assez contraignantes. De fait, il n'existe pas de systèmes de preuve 'réaliste' (i.e., dans lequel la vérification peut être réalisée efficacement) pour toutes les affirmations vraies (dans lesquels toutes les affirmations fausses ne pourraient pas être prouvées). Pour notre part, nous nous limiterons aux ensembles de problèmes qui possèdent des systèmes de preuve 'réaliste'.

I.3. Notion de connaissance

La notion de « connaissance » est très importante dans les protocoles de preuve à divulgation nulle. En effet, pour savoir si, durant l'échange d'informations (inévitable), la connaissance du vérifieur à augmentée (i.e., il y a eu divulgation, même partielle, du secret), il faut savoir ce qu'on entend par « connaissance ».

I.3.1. Gagner de la connaissance

Il est difficile de définir ce qu'est la connaissance. Cependant, pour ce qui nous concerne, ce n'est pas nécessaire. En fait, ce qui nous préoccupe c'est quand peut-on dire que l'on a gagné de la connaissance. Il est possible de répondre à cette question sans donner de définition à la connaissance.

Considérons une conversation entre Alice et Bob à propos d'un graphe assez large (connu d'eux deux). Alice connaît tout de ce graphe et Bob lui pose des questions. Si Bob demande à Alice si le graphe est eulérien, alors la réponse d'Alice ne lui apportera aucun connaissance car, à priori, Bob est parfaitement capable de répondre tout seul à cette question (en utilisant le théorème de Euler qui démontre qu'un graphe est eulérien si et seulement si ses sommets sont de degré pair). En revanche, s'il

demande à Alice si le graphe est hamiltonien, alors la réponse d'Alice lui fera gagner de la connaissance car aujourd'hui il n'y a pas de procédure efficace qui permette à Bob de répondre lui-même à sa question (et si $\mathcal{P}\neq\mathcal{NP}$, il n'en existe pas).

Donc on peut dire que Bob gagne de la connaissance si son pouvoir calculatoire a augmenté entre le moment où il a posé sa question et celui où Alice a répondu. Au contraire, si tout ce qu'il a appris de son échange avec Alice aurait pu être calculé par lui-même (avant l'échange), alors il n'a gagné aucune connaissance.

On peut se demander comment Alice peut tout connaître sur le graphe et en particulier savoir s'il est hamiltonien. Cela n'a pas beaucoup d'importance et nous verrons plus tard que cette question est réglée dans la définition formelle des preuves interactives (donc des preuves à divulgation nulle) en conférant à Alice un pouvoir de calcul illimité. Cependant, on peut aussi se dire qu'Alice a en sa possession des informations supplémentaires (dont ne dispose pas Bob) qui lui permettent de répondre efficacement aux questions de Bob.

I.3.2. Connaissance et information

Il faut préciser que ce qui est appelé « connaissance » ici est tout à fait différent de « l'information » au sens de la théorie de l'information.

D'une part, la connaissance est définie par rapport au pouvoir calculatoire, ce qui n'est pas le cas de l'information. Par exemple, d'un point de vue de la théorie de l'information, il n'y a pas de différence entre répondre à la question « est-ce que le graphe est eulérien ? » et « est-ce que le graphe est hamiltonien ? », alors que nous avons vu qu'il y en a une pour ce qui est de la connaissance.

D'autre part, la connaissance s'appuie sur des objets entièrement publiquement connus, alors que dans la théorie de l'information les objets ne sont que partiellement publiquement connus.

II. Preuves interactives

Les protocoles de preuves à divulgation nulle font partir de la famille des protocoles de preuves interactives. Ces systèmes de preuves interactives sont intéressants en soi cependant nous nous contenterons d'une présentation rapide permettant d'introduire les protocoles zero-knowledge.

II.1. Machines de Turing interactives

Une interaction met en jeu au moins deux participants. Chaque participant est décrit par une machine de Turing interactive ou plus simplement ITM (Interactive machine Turing en anglais).

II.1.1. Une ITM seule

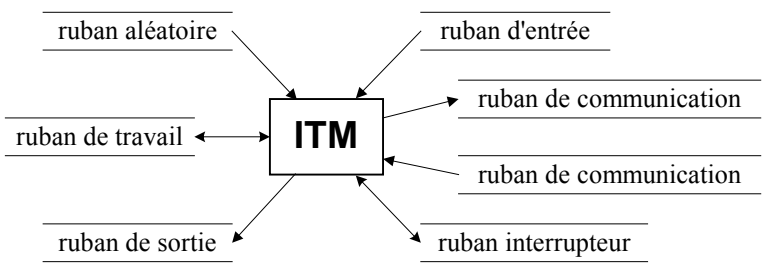


Figure 2 : Machine de Turing interactive

Définition II-1

Une machine de Turing interactive *(*ITM*) est une machine de Turing possédant sept rubans :*

- *un ruban d'entrée en lecture seule qui contient l'*entrée *;*
- *un ruban de sortie en écriture seule qui contient, lorsque la machine s'arrête, la* sortie *;*
- *un ruban aléatoire en lecture seule ;*
- *deux rubans de communication, le premier correspondant au flux envoyé à une autre ITM en écriture seule et le second correspondant au flux reçu d'une autre ITM en lecture seule ;*
- *un ruban de travail en lecture-écriture ;*
- *un ruban interrupteur en lecture-écriture.*

Chaque ITM est associée à un bit $\sigma \in \{0,1\}$ *appelé son* identité*. Elle est dite* active *si son identité est égale au contenu du ruban interrupteur, sinon elle est dite* endormie*. Lorsqu'elle est endormie, son état, la position de la tête de ses rubans et leur contenu n'est jamais modifié.*

On voit immédiatement que la définition ci-dessus ne permet pas à une ITM de fonctionner seule. En effet, si elle s'endort, elle ne peut plus redevenir active. De plus, les rubans de communication ne servent à priori à rien. En fait, tout ceci n'est pas gênant puisqu'une ITM n'est pas sensée travailler seule mais en couple avec une autre.

II.1.2. Couple d'ITM

Lorsque deux ITM travaillent conjointement, elles partagent certains de leurs rubans. Cela leur permet notamment de pouvoir travailler sur la même entrée et de communiquer ensemble.

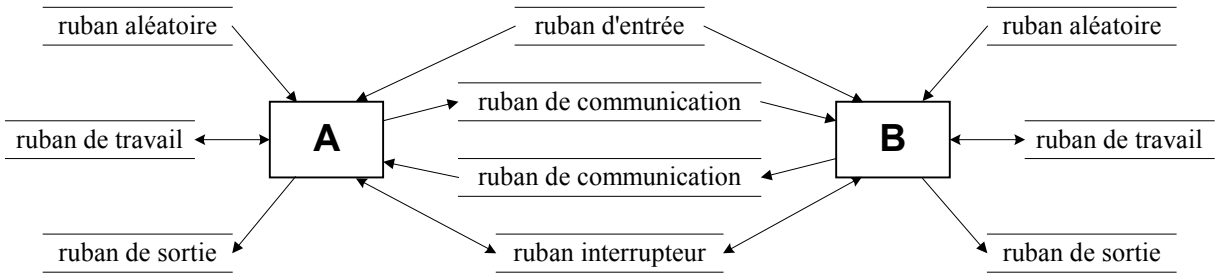


Figure 3 : Travail conjoint de deux ITM

Définition II-2

Deux ITM sont dites liées *si :*

- *elles ont des identités opposées ;*
- *leur ruban d'entrée coïncide ;*
- *leur ruban interrupteur coïncide ;*
- *le ruban de communication en lecture seule de l'une coïncide avec celui en écriture seule de l'autre et vice versa.*

Tous les autres rubans sont séparés.

La définition d'une ITM est donc plus claire maintenant. Voyons alors comment deux ITM liées travaillent ensemble.

