

T.C. GALATASARAY ÜNİVERSİTESİ  
MÜHENDİSLİK VE TEKNOLOJİ FAKÜLTESİ

BİLGİ AKTARIMI OLMADAN, İNTERAKTİF KİMLİK KANITLAMA  
(PROTOCOLES D'IDENTIFICATION ET PREUVE INTERACTIVE SANS  
APPORT DE CONNAISSANCE)

BİTİRME ÖDEVİ

Anet İZMİTLİ

Bölüm : BİLGİSAYAR MÜHENDİSLİĞİ  
Danışmanı : Dr. Emmanuel RIBOULET-DEYRIS

MAYIS 2005

## Préface

Ce projet, réalisé pour conclure mes études d'ingénierie informatique à l'Université Galatasaray, étudie globalement la cryptographie et surtout les différents protocoles d'identification, avec et sans apport d'informations.

Je veux surtout remercier Dr. Riboulet-Deyris pour m'avoir aidé dans mon choix du sujet, pour tout le support et l'aide qu'il m'a apporté pendant la réalisation de ce projet. Puis je remercie mes amis Kerem Kocaer et Ömer Kayış pour avoir réalisé avec moi, le mémoire : "La Cryptographie" en 2003 qui m'a été un bon guide. Et dernièrement, je remercie bien sûr Antoine Courbot, mes parents et ma soeur pour leur support et leur patience.

Anet Izmitli

16. 05. 2005

## Table des matières

<b>Préface</b>	<b>ii</b>
<b>Table des Matières</b>	<b>iii</b>
<b>Liste des Notations</b>	<b>v</b>
<b>Liste des Tableaux</b>	<b>vi</b>
<b>Résumé</b>	<b>vii</b>
<b>Özet</b>	<b>viii</b>
<b>Introduction</b>	<b>1</b>
<b>1 Protocoles d’Identification</b>	<b>3</b>
1.1 Introduction à la Cryptographie . . . . .	3
1.1.1 Histoire de la Cryptographie . . . . .	3
1.1.2 Cryptographie symétrique . . . . .	4
1.2 Cryptographie asymétrique . . . . .	5
1.3 Protocoles d’identification : les méthodes classiques . . . . .	9
1.3.1 Protocoles à une passe . . . . .	11
1.3.2 Protocoles à deux passes . . . . .	13
1.3.3 Protocole Kerberos . . . . .	13
<b>2 Introduction au Zero-Knowledge</b>	<b>16</b>
2.1 Présentation . . . . .	16
2.1.1 Définition et Exemples . . . . .	16
2.1.2 La trois colorabilité des graphes . . . . .	17

2.2	Protocoles "zero-knowledge" . . . . .	21
2.2.1	Principe général d'un protocole "zero-knowledge" . . . . .	21
2.2.2	Protocole de Fiat Shamir . . . . .	22
2.2.3	Protocole de Schnorr . . . . .	25
2.2.4	Protocole de Guillou Quisquater . . . . .	29
<b>3</b>	<b>Application</b>	<b>33</b>
3.1	Présentation du logiciel de calcul formel Magma . . . . .	33
3.2	Implémentation : . . . . .	34
3.2.1	Protocole Fiat Shamir . . . . .	34
3.2.2	Protocole de Schnorr . . . . .	36
3.2.3	Protocole de Guillou-Quisquater . . . . .	39
3.3	Tests et Résultats . . . . .	41
	<b>Conclusion</b>	<b>43</b>
	<b>A Algorithme d'Euclide</b>	<b>47</b>
	<b>B Théorème de Bézout</b>	<b>49</b>
	<b>C Le Théorème Chinois</b>	<b>51</b>
C.1	Le cas général . . . . .	51
C.2	Théorème Chinois pour les entiers R.S.A. . . . .	52
	<b>D Théorème de Fermat</b>	<b>54</b>
	<b>E Fonctions à sens unique</b>	<b>57</b>
E.1	Fonctions à sens unique . . . . .	57
E.2	Fonctions à sens unique à trappes . . . . .	59

## Liste des Notations

AES : Advanced Encryption Standard

CPU : Central Processing Unit

DES : Data Encryption Standard

J.C. : Jésus-Christ

NP : Non Deterministe Polynomial

R.S.A. : Rivest, Shamir, Adleman

## Liste des Tableaux

Tableau(2.1) : Protocole de Fiat Shamir

Tableau(2.2) : Protocole de Schnorr

Tableau(2.3) : Protocole de Guillou Quisquater

Tableau(3.1) : Résultats des tests

## Résumé

Dans le cadre du projet de fin d'études, j'ai travaillé, avec M. Riboulet sur les protocoles d'identification. Ce sujet est à la fois mathématique et informatique et si je l'ai choisi, c'est car j'ai pensé que ce serait une bonne occasion pour moi de conclure mes études d'ingénierie avec un sujet qui réunit ces deux domaines essentiels, surtout à l'heure actuelle.

Les protocoles d'identification sont, bien sûr, basés sur la cryptographie. En effet, nous possédons tous des informations que nous voulons protéger, nous faisons attention à les communiquer d'une façon discrète aux gens concernés pour que les autres ne les apprennent pas. C'est quand ces gens concernés sont loin de nous qu'il y a un problème car de là, naît le besoin d'envoyer le message, donc il faut que le canal de communication soit sécurisé ou que le message soit crypté pour qu'un pirate ne puisse pas accéder au message clair. La cryptographie est issue de cette problématique et elle s'appuie sur un grand nombre de notions mathématiques.

D'autre part, il faut aussi s'assurer de l'émetteur du message. Les protocoles d'identification permettent cela. La sécurité de ces protocoles est basée sur des notions comme la difficulté de la factorisation des grands nombres (qui sont le produit de deux grands nombres premiers), le logarithme discret etc. . .

Durant ce projet, la problématique de chiffrement et d'identification ont donc été abordé afin de les comprendre et de les mettre en oeuvre avec un langage de programmation : Magma.

## Özet

İnsanođlu varolduđundan beri hep yeni bilgiler peşinde koşmuştur. Bir yandan bu bilgilere ulaşırken, diđer yandan da ulaştığı bilgilerin bazılarını saklama telaşındadır. İşte bu noktada ortaya şifreleme kavramı çıkmıştır ve çok eskilerden günümüze kadar birçok yeni yöntemler üretilmiştir.

Saklanması gereken sırların da bir önem sırası vardır tabii, hoşlandığımız kişiyle aramızdaki son durumu en yakın arkadaşımızla paylaşmamız da sırdır, savaş esnasında bir ülkenin diđer ülkeye nükleer bomba atılması emrini vermesi de... Aslında şifrebilim daha çok ikinci tür sırlarla ilgilenir. Bu sırların başkaları tarafından öğrenilmemesi çok hayati önem taşıdığı için, bu bilgilerin bir elden diđer ele yolculuđu sırasında karşılarına çıkabilecek her korsanı atlatabilmeleri gerekmektedir. Dolayısıyla çok ciddi güvenlik önlemleri alınmalıdır.

Projenin ilk kısmında incelediğimiz konu da bu zaten : şifrebilim yani kriptografinin geçmişi ve kullanılan farklı yöntemler.

Güvenlik önlemlerinin artması gerektiğçe, yeni şifreleme yöntemleri ortaya çıktı. Bu yöntemleri simetrik ve asimetrik olarak ikiye ayırmamız mümkün.

Simetrik şifrelemede mesajı gönderen ve alan kişi arasında bir tane anahtar kullanılır. Mesaj bu anahtarla şifrelenir ve yine aynı anahtarla şifre açılır. Bu yöntemle ortaya çıkan en büyük problem, bu anahtarın mesajı atan ve alan kişiye ulaştırılmasıdır. Anahtar yollanırken aktarım hattını dinleyen korsan, anahtara sahip olabilir ve gönderilen mesajları okuyabilir. Ayrıca konuşmanın güvenliđi için her konuşan çift arasında ayrı bir anahtar oluşturulması gerekmektedir. Dolayısıyla, 10 kişinin aralarında konuştuđunu varsayarsak, her kişi 9 farklı kişiyle konuştuđu için, toplamda  $10 \times 9 = 90$  anahtara gereksinim vardır. Her çift tek anahtar kullandıđı için bu sonuç

2'ye bölünür ve sonuçta her kişide 9'ar, toplamda da 45 anahtar yaratma gereksinimi doğar ki bu da hiç pratik bir yöntem değildir.

Simetrik şifreleme yöntemlerinin ardından, asimetric şifreleme yöntemleri ortaya çıktı. Bu yöntemlerin özelliđi her kişinin iki farklı anahtara sahip olmasıdır. Bunlardan bir tanesi şifreleme anahtarıdır ve o herkes tarafından bilinir. Diđeri ise şifreyi açıp asıl mesaja ulaşılmasını sađlayan ve sadece anahtar sahibi tarafından bilinen anahtardır.

Bir kişiye şifreli bir mesaj yollamak isteyen kişi, mesajı alan kişinin herkes tarafından bilinen şifreleme anahtarıyla mesajı şifreler ve şifreli mesajı alan kişi, şifreleme anahtarının eđi olan deşifre etme anahtarı yardımıyla mesajı açıp okuyabilir. Asıl mesajı açmaya yarayan bu anahtar asla el deđiştirmediđi için asimetric şifreleme, simetrik şifrelemeden çok daha güvenlidir.

Bu anahtar çiftlerini yaratmanın farklı yöntemleri ve günümüzde kullanılan farklı asimetric şifreleme algoritmaları vardır. Biz bu projede özellikle R.S.A. şifreleme yöntemini inceledik çünkü R.S.A., şifreleme yöntemlerinin şu anda en çok kullanılanı ve bunun sebebi de şüphesiz güvenliđi ve hızlı bir şekilde çalışabilmesi.