Définition II-3

Le travail conjoint de deux ITM liées*, sur une entrée commune x, est une séquence de paires (A,B). Chacune de ces paires est constituée de la configuration locale des deux machines et chaque fois une seule machine est active (pas nécessairement la même) tandis que l'autre est endormie.*

Lorsqu'une ITM active s'arrête, on considère que les deux ITM s'arrêtent et la sortie correspond à la sortie de la machine qui s'est arrêtée.

II.1.3. ITM avec entrée auxiliaire

On peut remarquer dans les deux dernières définitions que les ITM liées n'ont pas d'entrée privée. Cette restriction nous permet d'aborder les systèmes de preuves interactives et les protocoles zero-knowledge plus simplement.

Cependant, cela n'est pas très réaliste (le prouveur peut en avoir besoin pour calculer efficacement sa preuve ou le vérifieur pour essayer de tricher…). Nous introduisons donc ici, par rapport au modèle précédent, une entrée auxiliaire.

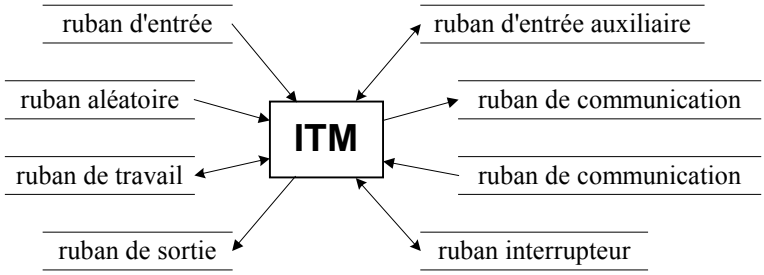


Figure 4 : ITM avec entrée auxiliaire

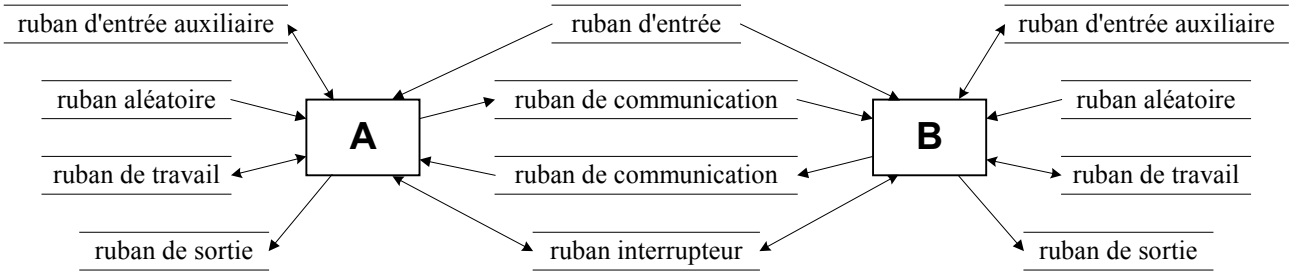


Figure 5 : Travail conjoint de deux ITM avec entrée auxiliaire

Définition II-1 doit être légère modifiée pour introduire un ruban d'entrée auxiliaire accessible en lecture-écriture, mais **Définition II-2** et **Définition II-3** restent inchangées.

Précisons seulement que :

- La complexité d'une ITM se calcule toujours par rapport à son entrée (et pas son entrée auxiliaire).
- La machine A n'a aucune idée de ce qui se trouve dans l'entrée auxiliaire de B et vice versa.

II.2. Systèmes de preuves interactives

Maintenant que nous avons présenté les machines de Turing interactives, nous pouvons aborder les systèmes de preuves interactives dans lesquels les deux antagonistes (le prouveur et le vérifieur) sont des ITM liées.

II.2.1. Définition

Nous avions dit au chapitre I.2.2 page 3 que, d'une manière générale, il est beaucoup plus difficile d'établir une preuve que de la vérifier. Les systèmes de preuves interactives n'échappent pas à cette règle. De fait, le pouvoir calculatoire des deux ITM liées P et V (pour prouveur et vérifieur) n'est pas du tout le même suivant le rôle qu'elles jouent. Ainsi, P dispose d'une puissance de calcul infini alors que V est limité à des algorithmes en temps polynomial par rapport à l'entrée commune.

Nous avions aussi vu, au chapitre I.2.3, que tout système de preuve doit respecter deux règles fondamentales : la validité et la complétude. Dans les systèmes de preuves interactives, des erreurs (à ces deux règles) peuvent cependant être acceptées tant qu'elles restent dans des proportions négligeables (il est toujours possible de diminuer la probabilité d'erreur en répétant la procédure de vérification autant de fois que nécessaire).

Rem : Dans la définition qui suit, le vérifieur retourne 1 s'il accepte la preuve et 0 dans le cas contraire.

Définition II-4 :
Soient (P,V) *une paire de machines de Turing interactives liées ;*
x l'entrée commune.
On note $\langle P,V\rangle(x)$ *la sortie de* (P,V) *sur x (après interaction).*
(P,V) *est un* système de preuves interactives *pour le langage L si V s'exécute en temps polynomial sur x et si les deux conditions sont respectées :*

- *La complétude :* $\forall x \in L, \Pr(\langle P,V\rangle(x)=1) \geq \frac{2}{3}$
- *La validité :* $\forall x \notin L, \Pr(\langle B,V\rangle(x)=1) \leq \frac{1}{3}$ *pour toute ITM B.*

Il faut souligner que la condition de validité doit être bien sûr vérifiée pour P (c'est à dire une machine qui respecte le protocole) mais aussi pour n'importe quelles autres machines B (en particulier, celles qui essaient de tromper le vérifieur et ne suivent pas le protocole).

II.2.2. La classe IP

Définition II-5
La classe $\mathcal{IP}$ est constituée de tous les langages ayant des systèmes de preuves interactives.

II.2.2.1. Comparaison avec NP

La classe $\mathcal{IP}$ est une très grande classe de langages. Elle englobe notamment la classe $\mathcal{NP}$ ($\mathcal{NP} \subseteq \mathcal{IP}$). En effet, tous les langages de NP ont un système de preuves interactives.

Prenons par exemple un langage L de $\mathcal{NP}$. Mettons que R_L soit une relation particulière à ce langage reconnaissable en temps polynomial. Alors un système de preuves pour L est (sur une entrée $x \in L$) :

- P calcule y tel que $|y| \in \text{poly}(|x|) \wedge (x,y) \in R_L$ et l'envoie à V
- V retourne 1 si $|y| \in \text{poly}(|x|)$ et $(x,y) \in R_L$ ou 0 sinon

On voit immédiatement que le protocole ci-dessus est bien un le système de preuves tel que nous l'avons défini plus haut (vérifiant les propriétés de complétude et de validité).

On remarque aussi qu'il s'agit d'un protocole très particulier qui ne fait intervenir que des machines déterministes (les rubans aléatoires ne sont jamais utilisés). En fait, on peut voir NP comme une classe des systèmes de preuves interactives dans laquelle les interactions sont unidirectionnelles (du prouveur vers le vérifieur) et où le vérifieur est déterministe (et ne se trompe jamais).

II.2.2.2. Comparaison avec BPP

La classe des langages probabilistes polynomiaux $\mathcal{BPP}$ est aussi comprise dans $\mathcal{IP}$. En effet, pour tout langage $L \in \mathcal{BPP}$, il est possible de mettre en place un système de preuves interactives (sans interaction) où le vérifieur déciderait de l'appartenance ou non de x (l'entrée commune) à L tout seul en temps polynomial.

Donc $\mathcal{BPP} \cup \mathcal{NP} \subseteq \mathcal{IP}$.

II.3. Exemple : graphes non isomorphes

Maintenant que les systèmes de preuves interactives ont été définis rigoureusement, voyons un exemple concret un peu plus intéressant que les deux exemples génériques vus en II.2.2.1 et II.2.2.2. Considérons pour cela le langage des paires de graphes non isomorphes que nous appellerons *GNI*.