Internet gelişip hepimizin hayatına girince, tüm yollanan veriler Internet üzerinden yollanmaya başladı ve bu da tabii güvenlik açısından ciddi bir sorun yarattı. Normalde bir mektup yolladıđımızda altını imzalar ve çok önemli bir belgeyse, belgenin gideceđi yere kadar açılmadıđını kanıtlamak için zarfı kaşeleriz.

Artık tüm işlemler Internet üzerinden yapıldıđı için imza ve kaşe olaylarını da Internet'e taşımak gerekti ve bunun sonucunda elektronik imza kavramı ortaya çıktı.

Daha sonra asıl projemin konusu olan kimlik denetleme işlemine gereksinim duyulmaya başlandı. Bir bilgisayara veya bir sisteme bađlandıđımızda, önce kendimizi tanıtmamız gerekir ki sistem oraya bađlanmaya hakkımız olup olmadığını anlayabil-sin. Bu kimlik tanımlama işinin birçok farklı yöntemi vardır.

Bir sisteme bağlanmak istediğimizde, sistemin bizi diğer kişilerden ayırabilmesi için her kişinin bir sırrı olması gerekiyor. Sisteme bağlandığımızda tanınabilmemiz için yapabileceğimiz en basit şey şüphesiz şifremizi sisteme söylememizdir. Zaten bu hepimizin Windows veya Unix bilgisayarlarımıza bağlanmak için kullandığımız yöntemdir. Fakat bu yöntemin çok ciddi bir güvenlik açığı olduğu da bir gerçek. Şifremizi girerken biri bizi görebilir veya şifremizi yollarken biri Internet hattını dinliyor olabilir ve daha sonra bizim şifremizi kullanarak sisteme bizim adımıza bağlanabilir. Hatta kimliğimizi bir kişiye kanıtlıyorsak eğer, kendisine şifremizi verdiğimiz için kendisi başka bir yere bizim adımıza bağlanabilir. Dolayısıyla bu yöntem çok güvenli değildir.

İkinci bir yol asimetrik şifrelemede gördüğümüz yöntemleri kullanmaktır. Alice kişinin Bob kişiye bağlanmak istediğini varsayalım. Bu durumda Alice'in sırrı, kendi şifreleme anahtarıyla şifrelenmiş mesajları açmasına yarayan gizli anahtar oluyor. Bob, Alice'in şifreleme anahtarını kullanarak, herhangi bir mesajı şifreler ve Alice'e yollar. Alice, gizli anahtarıyla şifreyi açar ve mesajı Bob'a geri yollar. Bob, kendi seçtiği mesajla Alice'in yolladığı mesajı karşılaştırır ve aynı olması durumunda karşısındakinin Alice olduğundan emin olur çünkü o mesajı sadece Alice'in sahip olduğu gizli şifre açabilir. Alice gizli şifresini kimeseye paylaşmadığı için, başka birinin Alice olarak kendini tanıtmaya çok zordur.

Son ve en güvenli yöntem ise bilgi aktarımı olmadan yapılan kimlik kanıtlama protokolleridir. Bu protokollerin en büyük farkı, Alice'in kimlik kanıtlama işlemi sırasında, Bob'a veya hattı dinleyen herhangi bir korsana sırrı hakkında hiç bir bilgi vermemesidir. Zaten bu yüzden bu protokollere "bilgi aktarımı olmadan" yapılan protokoller denir.

Bu protokollerin bazı özelliklerinin olması gerekiyor :

- Bu protokoller iki veya daha fazla kişi arasında gerçekleştirilir (mesela Alice ve Bob),
- Alice aslında bilmediği bir sırrı bildiğini söyleyerek Bob'u kandıramaz,

- Bu protokoller Alice olduğunu söyleyen kişinin doğru söylediğini yüzde yüz kanıtlamaz fakat protokolün Bob tarafından birçok defa tekrar edilmesiyle, Bob'un kandırılma ihtimali çok küçüğe indirgenebilir,
- Bob, korsanlık yapmak istese dahi, Alice'in sırrı hakkında asla işe yarar bir bilgi edinemez,
- Bob, Alice'in sırrı hakkında işe yarar bir bilgi edinemediği için, üçüncü bir kişiye kendini Alice olarak tanıtmaması mümkün değildir, ki bu da güvenlik açısından çok önemlidir.

Bilgi aktarımı olmadan yapılan bir çok protokol vardır. Biz bu projede Fiat Shamir, Schnorr ve Guillou-Quisquater protokollerini inceledik. Bu protokoller üç ana adımdan oluşmuşlardır ve hepsinde de tek yönlü fonksyonlardan faydalanılmıştır. Tek yönlü fonksyonlar,  $x$ 'i bilmemiz durumunda  $f(x)$ 'i kolaylıkla hesaplayabildiğimiz fakat elimizde sadece  $f(x)$  olduğunda,  $x$ 'i bulmamızın çok zor, hatta imkansız olduğu fonksyonlardır.

Bu protokollerde kimliğini kanıtlayan kişinin (yani Alice'in), sadece kendisinin bildiği bir sırrı vardır ve bu sırrı tek yönlü bir fonksyondan geçirerek, bulduğu sonucu yayımlar. Bu protokoller üç ana adımdan oluşur :

1. Alice, bir sayı seçer ve o sayıyı tek yönlü bir fonksyondan geçirerek sonucu Bob'a yollar. Fonksyon tek yönlü olduğu için Bob eline geçen sonuçtan yola çıkarak Alice'in seçtiği sayıya ulaşamaz ve Alice bu sayıyı Bob'a yollayarak daha sonra hile yapmayacağını garantisini vermiş olur,
2. Bob rastgele bir sayı seçer ve Alice'e yollar,
3. Alice sırrını, ilk adımda seçmiş olduğu sayıyı ve Bob'un ikinci adımda yolladığı sayıyı kullanarak bir işlem yapar (ki bu işlem her protokolde farklıdır), ve yanıtı Bob'a yollar. Bob ise Alice'in sırrının tek yönlü fonksyondan geçmiş halini, ilk adımda Alice'in yolladığı sonucu ve ikinci adımda kendi seçtiği sayıyı kullanarak Alice'in sırrı gerçekten bilip bilmediğini kontrol eder.

Bütün bu işlemler sonunda, Bob, Alice'in sırrı hakkında hiç bir önemli bilgi edinemez, tek öğrendiği, Alice'in o sırra sahip olduğudur.

Bu protokoller diğer kimlik kanıtlama protokollerinden, bilgi aktarımı olmaması yönünden daha güvenlidir ve farkında olmasak da Internet üzerinden yaptığımız bir çok işlemde biz de bu protokolleri kullanıyoruz. Matematik ve bilgisayar dünyasında yapılan araştırmalar çok hızlı geliştikleri için, bugün şartlarında yeteri derecede güvenli olan bu protokoller, birkaç sene içinde yerlerini daha farklı protokollere bırakacaklar.

Projenin sonunda, incelediğimiz bu üç protokolü Magma dilinde programlayarak nasıl çalıştıklarını ve farklı şartlardaki performanslarını inceledik.

Bu proje bana, zaten ilgimi her zaman çekmiş olan şifreleme ve güvenlik konusunda bilgilerimi genişletme imkanını tanıdı. Yaptığım çalışma, bu konulara olan ilgimi daha da arttırdı ve yanısıra, master için de bu konulara yönelme isteğimin oluşmasına neden oldu.

## Introduction

Depuis toujours, l'homme a eu des informations qu'il n'a voulu partager qu'avec un nombre limité de gens. Celles-ci peuvent être des informations personnelles ou même des informations d'une très grande importance, comme l'éventuelle explosion d'une bombe nucléaire ou le début d'une guerre etc... C'est ce qui a fait naître la notion de *cryptographie* qui consiste en le chiffage et le déchiffage des informations.

Puis sont nées les notions de *signature*, d'*authentification* et d'*identification*. La signature consiste à prouver que l'émetteur est bien l'auteur du message envoyé. L'identification et l'authentification, consistent, d'une part, à permettre à une entité de se faire reconnaître d'un système par la communication d'un élément dont elle l'a dotée, d'autre part, à apporter la preuve de son identité.

Plus précisément ces protocoles cryptographiques sont une série d'étapes prédéfinies, qui permettent à plusieurs participants (généralement deux) de se prouver leur identité. Dans ce rapport, nous allons décrire dans un premier temps quelques notions principales de la cryptographie, ce qui nous permettra de bien situer le contexte et de nous familiariser avec ce domaine scientifique. Pour ce faire, dans le premier chapitre, nous allons tout d'abord parler un peu de l'histoire de la cryptographie et puis de ses deux notions principales : la *cryptographie symétrique* que nous allons expliquer sans donner beaucoup de détails, et la *cryptographie asymétrique* que nous étudierons plus profondément puisque ses notions nous serviront beaucoup pour la suite. Parmi les différents cryptosystèmes asymétriques nous allons spécialement nous intéresser au système R.S.A. car c'est le plus utilisé de nos jours.

Pour finir le premier chapitre, nous allons décrire la notion d'identification, et nous verrons les différents protocoles d'identification, notamment les protocoles à une

pas, à deux passes et le protocole Kerberos dont l'utilisation est actuellement très fréquente.

Dans le deuxième chapitre, nous allons enchaîner notre étude sur les protocoles d'identification avec les protocoles sans apport de connaissance (c'est là, le coeur de notre étude) qui est un domaine très à la mode, grâce aux avantages qu'il amène. Après avoir parlé du problème de trois coloriage des graphes dont le protocole est théoriquement très performant mais qui ne peut pas être vraiment pratiqué, nous présenterons les protocoles les plus connus, notamment le protocole de Fiat Shamir, le protocole de Schnorr et le protocole de Guillou-Quisquater afin de comprendre le mode de fonctionnement de ces derniers. Nous verrons, à cette occasion, les protocoles les plus performants.

Dans le dernier chapitre, nous mettrons ces protocoles en application à l'aide du langage Magma pour pouvoir faire des tests avec différents paramètres. L'objectif sera ici de type algorithmique et nous permettra de comprendre plus finement les protocoles sans apport de connaissance développés dans le chapitre précédent.