Ce langage n'est pas connu comme appartenant à $\mathcal{BPP} \cup \mathcal{NP}$ donc il n'a pas système de preuves interactives évident. De plus, puisque $GNI \in \mathcal{IP}$ (nous allons le montrer juste après) cela laisse à penser que $\mathcal{IP}$ est strictement supérieure à $\mathcal{BPP} \cup \mathcal{N}$ (en fait $\mathcal{IP}$ serait égale à $\mathcal{PSPACE}$).

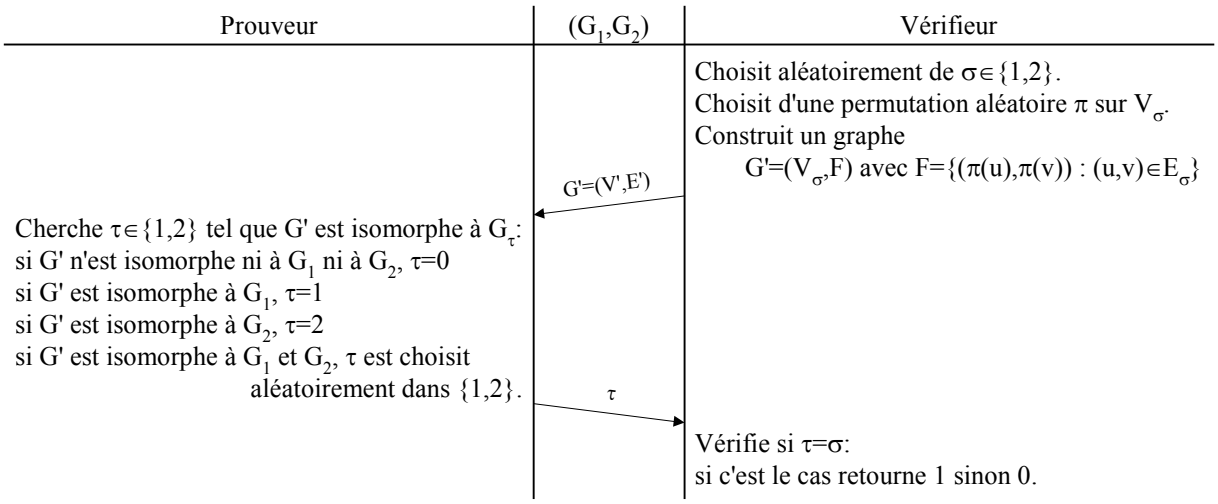
II.3.1. Protocole

L'entrée commune aux deux ITM P et V est une paire de graphes, $G_1=(V_1,E_1)$ et $G_2=(V_2,E_2)$.

Les étapes :

- Le vérifieur choisit aléatoirement un des deux graphes, c'est à dire qu'il choisit aléatoirement $\sigma \in \{1,2\}$.
  Ensuite, il construit un nouveau graphe *G'* isomorphe à G_σ, c'est à dire qu'il sélectionne une permutation aléatoire π sur V_σ et $G'=(V_\sigma,\{(\pi(u),\pi(v)) : (u,v)\in E_\sigma\})$.
  Enfin, il envoie *G'* au prouveur.
- Après réception de *G'*, le prouveur trouve $\tau \in \{1,2\}$ tel que *G'* est isomorphe à G_τ. Si aucun τ ne satisfait la condition, τ est mis à 0. Si plusieurs τ satisfont la condition, le prouveur en choisit un au hasard.
  Le prouveur renvoie τ à V.
- Si $\tau=\sigma$, alors le vérifieur retourne 1. Sinon, il retourne 0.

Soit sous forme schématique :



Le vérifieur peut aisément réaliser son travail en temps polynomial (sur l'entrée (G_1,G_2)). Au contraire il n'existe pas (jusqu'à aujourd'hui) d'algorithme polynomial pour le prouveur, mais celui-ci n'est pas limité dans sa puissance de calcul, donc cela ne pose pas de problème.

II.3.2. Démonstration

Intuitivement, on voit que les contraintes de complétude et de validité sont vérifiées :

- Complétude : Si les graphes G_1 et G_2 ne sont pas isomorphes, alors le prouveur trouvera un et un seul $\tau \in \{1,2\}$ tel que *G'* est isomorphe à G_τ et de plus $\tau=\sigma$ donc le vérifieur retournera 1 à coup sûr.
- Validité : Si les graphes sont isomorphes, *G'* est isomorphe à G_1 et G_2 donc le prouveur a 50% de chance de trouver le bon τ (tel que $\tau=\sigma$). On remarque que $\Pr(\langle P,V\rangle(x)=1) \le \frac{1}{2}$ alors que nous voulions $\Pr(\langle P,V\rangle(x)=1) \le \frac{1}{3}$. Pour obtenir la limite désirée, il suffit de réitérer le protocole.

Essayons de démontrer plus formellement que $GNI \in IP$. La propriété de complétude est évidente. Celle de validité l'est moins, focalisons nous donc dessus.

Pour démontrer que $\Pr(\langle P,V\rangle(x)=1) \le \frac{1}{2}$ si G_1 et G_2 sont isomorphes, nous allons montrer que *G'* ne donne aucune information sur σ.

On note ξ une variable uniformément distribuée sur {1,2} et Π une variable uniformément distribuée sur l'ensemble des permutations des sommets d'un graphe. Ces deux variables sont indépendantes. Il faut montrer que ξ et $\Pi(G_\xi)$ sont aussi statistiquement indépendants (donc *G'* ne donne aucune information sur σ).

Prouvons que (1) $\Pr(\xi=1|\Pi(G_\xi)=G') = \Pr(\xi=2|\Pi(G_\xi)=G') = \frac{1}{2}$. Remarquons d'abord que $S_1=\{\pi : \pi(G_1)=G'\}$ et $S_2=\{\pi : \pi(G_2)=G'\}$ sont de même cardinalité puisque sont G_1 et G_2 isomorphes. A partir de là, on a :

$$\begin{aligned}\Pr(\Pi(G_\xi)= G'|\xi=1) &= \Pr(\Pi(G_1)= G') \\ &= \Pr(\Pi\in S_1) \\ &= \Pr(\Pi\in S_2) \\ &= \Pr(\Pi(G_\xi)= G'|\xi=2)\end{aligned}$$

On retrouve donc (1) grâce à la règle de Bayes.

On utilise ensuite (1) pour montrer que pour tout algorithme R $\Pr(R(\Pi(G_\xi))=\xi) \le \frac{1}{2}$ (avec égalité si R retourne toujours un élément de $\{1,2\}$).

II.4. Augmentation du modèle

Dans une première approche des systèmes de preuves interactives, nous nous sommes contentés d'utiliser des ITM sans entrée auxiliaire pour des raisons de simplicité. Cependant, il nous faut étendre **Définition II-4** afin qu'elle prenne en compte les entrées auxiliaires car nous en auront besoin au chapitre III.3 page 14.

Définition II-6
Soient (P,V) *une paire de machines de Turing interactives avec entrée auxiliaire liées ;*
x l'entrée commune ;
y l'entrée auxiliaire de P ;
z celle de V.
On note $\langle P(y),V(z)\rangle(x)$ *la sortie de* (P,V) *sur x (après interaction).*
(P,V) *est un* système de preuves interactives *pour le langage L si V s'exécute en temps polynomial sur x et si les deux conditions sont respectées :*

- *La complétude :* $\forall x\in L, \exists y\in\{0,1\}^*, \Pr(\langle P(y),V(z)\rangle(x)=1) \ge \frac{2}{3}$ *pour tout* $z\in\{0,1\}^*$
- *La validité :* $\forall x\notin L, \Pr(\langle B(y),V(z)\rangle(x)=1) \le \frac{1}{3}$ *pour toute ITM B, tout* $y\in\{0,1\}^*$ et *tout* $z\in\{0,1\}^*$.

III. Preuves à divulgation nulle

Maintenant que nous savons ce que sont les systèmes de preuves interactives, nous allons voir une sous-classe de $\mathcal{IP}$ qui possède une propriété très intéressante en cryptographie, celle de ne rien divulguer du secret du prouveur. Ces systèmes de preuves interactives sont dits à divulgation nulle. On parle aussi de protocoles zero-knowledge.