Ainsi, ce mémoire nous aura permis de balayer les méthodes classiques d'identification, qu'elles soient ou non sans apport d'information et plus théoriquement de nous familiariser avec les concepts de base de la théorie de la complexité pour la cryptographie.

# Chapitre 1

## Protocoles d'Identification

Dans ce chapitre, nous allons commencer par introduire un peu la notion de cryptographie, et pour ce faire, nous allons parler un peu de son histoire cryptographie et de ses deux notions principales, notamment la cryptographie symétrique que nous allons expliquer sans donner beaucoup de détails et la cryptographie asymétrique qui nous servira beaucoup dans la suite du projet. Nous allons conclure notre chapitre par la présentation de la notion d'identification et nous allons citer quelques exemples de protocoles d'identification.

### 1.1 Introduction à la Cryptographie

#### 1.1.1 Histoire de la Cryptographie

Depuis toujours, les gens ont senti le besoin de cacher des informations personnelles et confidentielles, d'avoir des secrets qu'ils ne partagent qu'avec un nombre limité de gens, et cela bien avant l'apparition de l'informatique.

Le premier exemple de la cryptographie date du XVI<sup>e</sup> siècle avant J.C.. Il s'agit d'une tablette d'argile retrouvée en Irak. Le texte qui est gravé sur cette tablette, est codé. Il était inscrit sur cette tablette la recette du succès d'un potier babylonien ; comme il ne voulait pas la partager avec tout le monde, il l'avait masqué en la chiffrant. Après avoir rajouté quelques consonnes et reconnu les mots à l'orthographe fantaisiste, les archéologues ont réussi à déchiffrer la formule de fabrication d'un vernis que le potier gardait pour lui. L'histoire ne dit pas si sa ruse a été efficace.

Puis du V<sup>e</sup> au II<sup>e</sup> siècle avant J.C., les Grecs antiques, réputés pour leur art de la guerre, voulaient protéger les messages secrets. Ils ont donc mis au point un système efficace pour coder leurs messages. L'expéditeur enroulait une bande de papyrus en spires parallèles sur un bâton de diamètre défini. Il écrivait ensuite son message transversalement dans le sens du bâton. En déroulant la bande, le message devenait incompréhensible, à part pour la personne qui possédait un bâton de même diamètre (donc la personne qui possédait la clé du chiffrement).

Et pour l'informatique... Depuis sa création, qui paraît aujourd'hui assez lointaine, le réseau Internet a tellement progressé et s'est tellement ouvert au public, qu'il est devenu un outil essentiel de communication. Il est devenu donc primordial de garantir la sécurité et l'intégrité de l'information. C'est la cryptographie qui s'en charge.

Le terme *cryptologie* désigne la science dont l'objet d'étude est l'ensemble des techniques permettant de chiffrer, de déchiffrer ou encore de tester la probabilité de casser des messages. Le terme *cryptographie* ne représente lui que l'ensemble des méthodes de chiffrement de l'information.

La cryptographie comporte actuellement deux notions : la cryptographie symétrique et la cryptographie asymétrique...

### 1.1.2 Cryptographie symétrique

Un système de chiffrement à clé privée, dit aussi symétrique, consiste en un partage d'une même clé entre deux personnes qui veulent communiquer. Cette clé est utilisée à la fois pour le chiffage et le déchiffage. On appelle cette clé "secrète" du fait qu'elle n'est connue que par l'expéditeur et le destinataire.

Mais il faut dire que cette méthode de chiffrement a des inconvénients assez importants : premièrement, nous rencontrons le problème de la transmission de la clé au destinataire. Si cette transmission est interceptée par un pirate, celui-ci peut lire des

messages et il peut même en écrire et le faire passer à l'un des correspondants. Un deuxième problème important apparaît lorsqu'on veut faire parvenir le message à plusieurs personnes à la fois. Imaginons qu'il y ait  $N$  personnes qui veulent communiquer, comme chaque personne a  $(N - 1)$  correspondants, il faut donc  $N \times (N - 1)$  clés. Mais comme pour chaque couple une clé est suffisante, on divise ce résultat par 2 et on constate qu'il faut distribuer

$$\frac{N \times (N - 1)}{2}$$

clés, donc il faut considérer le temps de génération de toutes ces clés qui implique un temps global important. C'est notamment le problème des portefeuilles, la démultiplication des clés est un risque de sécurité évidente.

Il y a différentes sortes de protocoles de cryptage à clé secrète comme celui de César [6], de Vigenère [7], ou plus récemment, le fameux DES [8] (Data Encryption Standard) qui a été créé par IBM Corp et aussi il y a le cryptosystème AES [9] (Advanced Encryption Standard) qui a été créé en 1997 et qui va probablement remplacer DES....

## 1.2 Cryptographie asymétrique

La cryptographie à clé publique, dite aussi asymétrique, est un concept inventé en 1975 par deux ingénieurs en électronique de l'Université de Stanford : Whitfield Diffie et Martin Hellman.

Le chiffage asymétrique se diffère du chiffage symétrique par le fait qu'il utilise des fonctions "à sens unique" (Voir Appendice E, page 57), c'est à dire des fonctions qui sont faciles à calculer dans un sens mais très difficiles dans l'autre. C'est donc très difficile de trouver l'inverse de la fonction pour pouvoir faire le déchiffage, sans connaître une information en plus (Voir Appendice E.2, page 59). Ceci-dit, trouver l'inverse n'est pas impossible ; mais avec les moyens mathématiques et informatiques

d'aujourd'hui, ce calcul durerait un temps énorme, voire même des millions d'années de temps CPU.

Dans ce modèle de cryptographie, chaque personne possède sa clé privée qui est secrète et sa clé publique qui est connue par tout le monde. Supposons qu'Alice veuille envoyer un message chiffré avec une méthode de cryptographie à clé publique et examinons les étapes pas à pas :

- Bob a une clé publique qu'il annonce aux gens qui veulent lui envoyer un message et une clé privée pour déchiffrer les messages qu'il reçoit.
- Alice, récupère la clé publique de Bob et s'en sert pour chiffrer le message qu'elle veut lui transmettre.
- Bob qui reçoit le message chiffré, utilise sa clé privée pour la déchiffrer et obtient le message clair.

Si, pendant la transmission, un pirate réussit à obtenir le message crypté, il ne pourra pas le déchiffrer comme il ne possède pas la clé privée correspondante.

Comme exemple à ce type de cryptage, on peut citer Diffie-Hellman, El-Gamal ou R.S.A., nous allons juste parler de R.S.A. dans cette partie car ce sera utile pour la suite.

### **R.S.A.**

Après la proposition d'une cryptographie à clé publique par W. Diffie-M. E. Hellman (pour plus de détails voir [10]), trois mathématiciens de MIT : Ron **R**ivest, Adi **S**hamir et Len **A**delman, qui n'étaient pas convaincus de la sécurité de ce système cryptographique, ont voulu prouver que tout système à clé publique possédait une faille. Cependant, au lieu d'aboutir à ce résultat, ils ont inventé, en 1978, une nouvelle méthode de cryptographie basée sur ce même concept de clé publique : le système R.S.A. (voir leur article [11] pour plus de détails).

Premièrement essayons de voir quelles sont les données nécessaires pour utiliser ce cryptosystème, comment on chiffre, comment on déchiffre et sur quoi repose sa sécurité.

Considérons Alice et Bob qui désirent communiquer entre eux et qu'ils choisissent le système R.S.A. pour assurer la confidentialité de leur conversation. Bob génère deux nombres grands premiers  $p$  et  $q$  qu'il garde secret, puis il multiplie ces deux nombres et il obtient  $n = p \times q$ . Ce nombre  $n$  est publique. Ensuite il calcule  $\phi(n) = (p - 1) \times (q - 1)$ , et il génère un nombre  $e$ , premier avec  $\phi(n)$ ,  $e$  est le paramètre de chiffrement. Il calcule l'inverse de  $e$  modulo  $n$ , ce qui donne  $d$ , le paramètre de déchiffrement. La clé publique de Bob est le couple  $(n, e)$  et la clé privée est  $(p, q, d)$ .

Alice, qui veut envoyer le message  $m \in \mathbb{Z}/n\mathbb{Z}$  à Bob, récupère tout d'abord la clé publique de Bob et elle chiffre son message en faisant le calcul  $m^e \pmod n$ , et elle envoie ce message chiffré à Bob. Bob ayant reçu  $m^e \pmod n$ , utilise  $d$ , sa clé privé de déchiffrement, et il calcule :  $(m^e)^d$  pour récupérer la valeur de  $m$ .

Nous avons  $e \times d \equiv 1 \pmod{\phi(n)}$ . Donc  $(m^e)^d \equiv m^{ed} \equiv m^{1+k\phi(n)} \equiv m \times (m^{\phi(n)})^k \pmod n$  et  $(m^{\phi(n)})^k \equiv 1 \pmod n$ . Enfin  $(m^e)^d \equiv m \pmod n$ .

Bob, sans connaître la valeur de  $d$ , n'aurait pas pu déchiffrer le message  $m$ . Nous verrons par la suite que la sécurité du cryptosystème R.S.A. repose sur la difficulté de factoriser  $n$ . Celui qui est capable de factoriser  $n$  et donc de retrouver  $p$  et  $q$  est capable de casser le cryptosystème R.S.A..

**Fait :** Si on sait factoriser  $n$  alors on peut casser R.S.A.(car on peut retrouver  $d$ ).

**Démonstration :** Nous avons dit que nous savons factoriser  $n$  donc le pirate Oscar, connaît  $p$  et  $q$ , il peut facilement calculer  $\phi(n)$  aussi puisque  $\phi(n) = (p - 1) \times (q - 1)$ , il suffit qu'il utilise l'algorithme de Bézout qui dit que si  $e$  est premier avec  $\phi(n)$  alors

il existe deux entiers  $u$  et  $v$  tels que  $ue + v\phi(n) = 1$  (Voir Appendice B, page 49). Dans ce cas le  $d$  que nous cherchons correspond à  $u$ .

La complexité de ce calcul est  $O(\ln e \times \ln \phi(n))$  qui est de l'ordre de  $O((\ln n)^2)$ , c'est un algorithme polynomial en la taille de  $n$ .

Maintenant essayons de voir dans l'autre sens. Supposons qu'Oscar possède une méthode qui lui permette de casser le cryptosystème R.S.A. (donc la connaissance de  $e$  lui suffit pour calculer  $d$ ), alors est-ce qu'il est capable de retrouver  $p$  et  $q$ ?

Pour pouvoir étudier ce cas, nous aurons besoin du Théorème Chinois (Voir Appendice C.2, page 52),

Oscar sait retrouver le message clair, en partant du message chiffré, il sait donc calculer :

$$m^{ed} \equiv m \pmod{n}, \quad m \not\equiv 0 \pmod{n}$$

↓

$$m^{ed-1} \equiv 1 \pmod{n}$$

Nous savons que  $ed \equiv 1 \pmod{\phi(n)}$  donc :  $\phi(n)$  divise  $ed - 1$ , en plus  $p$  et  $q$  sont des grands nombres premiers donc impairs et  $\phi(n) = (p - 1)(q - 1)$ , nous en concluons que  $\phi(n)$  est pair (donc  $ed - 1$  aussi). Nous pouvons alors noter  $ed - 1$  de la façon suivante :

$$ed - 1 = 2^u v \text{ avec } u \in \mathbb{N}(\geq 2) \text{ et } v \in \mathbb{N}(\text{impair}).$$

Oscar sait calculer une suite du type :

$$m^v, m^{2v}, \dots, m^{2^{u-1}v}, m^{2^u v} \equiv 1 \pmod{n} \text{ et pour l'instant } p \text{ et } q \text{ sont inconnus.}$$

Mais en théorie nous savons que  $\mathbb{Z}/n\mathbb{Z} \leftrightarrow \mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z}$  (voir Appendice C.2, page 52) et les racines carrés de 1 sont  $(1, -1)$  pour  $\mathbb{Z}/p\mathbb{Z}$  et  $\mathbb{Z}/q\mathbb{Z}$ , ce qui signifie que nous avons les couples  $(1, 1), (1, -1), (-1, 1), (-1, -1)$ .

Donc la réponse est 1,  $-1$  ou deux nombres différents de 1 et  $-1$ , appelons les  $a$  et  $b$ . Nous savons que  $a^2 = b^2 = 1 \pmod n$ .

$a^2 \equiv 1 \pmod n$  donc  $a^2 - 1 \equiv 0 \pmod n \rightarrow (a+1) \times (a-1) \equiv 0 \pmod n$  mais comme  $a \neq 1$  et  $-1$  nous savons que  $(a+1) \neq 0$  et  $(a-1) \neq 0$ .

Par contre nous savons que  $pq$  divise  $(a-1)(a+1)$  donc pour retrouver  $p$  et  $q$ , il suffit de calculer :

$\text{pgcd}((a-1), n) = p$  ou  $q$ ,  $\text{pgcd}((a+1), n) = q$  ou  $p$ .

La probabilité de retrouver  $a$  ou  $b$  à partir de  $m^{2^{u-1}v}$  est de  $1/2$ , et une fois que nous avons  $a$  ou  $b$  c'est gagné, nous avons  $p$  et  $q$ ...

Ainsi, nous avons vu que si nous savons factoriser  $n$ , nous savons casser le système R.S.A. et aussi si nous savons casser le système R.S.A., nous savons factoriser  $n$ . Le paramètre de sécurité du système R.S.A. est l'entier  $n$ , plus sa factorisation est difficile, plus notre système est sécurisé.

Examinons finalement l'état des lieux du cryptosystème R.S.A. :

- C'est le cryptosystème le plus utilisé
- En 1996, R.S.A.-130, (R.S.A.-130 signifie que l'entier  $n$  est de 130 bits)
- En 1999, R.S.A.-155,
- En 2003, R.S.A.-160 a été cassé.
- Durant l'été 2005, il est prévu que R.S.A.-200 soit cassé...

A l'heure actuelle, les tailles recommandées sont de l'ordre de 1024-4096 bits.

### 1.3 Protocoles d'identification : les méthodes classiques

De manière générale, un *protocole* est un ensemble de règles régissant des interactions de nature diverses. Pour l'informatique, c'est une méthode *standard* qui permet la communication entre des processus (s'exécutant éventuellement sur différentes

machines), donc c'est un ensemble de règles et de procédures à respecter pour émettre et recevoir des données.

Les *protocoles d'identification*, se déroulent entre deux machines, dites le *prouveur*, que nous allons nommer Alice et le *vérifieur*, Bob. Le but d'Alice est de prouver à Bob, qu'elle est bien la personne qu'elle prétend être. Plus précisément, elle possède un secret et son but c'est de prouver à Bob qu'elle est bien Alice puisqu'elle connaît ce secret. Il y a différentes façons de faire cela, nous les étudierons dans la suite de ce projet.

Un protocole doit assurer plusieurs critères comme (voir dans [1]) :

- *la confidentialité*, ce qui veut dire que l'information partagée entre Alice et Bob, ne sera pas vu par quelqu'un d'autre, c'est les deux personnes autorisées qui auront accès à l'information.
- *l'authentification*, ce qui signifie qu'Alice doit pouvoir assurer Bob de son origine,
- *l'intégrité*, ce qui veut dire que Bob, doit être sûr que le message n'a pas été modifié,
- *la non répudiation ou le non désaveu*, c'est-à-dire qu'Alice ne doit pas pouvoir nier avoir envoyé ce message.

Tous les procédés d'identification font intervenir une donnée secrète, censée être connue seulement de la personne autorisée. Pour s'identifier, elle doit prouver qu'elle connaît cette donnée secrète et c'est le point le plus important de l'identification. Alice, qui veut échanger une donnée avec Bob, doit tout d'abord lui prouver qu'elle est bien Alice. La première méthode qui vient à l'esprit, c'est qu'elle lui envoie sa signature ou une donnée propre à elle qui prouve bien son identité (Voir Section 1.3.1, page 11). Mais il apparaît clairement que ce n'est pas une méthode sécurisée donc pas une méthode utilisée puisque Bob, qui a la donnée qui prouve l'identité d'Alice peut l'utiliser, à son tour, avec d'autres personnes et il peut bien se faire passer pour Alice. Ce n'est pas le seul risque. . . Bob peut être quelqu'un d'honnête et il peut ne pas utiliser les données qu'il a reçu de la part d'Alice, mais Oscar, le

pirate, peut écouter la ligne de conversation d’Alice et de Bob, et il peut obtenir les identifiants d’Alice. . .

Alors comme cette méthode est loin d’être sécurisée, il a fallu en chercher une autre. La méthode de nos jours utilisée est la suivante : Alice possède un secret, qu’elle ne partage avec personne. Elle fait son identification auprès de Bob en lui prouvant sa connaissance de ce secret. Mais en aucun cas, elle ne le partage avec qui que ce soit. Elle fait cela grâce à des protocoles que nous allons détailler dans la suite.

Il y a trois sortes de protocoles d’identification :

1. Protocoles à une passe ;
2. Protocoles à deux passes ;
3. Protocoles sans divulgation d’information (Zero-Knowledge).

### 1.3.1 Protocoles à une passe

– Les procédés d’identification à *mot de passe* :

La personne qui est autorisée, donc Alice, prouve à Bob qu’elle connaît le secret, en lui envoyant le secret en question. C’est une méthode assez souvent employé, par exemple dans le système d’exploitation UNIX. Un utilisateur, qui veut se connecter sur son compte sur une machine UNIX, va taper son mot de passe et dans le cas où le mot de passe est le bon, il aura passé le test. C’est également la méthode utilisée dans les transactions avec une carte bancaire à puce. L’inconvénient des procédés à mot de passe est qu’il n’y a qu’un seul secret et ce secret est communiqué au vérifieur, donc un attaquant qui voit le secret pendant l’identification peut s’identifier au nom d’Alice une prochaine fois. L’attaquant peut même être le vérifieur lui-même, donc il obtient le secret directement, sans aucun besoin d’attaque.

Maintenant voyons ce qui se passe quand nous nous connectons à notre compte sous Linux, comment sont conservées les mots de passes ?

Il y a un paquetage qui s'appelle "**Shadow Suite**" (voir [5] pour plus d'informations sur ce paquetage), nous allons voir comment se fait la conservation des mots de passes quand ce paquetage n'existe pas et dans le cas inverse. . .

- Si dans le système le **Shadow Suite** n'est pas installé alors les informations concernant les utilisateurs sont conservés dans `/etc/passwd`. Le mot de passe est conservé d'une manière *encodée* avec la fonction `crypt(3)`.

Pour encoder un mot de passe on prend les 7 premiers bits de chaque 8 premiers caractères du mot de passe, donc cela nous fait une clé de 56 bits. Puis on prend une valeur aléatoire qu'on appelle *salt* et on l'ajoute au mot de passe qu'on crypte avec la clé créée auparavant. Puis cette valeur est sauvegardée dans le répertoire `/etc/passwd` avec la valeur de *salt* et il faut noter que ce répertoire est accessible par tous les utilisateurs mais comme les mots de passes sont gardés d'une façon cryptés c'est quand même sécurisé.

Quand un utilisateur veut se connecter à son compte, l'ordinateur prend le mot de passe que l'utilisateur tape, puis cherche le *salt* utilisé pour crypter son mot de passe et refait l'encryption, puis il regarde si la valeur enregistrée dans l'ordinateur et la valeur trouvée en cryptant le mot de passe tapé sont pareilles ou pas.