III.1. Preuves à divulgation nulle parfaites et calculatoires

La définition des protocoles zero-knowledge s'appuie sur les remarques que nous avons faites sur la notion de « connaissance » au chapitre I.3 page 3. En effet, un protocole est dit zero-knowledge si tout ce qui peut être calculé efficacement après interaction avec le prouveur pouvait être aussi calculé efficacement avant grâce à l'entrée commune x.

III.1.1. Simulateurs

III.1.1.1. Motivations

Pour prouver qu'aucune machine V^* (c'est à dire un vérifieur qui essaierait de tricher en ne suivant pas le protocole) ne peut gagner de connaissance, on utilise une machine M^* appelée simulateur.

Ce simulateur est régit par deux contraintes :

- Il ne connaît pas le code du prouveur P.
- Il doit s'exécuter en temps polynomial (ou en temps attendu polynomial, suivant les définitions).

A partir de là, on dit que si M^* est capable de simuler l'interaction entre P et V^*, alors V^* n'a gagné aucune connaissance de P puisque la même sortie a pu être générée sans connaître le code de P. Donc s'il existe un simulateur pour toutes les machines V^* possibles, alors le protocole est zero-knowledge.

III.1.1.2. Fonctionnement

Il n'est pas possible de donner le fonctionnement exact de tous les simulateurs puisque celui-ci dépend d'une part du protocole zero-knowledge dont on veut faire la preuve et d'autre part de la machine V^*. On peut quand même dire que le simulateur essaie toujours de retourner une sortie identique à celle de V^* dans une interaction (réelle) avec P. Cependant, cela n'est pas toujours possible et parfois il retourne un symbole spécial $\perp$ pour indiquer qu'il a échoué.

Reprenons l'exemple de la caverne d'Ali Baba (page 2). Pour simuler le protocole, on fait appel à deux personnes A et B ne connaissant pas la formule magique :

- A va se placer au fond de la caverne aléatoirement soit à gauche, soit à droite.
- B rentre dans la caverne et demande à A de sortir par la droit ou par la gauche (aléatoirement).
- Si A est entré du même coté que celui que B a demandé, alors le simulateur retourne 1 (i.e., B est convaincu que A connaît la formule magique). Sinon, le simulateur retourne $\perp$ (échec).

Il est indispensable de permettre au simulateur d'échouer. Cependant, nous verrons qu'il faut limiter la probabilité de retourner $\perp$.

III.1.1.3. Types de protocoles zero-knowledge

Il existe plusieurs degrés d'exigence dans la non-divulgation de connaissance :

- Parfaite : Il existe un simulateur dont les sorties sont les mêmes que les interactions réelles. C'est le cas des graphes isomorphes (exemple du chapitre III.2 page 12).
- Calculatoire : Il existe un simulateur dont les sorties ne peuvent être distinguées (de façon calculatoire) des interactions réelles.
- Probabiliste : Il existe un simulateur dont les sorties sont les mêmes que les interactions réelles exceptées un nombre constant d'erreurs.
- Inutile : Il n'existe peut-être pas de simulateur mais on peut montrer que le vérifieur ne peut gagner qu'une quantité négligeable de connaissance. C'est le cas des preuves en parallèle.

Nous ne nous intéresserons qu'aux deux premiers types de preuves à divulgation nulle.

III.1.2. Preuve à divulgation nulle parfaite

Définition III-1
Soient L *un langage* $\in IP$
(P,V) *un système de preuves interactives pour* L
On dit que (P,V) *est à* divulgation parfaitement nulle *si, pour toute ITM* V^**, il existe une machine* M^* *probabiliste en temps polynomiale telle que pour tout entrée commune* $x \in L$ *:*

- *M* retourne le symbole spécial* $\perp$ *(échec) avec une probabilité* $\leq \frac{1}{2}$ $(\Pr(M^*(x)=\perp) \leq \frac{1}{2})$
- *Soit* $m^*(x)$ *une variable aléatoire décrivant la distribution de* $M^*(x)$ *telle que* $M^*(x)\neq\perp$, *alors les variables* $\langle P,V^*\rangle(x)$ *et* $m^*(x)$ *sont identiquement distribuées.*

La machine M est appelée* simulateur parfait.

La première condition est très importante. En effet, si on permettait à *M** de retourner $\perp$ autant de fois qu'il le souhaite, alors d'une certaine manière cela augmenterait son pouvoir calculatoire puisqu'il pourrait retourner $\perp$ une infinité de fois jusqu'à ce qu'il trouve la bonne réponse. Mais si le simulateur est plus puissant que *V** alors on ne plus en tirer aucune conclusion (car il pourrait exister des simulateurs pour des protocoles non zero-knowledge).

Il existe une autre définition des systèmes de preuves interactives à divulgation parfaitement nulle qui n'autorise pas le simulateur à retourner $\perp$. En contrepartie, celui-ci peut s'exécuter en temps attendu polynomial (i.e., un temps qui est en moyenne polynomial) et non plus en temps strictement polynomial. Cette définition peut sembler plus claire que celle que nous avons donnée. Cependant, elle est plus compliquée à utiliser à cause de la notion de temps polynomial en moyenne très subtile. De plus, on peut remarquer que la première définition implique la deuxième. Donc nous utiliserons **Définition III-1**.

III.1.3. Preuve à divulgation nulle calculatoire

En pratique, il n'est pas nécessaire d'avoir des systèmes de preuves interactives à divulgation parfaitement nulle. En effet, il suffit que les sorties de *M** soient calculatoirement indistingables de celles engendrées par l'interaction de *V** avec *P*. Ce relâchement reste compatible avec ce que nous avions demandé au début : « tout ce qui peut être calculé efficacement après interaction avec le prouveur pouvait être aussi calculé efficacement avant (sans interaction) grâce à l'entrée commune *x* ».

Avant de donner la définition des preuves à divulgation nulle calculatoire, rappelons ce que sont deux ensembles calculatoirement indistingables.

Définition III-2

Soit un langage L

On dit que les ensembles $\{Rx\}_{x\in L}$ *et* $\{Sx\}_{x\in L}$ *sont* calculatoirement indistingables *si :*

- *pour tout algorithme probabiliste en temps polynomial D,*
- *pour tout polynôme p,*
- *pour tout* $x\in L$ *suffisamment long,*

on a $|\Pr(D(x,R_x)=1) - \Pr(D(x,S_x)=1)| < \frac{1}{p(|x|)}$

D'où :

Définition III-3

Soient L un langage $\in$ *IP*

(P,V) *un système de preuves interactives pour L*

On dit que (P,V) *est à* divulgation calculatoirement nulle *si, pour toute ITM V*, il existe une machine M* probabiliste en temps polynomiale telle que les ensembles* $\{\langle P,V^*\rangle(x)\}_{x\in L}$ et $\{M^*(x)\}_{x\in L}$ *sont calculatoirement indistingables.*

La machine M est appelée* simulateur.

Il est possible de modifier légèrement cette définition pour se faciliter le travail dans les démonstrations. Plutôt que de ne considérer que la sortie du vérifieur après interaction, il est possible de mémoriser la séquence entière de toutes les configurations locales du vérifieur pendant l'interaction. En fait, il suffit de s'intéresser au contenu du ruban aléatoire du vérifieur et à la séquence de messages qu'il reçoit du prouveur (le reste de sa configuration se déduit de ça).

Définition III-4

Soient L un langage $\in$ *IP*

(P,V) *un système de preuves interactives pour L*

On note $\text{view}^P_{V^*}(x)$ *une variable aléatoire décrivant le contenu du ruban aléatoire de V* et les messages que V* reçoit de P pendant le calcul conjoint de P et V* sur l'entrée x.*

On dit que (P,V) *est* à divulgation calculatoirement nulle *ou simplement* à divulgation nulle *si, pour toute ITM V*, il existe une machine M* probabiliste en temps polynomiale telle que les ensembles* $\{\text{view}^P_{V^*}(x)\}_{x\in L}$ et $\{M^*(x)\}_{x\in L}$ *sont calculatoirement indistingables.*

Les deux définitions ci-dessus sont équivalentes.