- Si dans le système le **Shadow Suite** est installé alors les mots de passes sont enregistrés dans un répertoire qui s'appelle `/etc/shadow` au lieu de `/etc/passwd` et ce répertoire n'est accessible que par le *root*, ce qui signifie qu'un pirate ne peut même pas avoir accès aux mots de passes cryptés donc cela augmente la sécurité. (Ce paquetage a également d'autres fonctionnalités mais ici, nous avons indiqué que celle-ci vu que c'est celle qui nous intéresse).

### **Le schéma de Lamport**

Ce schéma utilise une fonction à sens unique  $f$  (Voir Appendice.5 Fonctions à sens unique). Alice, qui veut s'identifier auprès de Bob, choisit un mot  $m$  et envoie

$m_0 = f^t(m)$  à Bob ( $t$  étant le nombre de fois dont on peut se servir de ce schéma avec un  $m$  choisi). Quand elle veut s'identifier, elle envoie, pour la  $i$ -ème connection (avec  $1 \leq i \leq t$ ),  $m_i = f^{t-i}(m)$ . Bob, pour s'assurer qu'il communique bien avec Alice, calcule  $f(m_i) = m_{i-1}$ . Si l'égalité est bien vérifiée, il mémorise  $m_i$  à la place de  $m_{i-1}$  et ainsi de suite.

L'avantage de ce protocole est qu'un attaquant qui écoute la ligne de communication de Bob et d'Alice, pourra bien obtenir des  $m_i$  mais vu que  $f$  est une fonction à sens unique, il ne pourra pas pouvoir calculer  $m_{i+1}$  donc il ne pourra pas tromper Bob en continuant l'identification à la place d'Alice, donc ce schéma est considéré sécurisé pour  $t$  connections. Si Alice veut s'identifier une  $t + 1$ ème fois, elle devra changer son mot  $m$ , sinon, elle recommencera à envoyer les mêmes valeurs.

### 1.3.2 Protocoles à deux passes

Les protocoles à deux passes consistent en l'identification à clé publique. Le secret d'Alice est sa clé privée. Elle démontre sa connaissance de ce secret en déchiffrant une valeur aléatoire que Bob lui envoie chiffrée.

Le protocole :

1. Bob choisit un message aléatoire qu'il crypte en utilisant la clé publique d'Alice et il le lui transmet,
2. Alice décrypte ce message avec sa clé privée et elle le transmet à Bob,
3. Bob vérifie si le message envoyé par Alice est bien le message qu'il avait choisi.

Si Oscar veut se faire passer pour Alice, il pourra pas décrypter le message vu qu'il ne connaît pas la clé de déchiffrement. Donc Alice, en pouvant retrouver le message original, prouve son identité à Bob.

### 1.3.3 Protocole Kerberos

Examinons maintenant un protocole, qui est assez connu et qui sert à la fois à l'identification et au chiffrement de messages.

Le protocole de Kerberos, est fondé sur la **distribution de clés** de Needham et Schroeder (pour plus de détails voir leur article [19]). Il fournit des clés de session qui permettent à deux clients de dialoguer ensemble en cryptant leurs messages. En cryptant des clés au lieu des mots de passe en texte clair, il empêche que des personnes non autorisées interceptent les mots de passe des utilisateurs. Son développement a commencé en 1978, a progressivement évolué et la version actuelle (version 5) date de 1994.

Alice qui désire s'identifier auprès de Bob, pour pouvoir communiquer avec lui, obtient une clé symétrique qu'elle partage avec Bob et de cette façon leur communication se fait en sécurité. L'utilité de ce protocole est que, Bob sera sûr qu'avec la clé de session qu'il utilise, il parle avec Alice et personne d'autre ne peut comprendre son message. Donc cela prouve l'identité d'Alice, le protocole a également une partie optionnelle qui permet à Alice d'être sûre de l'identité de Bob.

Ce protocole utilise, une autorité de confiance  $C$  qui génère la clé de session et chaque utilisateur  $U$  partage une clé symétrique  $K_U$  avec l'autorité pour lui prouver son identité.

Le protocole de Kerberos se déroule de la façon suivante :

- ***Demande de ticket*** : Alice s'adresse à l'autorité de confiance  $C$  pour obtenir une clé de session qu'elle utilisera pour communiquer avec Bob. Pour cela, elle envoie à  $C$ , sa propre identité, celle de Bob et un nombre aléatoire  $r$  pour éviter un rejeu éventuel.
- ***Obtention du ticket*** : L'autorité de confiance  $C$  choisit aléatoirement la clé de session  $K$  et chiffre avec la clé d'Alice ( $K_A$ ) le message  $(K, r, L, B)$  avec  $L$  représentant l'intervalle de temps pendant lequel le ticket sera valide.  $L$  est donc composé de la date et de l'heure courantes avec un délai d'expiration.

A part cela l'autorité  $C$  chiffre cette fois-ci avec la clé de Bob ( $K_B$ ) le message  $(K, L, A)$ , appelé *le ticket* (A et B sont respectivement les identités d'Alice et de Bob).

Ces deux messages chiffrés sont ensuite envoyés à Alice.

- ***Transmission du ticket*** : Alice qui reçoit ses deux messages chiffrés de la part de l'autorité de confiance  $C$ , commence par déchiffrer le premier message avec sa clé  $K_A$ , elle vérifie la valeur de  $r$  et s'assure que c'est identique à la valeur qu'elle avait choisi, puis elle prend la clé de session  $K$  générée par  $C$  et elle s'en sert pour chiffrer un message d'identification pour Bob. Ce dernier contient sa propre identité et un nouvel horodatage  $h$ . Elle envoie le deuxième message qu'elle a reçu de  $C$  et ce message d'identification qu'elle a chiffré avec  $K$  à Bob.
- ***Identification d'Alice*** : Bob, qui reçoit les deux messages d'Alice, commence par déchiffrer le message de l'autorité  $C$  afin de pouvoir obtenir la clé de session  $K$  et il l'utilise pour déchiffrer le message d'identification d'Alice. Il vérifie que l'identité d'Alice  $A$  contenue dans le ticket correspond bien à celle contenue dans le message d'identification, et que l'horodatage  $h$  et l'indication de sa propre horloge appartiennent à l'intervalle de validité  $L$  du ticket.  
Optionnellement Alice peut vouloir que Bob s'identifie à elle, si le cas est, Bob lui transmet la valeur  $h$  chiffré à l'aide de  $K$ .
- ***Identification optionnelle de Bob*** : Si Alice veut l'identification de Bob, alors elle déchiffre la valeur de  $h$  que Bob lui avait envoyé et elle vérifie si elle correspond bien à la valeur de  $h$  qu'elle possède.

Il y a aussi les protocoles sans divulgation d'information (zero-knowledge) qui consistent en une identification faite entre deux personnes sans que le prouveur donne une quelconque information sur son secret. Plus précisément Bob, à la fin du protocole, est convaincu qu'il communique bien avec Alice mais sans avoir appris quelque chose sur son secret, juste en voyant qu'elle le possède. Nous allons étudier ces protocoles en détail dans le chapitre suivant.

## Chapitre 2

### Introduction au Zero-Knowledge

Dans ce chapitre nous allons voir, dans un premier temps, une définition des protocoles d'identification *zero-knowledge* (sans divulgation d'information), nous allons en citer des exemples et nous allons conclure avec une comparaison de ces protocoles.

#### 2.1 Présentation

##### 2.1.1 Définition et Exemples

L'accès aux ordinateurs se fait en général avec un mot de passe qui est difficile à deviner et qui aide l'ordinateur à reconnaître la personne. Cette méthode, n'est pas vraiment sûre car nous nous connectons pendant une période avec le même mot de passe et il suffit que quelqu'un le voit pour qu'il puisse accéder à notre compte. Mais quand même elle peut être acceptée pour l'accès à un ordinateur qui est en face de nous.

De nos jours, les gens se connectent sur leur compte à distance et comme le réseau de communication peut être facilement écouté, cela rend cette méthode encore moins sécurisée. Donc il faut éviter l'utilisation du même mot de passe à chaque connection. Le schéma de Lamport que nous avons expliqué dans la partie 1.3.1, à la page 12, peut être une bonne méthode pour éviter ce problème mais ce n'est pas vraiment suffisant puisque si une connection échoue, on recommence avec le même mot de passe.

L'idéal serait que chaque personne possède un secret  $a$  propre à lui et qu'il puisse, à chaque connexion, convaincre la machine qu'il connaît ce secret en répondant aux

différentes questions posées par celle-ci, mais tout cela sans que la machine puisse avoir la moindre information sur  $a$ . Un tel protocole s'appelle un protocole sans apport d'information, ou zero-knowledge.

Au milieu des années 80, sont apparues des réalisations concrètes de ces protocoles et ceci a été aussi révolutionnaire que l'apparition de la cryptographie à clé publique.

Nous allons maintenant parler de la trois colorabilité des graphes et du protocole qu'on peut créer en se servant de ce problème et puis nous allons étudier en détail trois protocoles sans apport de connaissances, notamment le protocole de Fiat Shamir, le protocole de Schnorr et le protocole de Guillou Quisquater.

### 2.1.2 La trois colorabilité des graphes

#### Enoncé du problème

La question qui est posée dans le problème des trois couleurs est la suivante :

*Existe-il un coloriage des sommets d'un graphe  $G$  avec trois couleurs tel que deux sommets adjacents soient toujours coloriés par des couleurs différentes ?*

Dans un premier temps, essayons de voir en quoi consiste le trois coloriage des graphes. Un graphe est *trois colorié*, si tous ses sommets sont coloriés de telle façon que deux sommets voisins sont de couleurs différentes. Il faut aussi qu'on ne se soit servi que de trois couleurs en faisant cette coloriation.

Ce problème est NP-Complet (pour les définitions voir [13], pour plus de détails théoriques voir [18] pp. 181-218) de complexité  $O(e^n)$ ,  $n$  étant la taille du graphe donc le nombre de sommets : La classe NP (Non Déterministe Polynomial) est la classe des problèmes de décision pour lesquels la réponse "oui" peut être décidée par un algorithme non-déterministe en un temps polynomial par rapport à la taille de l'instance. Et les problèmes NP - Complet sont les problèmes NP les plus difficiles. Plus précisément, dire qu'un problème peut être résolu à l'aide d'un algorithme non-

déterministe revient à dire que si on a une solution donnée, c'est facile de vérifier en temps polynomial si elle répond au problème ; mais que le nombre de solutions à tester pour résoudre le problème sans avoir aucune solution, est exponentiel par rapport à la taille de l'instance.