III.1.4. Complexité

Définition III-5
La classe $\mathcal{PZK}$ est constituée des langages ayant des systèmes de preuves interactives à divulgation parfaitement nulle.
La classe $\mathcal{CZK}$ est constituée des langages ayant des systèmes de preuves interactives à divulgation calculatoirement nulle.
Cette dernière se note aussi plus simplement $\mathcal{ZK}$.

Il est possible de construire un protocole zero-knowledge parfait avec tous les langages L de $\mathcal{BPP}$ où le prouveur ne fait rien et le vérifieur vérifie directement si l'entrée commune $x \in L$ ou non. Pour le démontrer, on construit un simulateur M^* identique à V^* excepté que le ruban de communication de V^* (inutilisé de toute façon) est considéré dans M^* comme un simple ruban de travail. Donc $\mathcal{BPP} \subseteq \mathcal{PZK}$.

De plus, il est évident que $\mathcal{PZK} \subseteq \mathcal{CZK}$.

Donc $\mathcal{BPP} \subseteq \mathcal{PZK} \subseteq \mathcal{CZK} \subseteq \mathcal{IP}$. Il est probable que les deux premières inclusions soient strictes et si les fonctions à sens unique existent alors la dernière est une égalité.

Enfin, il semble aussi que $\mathcal{NP} \subseteq \mathcal{CZK}$ (si les schémas de mise en gage (commitment schemes) avec contrainte de secret et de non-ambiguïté existent).

III.2. Exemples : Graphes isomorphes

Considérons un exemple de protocoles zero-knowledge plus formel que celui de la caverne d'Ali Baba.

Le langage des graphes isomorphes que nous appèlerons *GI* permet en effet de construire un protocole théorique (impossible à utiliser en pratique).

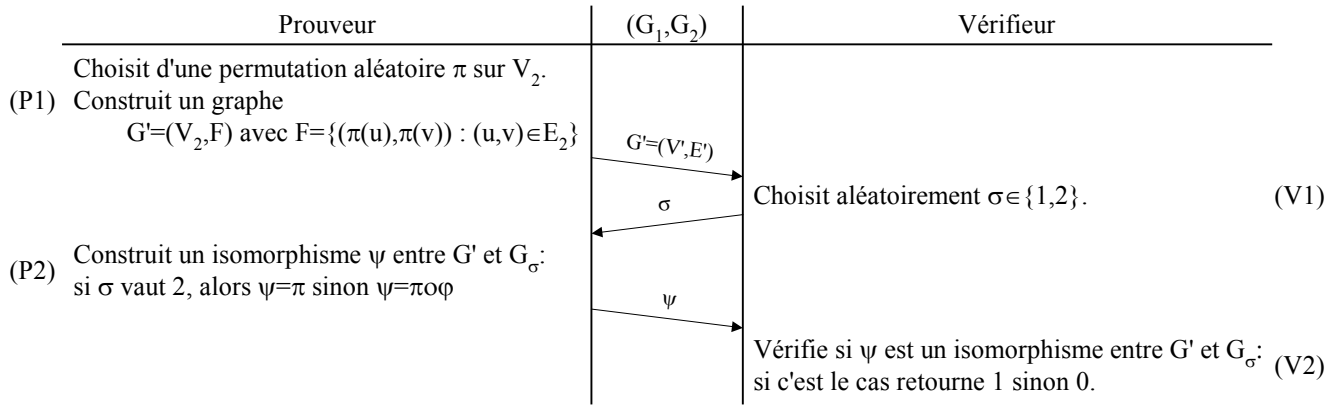
III.2.1. Protocole

L'entrée commune à P et V est une paire (G_1,G_2) de graphes isomorphes. On note $G_1=(V_1,E_1)$, $G_2=(V_2,E_2)$ et φ l'isomorphisme qui transforme G_1 en G_2.

Les étapes :

- Le prouveur sélectionne une permutation aléatoire π sur V_2 et construit $G'=(V_2,\{(\pi(u),\pi(v)) : (u,v)\in E_2\})$.
  Il envoie G' au vérifieur.
- Le vérifieur reçoit $G'=(V',E')$.
  Il choisit aléatoirement $\sigma \in \{1,2\}$ et demande à voir un isomorphisme entre G_σ et G' en envoyant σ au prouveur.
- Le prouveur reçoit σ.
  Si σ vaut 2, alors il retourne π sinon il retourne $\pi \circ \varphi$ au vérifieur.
- Le vérifieur reçoit une permutation ψ.
  Si c'est un isomorphisme entre G_σ et G', alors il écrit sur sa sortie 1, sinon 0.

Soit sous forme schématique :



On note P_{GI} le programme du prouveur. Il peut être implémenté en temps probabiliste polynomial si on lui autorise une entrée auxiliaire contenant φ (cependant on ne considère pas l'entrée auxiliaire dans la démonstration qui suit).

Le vérifieur peut être facilement implémenté en temps probabiliste polynomial.

III.2.2. Démonstration

Pour montrer que le protocole ci-dessus est zero-knowledge, il faut montrer :

- Qu'il s'agit d'une preuve interactive :
  1. Si G_1 et G_2 sont isomorphes, alors le vérifieur accepte toujours.
  2. Si G_1 et G_2 ne sont pas isomorphes, alors le vérifieur rejette l'entrée avec une probabilité d'au moins ½.
- Qu'il s'agit d'une preuve à divulgation parfaitement nulle : pour toutes les ITM en temps probabiliste polynomial V^*, il existe une machine M^* en temps probabiliste polynomial telle que :
  3. M^* retourne $\perp$ avec une probabilité d'au plus ½.
  4. Les ensembles $\text{view}_{V^*}^{P_{GI}}(x)$ et $m^*(x)$ (tels qu'ils ont été définis auparavant) sont identiquement distribués.

La condition (1) est évidente si chaque partie suit le protocole puisque si G_1 et G_2 sont isomorphes alors ils sont tous les deux isomorphes à G'.

La condition (2) est aussi assez simple. En effet, si G_1 et G_2 ne sont pas isomorphes, alors G' ne peut pas être isomorphe aux deux graphes. Donc quelle que soit la façon d'opérer du prouveur (éventuellement en trichant), il y a un $\sigma \in \{1,2\}$ tel que G_σ et G' ne sont pas isomorphes. Alors, si le vérifieur suit le protocole, il rejette l'entrée avec une probabilité d'au moins ½.

Pour démontrer les conditions (3) et (4), il faut construire une machine M^* dans le cas où V suit le protocole (très simple) et aussi dans le cas général (V^* a un comportement arbitraire en temps probabiliste polynomial). Notre simulateur doit donc incorporer le code de V^* et il agit comme ce qui suit sur l'entrée x :

- Remplissage du ruban aléatoire du vérifieur avec une chaîne aléatoire $r \in \{1,2\}^{p(|x|)}$ (p est un polynôme qui limite le temps d'exécution du simulateur).
- Simulation de (P1) : Le simulateur choisit aléatoirement τ sur $\{1,2\}$ et une permutation ψ sur V_τ. Puis il construit $G''=(V_\tau, \{(\psi(u),\psi(v)) : (u,v)\in E_\tau\})$.
- Simulation de (V1) : Le simulateur initialise V^* en plaçant r sur le ruban aléatoire de V^*, x sur son ruban d'entrée et G'' sur son ruban de communication entrante. Il exécute V^* dans ces conditions et note sa sortie σ (on normalise σ de sorte que $\sigma=1$ si $\sigma \neq 2$ et $\sigma=2$ sinon).
- Simulation de (P2) et (V2) : Si $\tau=\sigma$, alors le simulateur s'arrête avec en sortie (x,r,G'',ψ). Sinon, il retourne $\perp$.

On remarque déjà que si V^* est en temps polynomial, alors M^* l'est aussi.

Pour prouver la condition (3) (le simulateur retourne $\perp$ avec une probabilité d'au plus ½), on note ξ une variable uniformément distribuée sur {1,2} et Π une variable uniformément distribuée sur l'ensemble des permutations des sommets d'un graphe et on montre que $\Pr(\xi=1|\Pi(G_\xi)=G'') = \Pr(\xi=2|\Pi(G_\xi)=G'')$ (la démonstration a déjà été faite au chapitre II.3.2 page 8). Donc quel que soit l'algorithme utilisé par V^* qui prend en entrée G'' et qui sort une valeur dans {1,2}, ce programme retournera ξ sur l'entrée $\Pi(G_\xi)$ avec une probabilité de ½. Donc $\tau \neq \sigma$ avec une probabilité de ½. Donc M^* retourne $\perp$ avec une probabilité ½.

Il ne nous reste plus qu'à démontrer la condition (4) (i.e., $\Pr[\text{view}_{V^*}^{P_{GI}}(x) = (x,r,H,\psi)] = \Pr[M^*(x) = (x,r,H,\psi) \mid M^*(x)\neq\perp]$ pour toute chaîne r, graphe H et permutation ψ). Nous ne finirons pas la preuve, cependant on peut remarquer que $\text{view}_{V^*}^{P_{GI}}(x)$ et $m^*(x)$ sont distribués sur des quadruplets de la forme $(x,r,\_,\_)$ et que r est uniformément distribué sur $\{1,2\}^{p(|x|)}$. On note $\lambda_x(r)$ la variable aléatoire distribuée sur les deux derniers éléments de $\text{view}_{V^*}^{P_{GI}}(x)$ et $\mu_x(r)$ celle sur les deux derniers éléments de $m^*(x)$. Il ne reste donc plus qu'à montrer que $\lambda_x(r)$ et $\mu_x(r)$ sont identiquement distribués pour chaque x et r ce que nous ne ferons pas ici.

III.3. Preuves à divulgation nulle avec entrée auxiliaire

Dans le chapitre III.1 page 9, nous avons formalisé ce que nous avions compris intuitivement avec l'exemple de la caverne d'Ali Baba en donnant aux systèmes de preuves interactives à divulgation nulle une définition rigoureuse. Cependant, nous avons pu nous rendre compte que cette définition est très restrictive dans le sens où elle n'utilise par hypothèse que des ITM sans entrée auxiliaire. Cette hypothèse rend impossible d'utiliser ces protocoles en pratique où les machines disposent presque toujours d'entrée auxiliaire. Il est donc nécessaire d'étendre les définitions que nous avons données afin qu'elles utilisent des ITM avec entrée auxiliaire.

Définition III-6
Soient L *un langage* $\in I\mathcal{P}$
(P,V) *un système de preuves interactives pour* L
On note $P_L(x)$ *l'ensemble des chaînes* y *telles que la condition de complétude est respectée avec* $x\in L$ *(i.e., pour toute chaîne* $z\in\{0,1\}^*$, $\Pr(\langle P(y),V(z)\rangle(x)=1) \geq {}^2/_3$*).*
On dit que (P,V) *est à* divulgation calculatoirement nulle avec entrée auxiliaire *si,*
pour toute ITM probabiliste en temps polynomial avec entrée auxiliaire V^**,*
il existe une machine M^* *probabiliste en temps polynomiale par rapport à l'entrée commune telle que*
les ensembles $\{\langle P(y),V^*(z)\rangle(x)\}_{x\in L,y\in P_L(x),z\in\{0,1\}^*}$ et $\{M^*(x)\}_{x\in L,z\in\{0,1\}^*}$ sont calculatoirement indistingables.

Dans la définition précédente, y représente une information permettant à priori au prouveur de calculer efficacement sa preuve ce qui n'est pas utile en théorie (puisqu'il dispose d'un pouvoir de calcul infini) mais qui l'est en pratique. Si le prouveur se sert de son entrée auxiliaire, il faut faire attention que cela ne le perturbe pas (i.e., qu'il puisse remplir les conditions de complétude et de validité des preuves interactives aussi bien que sans son entrée auxiliaire) c'est pour cette raison que y doit appartenir à l'ensemble $P_L(x)$. Il n'y a pas de restriction de ce type sur z le contenu du ruban d'entrée auxiliaire du vérifieur. Celui-ci peut contenir n'importe quel type d'informations, notamment des informations en relation avec x afin d'essayer de tricher.

Donc maintenant, intuitivement, on dit d'un protocole qu'il est zero-knowledge si tout ce qui peut être calculé efficacement avec x et z après interaction avec le prouveur sur l'entrée commune x, peut aussi être calculé efficacement avec x et z sans interaction.

Ce petit changement paraît bénin, cependant il a une grande importance. En effet, de cette dernière définition, il découle un lemme crucial dans l'implémentation pratique des protocoles zero-knowledge.

Lemme (lemme de composition séquentielle)
Soient L *un langage* $\in \mathcal{ZK}$
P *une ITM avec entrée auxiliaire (un prouveur en fait) qui est zero-knowledge avec entrée auxiliaire sur* L*.*
Q *un polynôme.*

> *P_Q une ITM avec entrée auxiliaire qui procède sur l'entrée x en $Q(|x|)$ étapes, chacune de ces étapes consistant à lancer P sur x. (On souligne que si P est probabiliste, alors P_Q doit utiliser des séquences aléatoires indépendantes à chaque étape).*
> *Alors P_Q est un système de preuves interactives à divulgation nulle avec entrée auxiliaire sur L.*
> *De plus, si P est à divulgation parfaitement nulle, alors P_Q l'est aussi.*

Bien que le lemme paraisse assez évident, sa démonstration est plutôt longue. Nous ne la donnerons donc pas ici.

Ce lemme est très important car généralement après une itération d'un protocole zero-knowledge, on a une petite conviction de la vérité (le prouveur ment ou il connaît réellement son secret) mais cette conviction est trop faible pour en tirer des conclusions. Il faut donc réitérer le protocole pour renforcer cette conviction. Le problème dans ce cas, c'est qu'en réitérant le protocole, on n'est, à priori, plus assuré de ne pas divulguer d'information. Ce lemme, nous assure en fait que ce problème ne se pose pas dans le cas des systèmes de preuves interactives à divulgation nulle.

On peut remarquer ici un des aspects pratiques intéressants de ces protocoles. En effet, ces protocoles sont toujours de type « à clef publique » (voir exemple chapitre III.4.2 page 18). Mais contrairement aux autres protocoles à clef publique qui demandent un gros calcul, les systèmes de preuves interactives à divulgation nulle, eux, se décomposent en plusieurs calculs moyens et peuvent donc être implémentés sur des machines ayant des ressources limitées.

III.4. Exemples

III.4.1. Graphes trois-colorables

Le langage des graphes trois-colorables, appelé *G3C*, est constitué de tous les graphes simples (i.e., sans arête parallèle ou allant d'un sommet au même sommet) dont les sommets peuvent être colorés, avec trois couleurs, de telle sorte que deux sommets adjacents n'ont pas la même couleur. Ce langage permet de constituer un protocole zero-knowledge avec entrée auxiliaire.