### **Un protocole en forme de paradigme**

Alice qui veut prouver son identité à Bob, se sert d'un protocole créé par le problème du trois coloriage des graphes. Etudions pas à pas ce protocole :

- Pour assurer la sécurité de ce protocole, il faut utiliser des graphes de tailles assez grandes mais 3-colorier un graphe ayant beaucoup de sommets est assez difficile, donc Alice va préférer créer les sommets de trois différentes couleurs et mettre les arêtes aléatoirement en faisant attention à ne jamais avoir deux sommets voisins de même couleur.
- Une fois qu'elle a construit son graphe, elle le publie en cachant les couleurs des sommets. Donc la clé publique de ce protocole est le graphe décolorié et la clé privée est le coloriage. Comme Alice a publié le graphe, elle s'est engagée, donc elle ne peut plus changer les couleurs des sommets.
- Quand Bob reçoit le graphe il demande à Alice de lui ouvrir 2 sommets qu'il choisit.
- Alice, pour ne pas dévoiler son secret, donc le coloriage de son graphe, permute aléatoirement les couleurs (par exemple, elle colorie en noir tous les sommets rouges, en rouge tous les sommets bleus et en bleu tous les sommets noirs), et finalement elle ouvre les sommets demandés.
- Bob vérifie si vraiment ces deux sommets voisins sont de couleurs différentes.

Maintenant essayons de modéliser cela d'une façon plus théorique :

## Un protocole cryptographique théorique

Pour pouvoir construire un protocole sans transfert de connaissance, il faut réaliser un engagement sur un élément d'un ensemble  $\{1, 2, 3\}$  de taille trois. Donc Alice, en publiant son graphe trois colorié s'engage pour la couleur de tous les sommets, nous appelons ces trois couleurs avec les chiffres 1, 2 et 3.

Considérons une fonction bijective à sens unique  $f$  et supposons son ensemble de départ partitionné en trois ensembles  $X_0, X_1, X_2$ . Supposons qu'en connaissant  $f(x)$ , il soit impossible, en temps polynomial, d'obtenir une quelconque information significative sur lequel des  $X_i$  contient l'élément  $x$ .

Dans ce problème, donner  $f(x)$  à un vérificateur revient à lui donner  $i$  enfermé dans une boîte et lui donner  $x$  plus tard revient à lui ouvrir la boîte.

Avec ce problème, on peut créer une preuve sans transfert de connaissance. L'idée du protocole est de décomposer la preuve en plusieurs morceaux (un nombre polynomial) de telle manière que chaque morceau ne divulgue aucune information, mais que l'ensemble garantit la validité de la déclaration du prouveur (qu'il connaît une trois-coloration pour ce graphe). On obtient le protocole suivant [8] :

1. Soit  $\psi$  une 3-coloration de  $G$ , une fonction de l'ensemble des graphes  $G$  dans  $\{1, 2, 3\}$ . Alice choisit aléatoirement une permutation

$$\pi : \{1, 2, 3\} \rightarrow \{1, 2, 3\}.$$

Puis pour chaque sommet  $s$  Alice s'engage sur  $\pi \circ \psi (s)$ , c'est à dire qu'elle donne à Bob une quantité  $f(x_s)$  avec  $x_s \in X_{\pi \circ \psi_s}$ .

2. Bob choisit au hasard une arête  $\{s, t\}$  du graphe et la communique à Alice.
3. Alice communique le contenu des deux "boîtes" associées à  $s$  et  $t$ , donc  $x_s$  et  $x_t$ . Bob vérifie les valeurs  $f(x_s)$  et  $f(x_t)$  et s'assure que  $x_s$  et  $x_t$  sont bien dans

des  $X_i$  différents, ce qui montre que  $s$  et  $t$  sont bien coloriés par des couleurs différentes.

La sécurité de ce protocole est paramétré par la taille du graphe, donc le nombre de sommets. Plus on a de sommets, plus le protocole est sécurisé.

Le problème de trois colorabilité est NP-complet et il n'existe pas d'algorithme efficace pour décider si un graphe est 3-coloriable ou non pour des tailles suffisamment grandes.

Si l'attaquant Oscar, veut se présenter comme Alice et qu'il veut s'identifier à Bob sans avoir une 3-coloriation du graphe en question, alors quelle que soit la famille  $(x_S)$  que P choisit, il existe au moins deux sommets adjacents  $s$  et  $t$  de même couleur, donc les  $x_s$  et  $x_t$  correspondants appartiennent au même  $X_i$ . S'il y a  $m$  arêtes dans le graphe G, alors Bob se rend compte que c'est un attaquant avec probabilité  $\geq 1/m$ .

Donc au bout de  $k$  itérations du protocole, Bob est convaincu que G est 3-coloriable avec une probabilité  $1 - (1 - 1/m)^k$ .

Ce protocole est sans transfert de connaissance puisque même si Bob demande à Alice d'ouvrir tous les sommets deux à deux, comme Alice va faire une permutation aléatoire des couleurs à chaque fois, il pourra jamais avoir une information utile. Donc au bout de  $k$  itérations, tout ce dont Bob dispose est indistinguable de ce qu'il aurait obtenu tout seul en prenant  $k$  fois une arête aléatoire du graphe et en coloriant aléatoirement ses extrémités par des couleurs différentes. Bien sur que nous prenons ici l'hypothèse que Bob est incapable d'obtenir tout seul une quelconque information utile sur  $x$  à partir de  $f(x)$ .

Ce dernier point constitue une différence importante de ce protocole avec tous les autres protocoles sans transfert de connaissance. C'est la raison pour laquelle, ce protocole est appelé "sans apport de connaissance *au sens calculatoire*" (computa-

tional zero-knowledge). Les autres protocoles sont appelés "sans apport de connaissance *au sens parfait*" (perfect zero-knowledge) parce que la preuve qu'on fait pour prouver qu'ils sont des protocoles sans rapport de connaissance ne repose sur aucune hypothèse cryptographique comme l'existence d'une fonction à sens unique. Plus précisément les protocoles sans apport de connaissances au sens calculatoire sont moins sûrs que ceux sans apport de connaissance au sens parfait.

## 2.2 Protocoles "zero-knowledge"

Comme nous avons indiqué au paravant les protocoles zero-knowledge ont comme spécialité le fait qu'à la fin de l'identification qui se fait entre Alice et Bob, ni Bob, ni un éventuel pirate qui écoute la ligne de conversation, n'apprennent rien sur le secret d'Alice, ils apprennent juste, qu'elle connaît le secret.

### 2.2.1 Principe général d'un protocole "zero-knowledge"

Parmi les spécificités des protocoles zero-knowledge, citons :

- La mise en oeuvre des protocoles zero-knowledge nécessite la présence d'au moins deux parties, le prouveur (Alice) et le vérifieur (Bob).
- Ils permettent de réaliser des opérations comme l'identification et la signature...
- Alice ne peut pas tromper Bob en lui disant qu'elle connaît un secret qu'elle ne possède pas en réalité. En fait, les protocoles zero-knowledge ne permettent pas d'être vraiment sûr qu'Alice ne triche pas mais en les répétant plusieurs fois, il est possible que Bob réduise la probabilité qu'il soit trompé.
- Même en essayant de tricher, Bob ne peut rien apprendre de significative sur le secret d'Alice.
- Comme Bob ne peut rien apprendre de significatif sur le secret d'Alice, il ne peut non plus pas convaincre une troisième personne qu'il est Alice même s'il a enregistré l'échange.

(Pour plus de détails et des exemples voir [2]).

Etudions maintenant plus en détail quelques protocoles d'identifications à divulgation nulle de connaissance.

### 2.2.2 Protocole de Fiat Shamir

Le protocole de Fiat-Shamir (voir [14]) est un protocole zero-knowledge qui est basé sur **la complexité du calcul d'une racine carrée dans  $\mathbb{Z}/n\mathbb{Z}$**  où  $n$  est le produit de deux grands nombres premiers  $p$  et  $q$ , donc  $n$  est un entier R.S.A.. Dans ce protocole le secret détenu par le prouveur Alice est la racine carrée  $a \pmod n$  d'un entier public  $A \in [0, n - 1]$ . Alice doit prouver au vérifieur Bob que  $A$  est un carrée dans  $\mathbb{Z}_n^*$ . Elle s'identifie auprès de Bob à l'aide de trois échanges :

Tableau (2.1) Protocole de Fiat Shamir

<p>1. <b>L'engagement</b> : Alice choisit un nombre aléatoire <math>k</math> dans <math>[1, n - 1]</math>, puis elle calcule <math>K = k^2 \pmod n</math> et le transmet à Bob.</p> <p>2. <b>Le défi</b> : Bob choisit un bit aléatoire <math>r = 0</math> ou <math>1</math> et le communique à Alice.</p> <p>3. <b>La réponse</b> : Alice calcule <math>y = k \times a^r \pmod n</math> et le transmet à Bob.</p> <p>Bob vérifie que <math>y^2 \equiv K \times A^r \pmod n</math> (donc il essaye de voir si <math>A</math> est résidu quadratique ou pas).</p>
--

Ce protocole est **consistant** (completeness property), cela signifie que la connaissance du secret  $a$  permet à Alice de donner la bonne réponse à Bob quoi qu'il choisisse comme défi.