III.4.1.1. Protocole

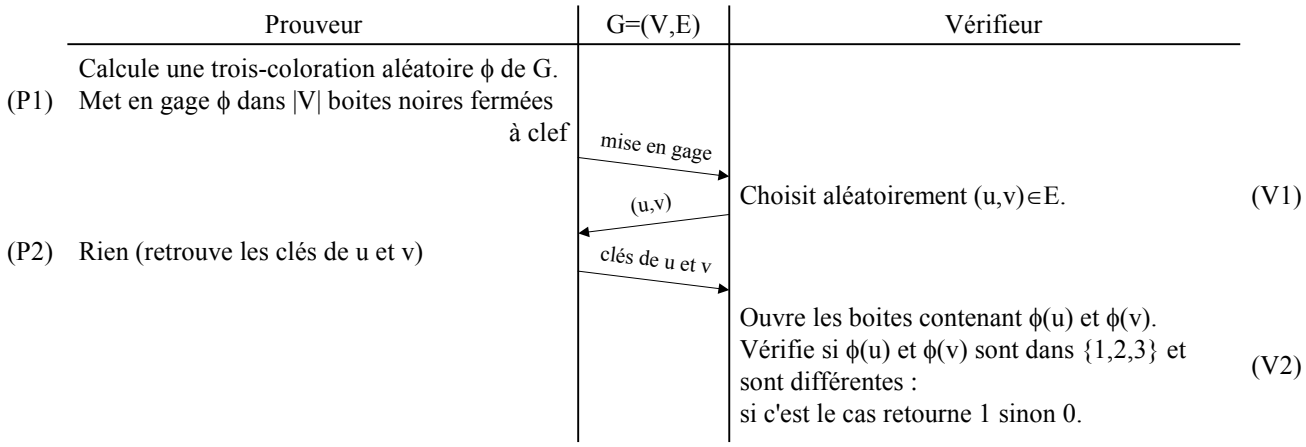
L'entrée commune est constituée d'un graphe $G=(V,E)$. Si $G \in G3C$, alors on note ψ une trois-coloration de G (i.e., $\psi : V \to \{1,2,3\}$ telle que $\psi(u) \neq \psi(v)$ pour tout $(u,v) \in E$).

L'idée du protocole est de décomposer la preuve en plusieurs (un nombre polynomial) morceaux de telle manière que chaque morceau ne divulgue aucune information, mais que l'ensemble garantit la validité de la déclaration du prouveur (i.e., qu'il connaît une trois-coloration pour ce graphe).

Les étapes :

- Le prouveur sélectionne une permutation aléatoire π sur $\{1,2,3\}$ et calcule une nouvelle trois-coloration ϕ de G ($\phi(v)=\pi(\psi(v)) \ \forall v \in V$). Ensuite il envoie au vérifieur $|V|$ boites noires fermées à clef (des mises en gages) telles que la $v^{\text{ème}}$ contienne la valeur $\phi(v)$.
- Le vérifieur choisit aléatoirement une arête $(u,v) \in E$ et l'envoie au prouveur.
- Le prouveur envoie au vérifieur les clefs pour les boites u et v.
- Le vérifieur ouvre les dites boites et vérifie si elles contiennent deux couleurs dans $\{1,2,3\}$ différentes.
  Il accepte (retourne 1) si c'est le cas, sinon il refuse l'entrée.

Soit sous forme schématique :



On note P_{G3C} le programme du prouveur. Il peut être implémenté en temps probabiliste polynomial grâce à son entrée auxiliaire contenant ψ.

Le vérifieur peut être facilement implémenté en temps probabiliste polynomial.

III.4.1.2. Démonstration

Nous avons déjà fait une démonstration propre pour P_{GI} (chapitre III.2.2 page 13). Nous nous contenterons ici de ne donner que quelques éléments de preuve.

Rappelons déjà ce qu'il faut démontrer :

- Il s'agit d'une preuve interactive :
  1. Si G est trois-colorable, alors le vérifieur accepte toujours.
  2. Si G n'est pas trois-colorable, alors le vérifieur rejette l'entrée avec une probabilité d'au moins $\frac{1}{|E|}$.
- Il s'agit d'une preuve à divulgation calculatoirement nulle avec entrée auxiliaire : pour toutes les ITM en temps probabiliste polynomial V^*, il existe une machine M^* en temps probabiliste polynomial telle que :
  3. M^* retourne $\perp$ avec une probabilité d'au plus ½.
  4. Les ensembles $\text{view}_{V^*}^{P_{G3C}}(x,y,z)$ et $\{M^*(x,z) \mid M^*(x,z) \neq \perp\}$ sont calculatoirement indistingables.

La condition (1) est évidente si chaque partie suit le protocole puisque si G est trois-colorable alors toutes les boites u et v telles que (u,v)∈E mises en gage à l'étape (P1) contiennent des valeurs $\phi(u) \neq \phi(v)$.

La condition (2) est aussi assez simple. En effet, si G n'est pas trois-colorable, alors il existe au moins une arête (u,v) mal coloriée. Donc quelle que soit la façon d'opérer du prouveur (éventuellement en trichant), il y a au moins deux boites mises en gage s et t contenant la même couleur et telles que $(s,t) \in E$. Alors, si le vérifieur suit le protocole, il rejette l'entrée avec une probabilité d'au moins $\frac{1}{|E|}$.

Pour démontrer les conditions (3) et (4), il faut construire une machine M^* dans le cas où V suit le protocole (très simple) et aussi dans le cas général (V^* a un comportement arbitraire en temps probabiliste polynomial). Notre simulateur doit donc incorporer le code de V^* et il agit comme ce qui suit sur l'entrée x avec y en entrée auxiliaire pour le prouveur et z dans celle du vérifieur :

- Remplissage du ruban aléatoire du vérifieur avec une chaîne aléatoire $r \in \{1,2\}^{p(|x|)}$ (p est un polynôme qui limite le temps d'exécution du simulateur).
- Simulation de (P1) : Le simulateur choisit aléatoirement et indépendamment $|V|$ valeurs $e_1,\ldots,e_{|V|} \in \{1,2,3\}$ en guise de trois-coloration. Il met en gage ces valeurs.

- Simulation de (V1) : Le simulateur initialise V^* en plaçant r sur le ruban aléatoire de V^*, x sur son ruban d'entrée, z sur son ruban d'entrée auxiliaire et la mise en gage sur son ruban de communication entrante. Il exécute V^* dans ces conditions et note sa sortie ω (on suppose sans perte de généralité que $\omega \in E$).
- Simulation de (P2) et (V2) : On note $\omega=(u,v)$. Si $e_u \neq e_v$, alors le simulateur s'arrête avec en sortie $(x,z,r,$ *« mise en gage »*$,(u,e_u,v,e_v))$. Sinon, il retourne $\perp$.

Clairement, le simulateur s'exécute en temps polynomial si V^* est en temps polynomial.

Nous ne démontrerons pas que ce simulateur vérifie bien les propriétés (3) et (4).

III.4.1.3. Remarques

Il y a plusieurs remarques à faire sur cet exemple.

III.4.1.3.1. Les langages de NP

Tout d'abord, il faut signaler que *G3C* est un langage *NP*-complet. Comme *G3C* possède un système de preuves interactives à divulgation nulle (avec entrée auxiliaire), on peut penser que c'est le cas de tous les langages de $\mathcal{NP}$. Cette intuition est exacte cependant la preuve ne sera pas faite ici. En fait, on peut aller plus loin et dire que la méthode utilisée pour construire P_{G3C} (c'est à dire en se basant sur des mises en gage) est identique pour construire un protocole zero-knowledge avec tous les langages de $\mathcal{NP}$. Malheureusement, il n'est pas garantit qu'il existe un système de preuves interactives à divulgation nulle efficace pour tout langage de $\mathcal{NP}$.

III.4.1.3.2. Un protocole non zero-knowledge

Intéressons nous au protocole suivant :

- Le prouveur connaît une trois-coloration ψ de G. Il envoie au vérifieur $|V|$ boites noires fermées à clef (des mises en gages) telles que la $v^{\text{ème}}$ contienne la valeur $\psi(v)$.
- Le vérifieur choisit aléatoirement une arête $(u,v) \in E$ et l'envoie au prouveur.
- Le prouveur envoie au vérifieur les clefs pour les boites u et v.
- Le vérifieur ouvre les dites boites et vérifie si elles contiennent deux couleurs dans $\{1,2,3\}$ différentes.
  Il accepte (retourne 1) si c'est le cas, sinon il refuse l'entrée.

Ce protocole, appelons le Π' est quasiment identique à celui qui a été donné un peu plus haut, disons Π, sauf que le prouveur ne prend pas la peine de générer à chaque fois une nouvelle trois-coloration ϕ avant de la mettre en gage.