Ce protocole est **significatif** (soundness), cela signifie que, pour pouvoir réussir son identification avec probabilité acceptable (donc dans ce cas supérieure à  $1/2$ ) Alice doit connaître le secret  $a$ . Comme dans tous les protocoles, dans ce protocole aussi l'ordre des interactions est très important. Alice ne peut pas prévoir la valeur du défi que Bob va choisir avant de transmettre  $K$  à Bob, donc elle ne peut pas

choisir  $K$  d'après le défi, d'ailleurs c'est pourquoi on appelle  $K$  l'engagement. Alice s'engage et elle prouve qu'elle ne changera pas le  $k$  qu'elle a choisit en envoyant son carrée à Bob. Alice, pour ne pas échouer dans son identification, doit être capable de fournir à Bob les deux réponses possibles :  $k$  et  $k \times a$  (mais bien sûr qu'elle n'en fournira qu'une seule à la fois pour ne pas dévoiler son secret) et pour cela, elle doit connaître le secret  $a$ .

Concrétisons tout cela sur un exemple précis : (normalement pour la sécurité du protocole, il faut travailler avec des nombres très grands mais là, pour faciliter la compréhension du protocole, nous allons utiliser des nombres de 8 bits que nous allons écrire en base 10).

- Nous avons tout d'abord besoin de deux nombres premiers  $p$  et  $q$  qui seront privés :

$$p = 173,$$

$$q = 107.$$

- La multiplication de ces deux nombres  $n$  est publique :

$$n = p \times q = 173 \times 107 = 18511.$$

- Alice choisit aléatoirement un nombre  $A \in [0, n - 1]$  qui est public et son secret c'est la racine carrée  $a \pmod n$  de ce nombre.

$$a = 2052 \text{ et } A = a^2 \pmod n = 8707.$$

- Le protocole :

1. Alice choisit un entier aléatoire  $k \in [0, n - 1]$  (l'engagement), et elle calcule  $K = k^2 \pmod n$ , puis elle transmet  $K$  à Bob :

$$k = 3628,$$

$$K = k^2 \pmod n = 1063.$$

2. Bob choisit un booléen  $r \in [0, 1]$ , (le défi) et le transmet à Alice :

$$r = 0.$$

3. Alice calcule  $y = k \times a^r \pmod n$ , (la réponse) et le transmet à Bob :

$$y = k \times a^r \pmod n = 3628.$$

Et finalement Bob vérifie si  $y^2 = K \times A^r \pmod n$

$$K \times A^r = 3628$$

L'égalité est vérifiée!!

Au bout de  $t$  applications du protocole, Bob est convaincu que  $A$  est résidu quadratique avec probabilité  $1 - 1/2^t$ . Il s'agit bien d'un protocole sans transfert de connaissance car toutes les données que Bob a pu obtenir pendant l'interaction, consistent simplement en une liste d'entiers aléatoires et leurs carrés modulo  $n$ . Grâce à ce protocole répété plusieurs fois, Bob est certes convaincu que  $A$  est un carré modulo  $n$ , mais il est surtout convaincu qu'Alice détient une racine de  $A$  (donc  $a$ , qui est son secret).

Un attaquant Oscar, souhaitant se faire passer pour Alice mais ne connaissant pas  $a$  peut tenter de choisir un  $k$  aléatoire et calculer le carré, donc  $K$ , et l'envoyer à Bob. Si le défi choisi par Bob est 0, il pourra fournir à Bob sans problème la valeur de  $y = k$ , mais dans le cas où  $r$  sera 1, il sera bloqué.

Ou sinon, il peut choisir  $k$  au hasard et transmettre  $K = k^2/A$  à Bob. Dans ce cas, si le défi  $r = 1$ , il pourra fournir à Bob la racine carrée  $k$  de  $K \times A$ , mais cette fois-ci, il sera bloqué pour  $r = 0$ . Donc au mieux, Oscar peut duper Bob au plus une fois sur deux.

Bob, peut rendre la probabilité de se faire tromper arbitrairement faible, en répétant cette identification plusieurs fois.

La sécurité de ce protocole est basé sur la difficulté de la factorisation de  $n$  et sur la difficulté de calculer les racines carrés de  $A$  dans  $\mathbb{Z}/n\mathbb{Z}$ .

### 2.2.3 Protocole de Schnorr

Le protocole de Schnorr (voir [16]) est un protocole basé sur **le problème du logarithme discret**. Tout d'abord voyons en quoi consiste ce problème. Il faut considérer un groupe qui est une structure mathématique telle que :

1. C'est un ensemble, muni d'une opération et d'un élément neutre ( $1_G$ ),
2. Il est associative :  $g_1 * (g_2 * g_3) = (g_1 * g_2) * g_3$
3.  $\forall g \in G$ , il existe  $g^{-1} \in G$  tel que  $g * g^{-1} = g^{-1} * g = 1_G$ .

Globalement un groupe est la structure "minimale" sur laquelle on peut faire des opérations.

**Définition :** Le problème du logarithme discret dans un groupe  $G$  est : étant donné  $g \in G$  et  $h = g^x \in G$  où  $x$  est inconnu, retrouver  $x$ .

Ce problème est la source essentielle du protocole de Schnorr qui marche de la façon suivante :

Une autorité publie deux grands nombres premiers  $p$  et  $q$  tels que,  $q$  est un diviseur de  $p - 1$  ainsi qu'un entier  $g$  d'ordre  $q$  modulo  $p$  ( $g^q = 1 \pmod p$ ). Ces données sont choisies de telle sorte que le logarithme discret dans  $(\mathbb{Z}/p\mathbb{Z})^*$  soit "difficile".

Le prouveur Alice choisit comme secret, un entier  $a \in [0, q - 1]$ , c'est sa clé privée, puis elle calcule  $A = g^{-a} \pmod p$ , c'est sa clé publique. Alice qui veut s'identifier auprès de Bob avec son couple de clés privée - publique  $(a, A)$ , doit prouver sa connaissance du secret  $a$ , et pour cela elle fait ces trois échanges :

Tableau (2.2) Protocole de Schnorr

<p>1. <b>L'engagement</b> : Alice choisit un nombre aléatoire <math>k</math> dans <math>[0, q - 1]</math>, puis elle calcule <math>K = g^k \pmod p</math> et le transmet à Bob.</p> <p>2. <b>Le défi</b> : Bob choisit un nombre aléatoire <math>r</math> dans <math>[0, q - 1]</math> et le communique à Alice.</p> <p>3. <b>La réponse</b> : Alice calcule <math>y = k + a \times r \pmod q</math> et le transmet à Bob.</p> <p>Bob vérifie que <math>g^y \times A^r \equiv K \pmod p</math>. Finalement, Bob accepte la preuve d'Alice si et seulement si Alice répond correctement à <math>t</math> interactions successives suivant la sécurité.</p>
---

Concrétisons tout cela sur un exemple précis : (normalement pour la sécurité du protocole, il faut travailler avec des nombres très grands mais là, pour faciliter la compréhension du protocole, nous allons utiliser des nombres de 8 bits que nous allons écrire en base 10).

- Nous avons tout d'abord besoin de deux nombres premiers  $p$  et  $q$  tels que  $q$  divise  $p - 1$  ces nombres seront privés :

$$p = 2111,$$

$$q = 211.$$

- Puis nous avons besoin d'un entier  $g$  d'ordre  $q$  modulo  $p$  :

$$g = 228,$$

$$228^{211} = 1 \pmod{2111}.$$

- Alice choisit aléatoirement un nombre  $a \in [0, q - 1]$  qui est sa clé privée et elle calcule  $A = g^{-a} \pmod{p}$  et ceci est sa clé publique.

$$a = 151 \text{ et } A = g^{-a} \pmod{p} = 1507.$$

- Le protocole :

1. Alice choisit un entier aléatoire  $k \in [0, q - 1]$  (l'engagement), et elle calcule  $K = g^k \pmod{p}$ , puis elle transmet  $K$  à Bob :

$$k = 95,$$

$$K = g^k \pmod{p} = 947.$$

2. Bob choisit un entier  $r \in [0, q - 1]$ , (le défi) et le transmet à Alice :

$$r = 121$$

3. Alice calcule  $y = k + a \times r \pmod q$ , (la réponse) et le transmet à Bob :

$$y = k + a \times r \pmod q = 9.$$

Et finalement Bob vérifie si  $g^y \times A^r \equiv K \pmod p$

$$g^y \times A^r = 947 = K$$

L'égalité est vérifiée!!

Ce protocole est **consistant**, Alice qui connaît  $a$ , sait répondre à la question de Bob, pour n'importe quelle valeur de  $r$  puisque la vérification faite par Bob est :  $g^y A^r \stackrel{?}{=} K \pmod p$ , nous supposons qu'Alice ne triche pas donc elle connaît  $a$ , elle sait calculer  $y = k + ar$ . Alors comme Alice a envoyé  $y$  à Bob, la vérification que Bob fait est la suivante :

$$g^y A^r \stackrel{?}{=} K \pmod p,$$

$g^y = K A^{-r}$ , et nous avons :  $y = k + ar$ , on remplace  $y$  par sa valeur,

$$g^y = g^{k+ar} = g^k (g^a)^r \text{ et donc}$$

$g^y = K A^{-r} \Rightarrow g^y A^r = K \pmod p$ , donc si on connaît  $a$  c'est sûr qu'on donne la bonne réponse.

Ce protocole est **significatif** : quelqu'un qui sait répondre d'une façon correcte à deux questions différentes connaît le secret  $a$ .