Clairement Π' est un système de preuves interactives à divulgation non nulle. En effet, si V^* possède une entrée auxiliaire sur laquelle il mémorise le résultat de toutes itérations antérieures de Π', alors au bout de $O(|V|)$ itérations, il connaîtra ψ. Pourtant, on pourrait penser que le simulateur de Π peut être facilement adapté à Π' :

- Remplissage du ruban aléatoire du vérifieur avec une chaîne aléatoire $r \in \{1,2\}^{p(|x|)}$ (p est un polynôme qui limite le temps d'exécution du simulateur).
- Construction d'un pseudo trois-coloriage : le simulateur choisit aléatoirement et indépendamment $|V|$ valeurs $e_1,\ldots,e_{|V|} \in \{1,2,3\}$ et on note $\psi' : v \rightarrow e_v \ \forall v \in V$ le trois-coloriage ainsi construit.
- On répète un nombre polynomial de fois :
  - Simulation de (P'1) : Le simulateur met en gage ψ'.
  - Simulation de (V'1) : Le simulateur initialise V^* en plaçant r sur le ruban aléatoire de V^*, x sur son ruban d'entrée, z sur son ruban d'entrée auxiliaire et la mise en gage sur son ruban de communication entrante. Il exécute V^* dans ces conditions et note sa sortie ω (on suppose sans perte de généralité que $\omega \in E$).
  - Simulation de (P'2) et (V'2) : On note $\omega=(u,v)$. Si $e_u \neq e_v$, alors le simulateur s'arrête avec en sortie $(x,z,r,$ *« mise en gage »*$,(u,e_u,v,e_v))$. Sinon, il retourne $\perp$.

Donc il semblerait qu'on ait trouvé un simulateur pour un protocole qui n'est pas zero-knowledge. Si c'était le cas, ce serait particulièrement grave. En fait le simulateur de Π' n'est pas valide car il sort ⊥ avec une probabilité supérieure à ½. En effet, intuitivement, si on prend comme V^* un vérifieur qui mémorise les coups précédents, alors plus on va lancer V^* plus le simulateur retournera fréquemment ⊥.

III.4.2. Fiat-Shamir

Le langage des graphes isomorphes est un bon exemple de protocole zero-knowledge. Malheureusement, il n'est pas utilisable en pratique car il est difficile de manipuler des graphes de taille suffisamment importante pour empêcher les attaques par force brute (où le tricheur va essayer toutes les possibilités pour trouver un isomorphisme entre G_1 et G_2).

Nous allons donc voir maintenant un protocole zero-knowledge utilisé en pratique. Le protocole Fiat-Shamir est un protocole d'identification à clef publique.

III.4.2.1. Protocole

Une autorité choisit deux nombres premiers p et q et calcule $n = p.q$. Elle choisit ensuite $s \in [1,n-1]$ secret premier avec n et calcule $v=s^2 \bmod n$ public. Au final, l'entrée commune x est le couple (n,v) et le prouveur dispose sur son entrée auxiliaire de s (son secret).

Les étapes :

- Le prouveur choisit un nombre aléatoire r dans $[1,n-1]$ et calcule $x = r^2 \bmod n$.
  Il envoie x au vérifieur.
- Le vérifieur choisit $b \in \{0,1\}$ aléatoirement.
  Il envoie b au prouveur.
- Le prouveur calcule $y = r.s^b \bmod n$.
  Il envoie y au vérifieur.
- Si $y \neq 0$ et $y^2 \equiv x.v^b \pmod n$ alors le vérifieur accepte l'entrée (retourne 1) sinon il la refuse (retourne 0).

Soit sous forme schématique :

	Prouveur	(n,v)	Vérifieur	
(P1)	Choisit un nombre aléatoire r dans [1,n-1]. Calcule $x = r^2 \bmod n$.			
		x →		
			Choisit aléatoirement $b \in \{0,1\}$.	(V1)
		← b		
(P2)	Calcule $y = r.s^b \bmod n$.			
		y →		
			Vérifie si $y \neq 0$ et $y^2 \equiv x.v^b \pmod n$: si c'est le cas retourne 1 sinon 0.	(V2)

Le prouveur (grâce à son entrée auxiliaire) et le vérifieur peuvent tous les deux être implémentés efficacement (en temps polynomial).

III.4.2.2. Démonstration

La propriété de complétude est encore une fois évidente. Si le prouveur connaît s comme il le prétend, alors il peut calculer $y = r.s^b \bmod n$ et donc $y^2 \equiv r^2.s^{b2} \equiv x.v^b \pmod n$. Donc le vérifieur accepte toujours.

La propriété de validité. Si le prouveur est un imposteur et ne connaît pas s (ni p et q), alors il doit prévoir quel b le vérifieur va choisir pour calculer x en conséquence. Cependant comme il n'a aucun moyen de deviner ça et qu'il n'y a pas de x valable pour b=0 et b=1, alors le prouveur se trompe avec une probabilité de ½ et le vérifieur refuse avec la même probabilité.

Pour démontrer le caractère zero-knowledge de ce protocole, donnons son simulateur :

- Remplissage du ruban aléatoire du vérifieur avec une chaîne aléatoire $r \in \{1,2\}^{p(|x|)}$ (p est un polynôme qui limite le temps d'exécution du simulateur).
- Simulation de (P1) : Le simulateur choisit aléatoirement et indépendamment $y \in [1,n-1]$ et $e \in \{0,1\}$. Puis il calcule x tel que $x = \begin{cases} y^2 \bmod n & \text{si } e = 0 \\ y^2 \cdot v^{-1} \bmod n & \text{si } e = 1 \end{cases}$
- Simulation de (V1) : Le simulateur initialise V^* en plaçant r sur le ruban aléatoire de V^*, (n,v) sur son ruban d'entrée, z sur son ruban d'entrée auxiliaire et x sur son ruban de communication entrante. Il exécute V^* dans ces conditions et note sa sortie b (on normalise b de sorte que $b=0$ si $b \neq 1$ et $b=1$ sinon).
- Simulation de (P2) et (V2) : Si $b=e$, alors le simulateur s'arrête avec en sortie $((n,v),z,r,x,(b,y))$. Sinon, il retourne $\perp$.

III.4.2.3. Remarques

On peut remarquer sur ce protocole que le choix aléatoire de r est crucial. En effet, il ne faut surtout pas que le vérifieur puisse deviner r et notamment, il ne faut jamais réutiliser deux fois le même r. Dans ce cas en effet, si le vérifieur s'en apercevait, il pourrait découvrir le secret s en demandant pour les deux coups deux b différents : il obtiendrait deux réponses y_1 et y_2 telles que $y_1 = rs$ et $y_2 = r$, donc $s = \frac{y_1}{y_2}$.

Cette remarque est également valable pour le protocole basé sur *GI* (chapitre III.2 page 12) et aussi, mais dans une moindre mesure, pour celui basé sur *G3C*.

Cela signifie donc que les ensembles de départ (n pour Fiat-Shamir, (G_1,G_2) pour *GI*…) doivent être important pour avoir une probabilité de redondance la plus faible possible. D'autre part, ces ensembles doivent aussi être grands pour éviter, dans la pratique, des attaques par force brute.

En conclusion, on peut dire que les systèmes de preuves interactives à divulgation nulle ont été une grande découverte il y a vingt ans puisqu'ils ont permis de faire avancer la cryptographie aussi bien sur le plan théorique (développement de certaines zones de la cryptographie et de la théorie de la complexité) que sur le plan pratique (apparition de nouveaux protocoles ou amélioration de protocoles existants).

Ce dossier qui n'est qu'une présentation des protocoles zero-knowledge n'a pas la prétention d'être exhaustif mais il nous tout de même permis de donner une définition rigoureuse et d'effleurer quelques sujets intéressants comme la complexité liée à ces protocoles ou leur mise en application pratique. Ce sont là deux sujets qui pourraient faire l'objet de deux autres dossiers.