Supposons que Bob ait posé deux questions distinctes  $(r_1)$  et  $(r_2)$ , il recevra alors deux réponses distinctes  $(y_1)$  et  $(y_2)$ . Et les vérifications qu'il fera seront de la forme :

$$g^{y_1} = K \times A^{r_1} \text{ et } g^{y_2} = K \times A^{r_2}.$$

Nous savons que  $r_1 \neq r_2$ , donc nous pouvons retrouver  $a$  de la façon suivante :

Divisons  $g^{y_1}$  par  $g^{y_2}$ . Cela nous donne :

$$g^{y_1 - y_2} = g^{-a(r_1 - r_2)}.$$

Donc,

$$-a \leftarrow \frac{y_1 - y_2}{r_1 - r_2},$$

$$\frac{(r_1 a + k) - (r_2 a + k)}{(r_1 - r_2)} = \frac{(r_1 - r_2)a}{(r_1 - r_2)} = -a.$$

#### 2.2.4 Protocole de Guillou Quisquater

Le protocole Guillou-Quisquater (voir [15]) utilise **une exponentiation R.S.A.** et il est, de nos jours, utilisé plutôt dans les cartes à puces. Une autorité de confiance publie un nombre  $n$  qui est obtenu en multipliant deux nombres premiers aléatoires :  $p$  et  $q$  grands. Avec ce nombre  $n$ , l'autorité publie un nombre  $E$  premier avec  $\varphi(n)$ , et c'est le exposant de chiffrement. Ces données peuvent être utilisées par plusieurs couples qui veulent communiquer entre eux. Donc  $n$  et  $E$  sont les clés publiques du protocole.

Par contre, l'autorité garde privé les valeurs  $p$ ,  $q$  et l'exposant de déchiffrement  $e = E^{(-1)} \pmod{\varphi(n)}$  (avec  $\varphi(n) = (p - 1) \times (q - 1)$ ). Ce nombre  $e$  est premier. Donc ces données constituent la clé privée du protocole.

Alice, qui veut prouver son identité à Bob, a besoin d'un secret  $a$  qui est privé et de  $A = a^{(-E)} \pmod{n}$  qui est public. La valeur de  $A$  est vérifiée et certifiée par l'autorité par un procédé de signature quelconque.

Pour s'identifier à Bob, Alice utilise les trois échanges qui suivent :

Tableau (2.3) Protocole de Guillou Quisquater

<p>1. <b>L'engagement</b> : Alice choisit un nombre aléatoire <math>k \in \mathbb{Z}_n^*</math>, puis elle calcule <math>K = k^E \pmod n</math> et le transmet à Bob avec sa clé publique certifiée par l'autorité.</p> <p>2. <b>Le défi</b> : Bob, après avoir vérifié la valeur de A auprès de l'autorité, choisit un nombre aléatoire <math>r \in [0, E - 1]</math> et le communique à Alice.</p> <p>3. <b>La réponse</b> : Alice calcule <math>y = k \times a^r \pmod n</math> et le transmet à Bob.</p> <p>Bob vérifie que <math>y^E \times A^r \equiv K \pmod n</math>.</p>
---

Concrétisons cela sur un exemple précis : (normalement pour la sécurité du protocole, il faut travailler avec des nombres très grands mais là, pour faciliter la compréhension du protocole, nous allons utiliser des nombres de 8 bits que nous allons écrire en base 10).

- Nous avons tout d'abord besoin de deux nombres premiers  $p$  et  $q$  tels que  $q$  divise  $p - 1$  ces nombres seront privés :

$$p = 61,$$

$$q = 229.$$

- La multiplication de ces deux nombres  $n$  est public :

$$n = p \times q = 61 \times 229 = 13969.$$

- Puis nous avons besoin de  $\phi$  qui est  $(p - 1) \times (q - 1)$  :

$$\phi(n) = (p - 1) \times (q - 1) = 13680.$$

Nous générons aléatoirement l'exposant de chiffrement  $E$ , il faut juste qu'il soit inversible pour qu'on puisse calculer  $e = E^{(-1)} \pmod{\phi(n)}$ .

$$E = 10967.$$

- Alice choisit aléatoirement un nombre  $a \in [0, n - 1]$  qui est sa clé privée et elle calcule  $A = a^{-E} \pmod{n}$  et ceci est sa clé publique.

$$a = 9397 \text{ et } A = g^{-a} \pmod{p} = 5395.$$

- Le protocole :

1. Alice choisit un entier aléatoire  $k \in [0, n - 1]$  (l'engagement), et elle calcule  $K = k^E \pmod{n}$ , puis elle transmet  $K$  à Bob :

$$k = 1460,$$

$$K = k^E \pmod{n} = 9288.$$

2. Bob choisit un entier  $r \in [0, E - 1]$ , (le défi) et le transmet à Alice :

$$r = 4241.$$

3. Alice calcule  $y = k \times a^r \pmod{n}$ , (la réponse) et le transmet à Bob :

$$y = k \times a^r \pmod{n} = 10358.$$

Et finalement Bob vérifie si  $y^E \times A^r \equiv K \pmod{n}$

$$y^E \times A^r = 9288 = K.$$

L'égalité est vérifiée!!

Ce protocole est **consistant**, donc Alice, qui connaît  $a$ , saura répondre correctement à toute question posée par Bob.

Ce protocole est **significatif** : La probabilité qu'Oscar, qui veut s'identifier auprès de Bob en disant qu'il est Alice, donc ne connaissant pas le secret  $a$ , est de  $1/e$ .

Jusqu'ici nous ne nous sommes jamais servi de  $e$  qui est l'exposant de déchiffrement qui est gardé par l'autorité de composant. Alors on peut se demander pourquoi on a besoin de ce nombre. En effet, ce nombre nous sert pour montrer que ce protocole est significatif :

Supposons qu'un prouveur ait donné deux fois de suite une bonne réponse à deux questions distinctes  $r_1$  et  $r_2$  posées par Bob. Soient ces réponses  $y_1$  et  $y_2$ . Alors :

$$A^{r_1}y_1^e \equiv A^{r_2}y_2^e \pmod{n}.$$

Donc :

$$A^{r_1-r_2} \equiv (y_2y_1^{-1})^e \pmod{n}.$$

Soit  $g$  l'inverse de  $r_1 - r_2 \pmod{e}$ , donc :

$$(r_1 - r_2)g = ue + 1 \text{ avec } u \in \mathbb{Z}.$$

Cet inverse existe puisque  $e$  est un nombre premier et  $|r_1 - r_2| < e$ . Donc :

$$A \equiv A^{(r_1-r_2)g}A^{-ue} \equiv (y_2y_1^{-1})^{eg}A^{-ue} \equiv ((y_2y_1^{-1})^g(A^u)^{-1})^e \pmod{n}.$$

Donc le prouveur peut trouver la  $e$ -ième racine de  $A \pmod{n}$ .

Le protocole de Fiat-Shamir, le protocole de Schnorr et le protocole de Guillou-Quisquater, répondent tous au problème de non divulgation d'information en se basant sur différentes méthodes et garantissant leur sécurité de différentes façons. Le protocole de Fiat Shamir se base sur la difficulté de factorisation de  $n$ , le protocole de Schnorr sur le problème du logarithme discret et le protocole de Guillou-Quisquater utilise la méthode d'exponentiation de R.S.A.

## Chapitre 3

### Application

Dans ce chapitre, nous allons tout d'abord présenter le langage MAGMA qui nous a servi pour implémenter les protocoles étudiés ci-dessus, et puis nous allons voir comment nous avons fait ces implémentations en question, en s'appuyant sur les points importants et finalement nous allons donner les résultats de quelques tests que nous avons réalisés.

#### 3.1 Présentation du logiciel de calcul formel Magma

Magma (voir le site Web pour plus de détails et la documentation [20]) est un logiciel de calcul formel pour l'algèbre, la théorie des nombres, la géométrie, l'analyse combinatoire, etc. . . Son développement a commencé en 1973 à Sydney par J.J. Cannon.

Dans le calcul numérique, les variables et les paramètres sont remplacés par des valeurs numériques mais, dans le calcul formel, au lieu de garder la valeur exacte d'un variable, on garde un certain nombre de propriétés de simplifications et un programme de calcul d'un nombre arbitraire de décimales. Par exemple dans le calcul numérique pour  $\pi$  on garde directement la valeur 3,1415927... mais dans le calcul formel on pourrait garder des propriétés comme  $\sin \pi = 0$ ,  $\cos \pi = 1$ , *etc* . . . et le remplacement par une valeur approchée ne se fait que si un utilisateur le demande (voir [3]).

Depuis quelques années, ces logiciels sont de plus en plus utilisés par les ingénieurs et les chercheurs car ils facilitent beaucoup les calculs. . .

La philisophie de Magma (voir [4]) :

- C'est un langage structuré ou typé,
- Il a une conception et une syntaxe homogène,
- Son utilisation est facile car les notations sont très claires :

**Exemple :** Pour dire :  $B = \{p : p \in A | p \text{ est premier}\}$  où  $A = \{1, \dots, 100\}$ ,

On dit :  $B := \{p : in A | IsPrime(p)\}$  where  $A := \{1..100\}$ ;

- Il est très performant : il semble qu'il soit le plus rapide des systèmes généralistes,
- Il tend vers l'exhaustivité : il a beaucoup de fonctionnalités,

Magma fournit un environnement mathématiquement rigoureux qui souligne le calcul structural. Il contient des réalisations de plusieurs classes concrètes importantes de la structure dans cinq branches fondamentales de l'algèbre, à savoir théorie de groupe, théorie d'anneau, théorie de champ, théorie de module et la théorie des algèbres.

## 3.2 Implémentation :

Pour ce rapport, nous avons implémenté en nous servant du langage *MAGMA* quelques protocoles, pour pouvoir les comparer et pour pouvoir faire des tests.

### 3.2.1 Protocole Fiat Shamir

```
//Generation de p et q
print "Pour ce protocole nous avons besoin de deux nombres
aleatoires p et q";
p :=RandomPrime(128);
q :=RandomPrime(128);

//n = p * q
printf "La multiplication de ces 2 nombres n est public";
printf "Mais comme p et q sont tres grands, c'est tres difficile de
factoriser n.";
n := p*q;
```

```

Zn := ResidueClassRing(n);

// Le secret d'Alice 'a'
printf "Alice choisit un nombre plus petit que n, ce nombre
'A' est public";
printf "Le secret c'est 'a' qui est la racine carree de A mod(n)";
a := Random(Zn);

//A = le carre de a (mod n)
A := Zn!a^2;

//protocole
printf "Ce protocole est compose de 3 echanges entre Alice et Bob";

//1er echange
printf "Alice choisit un entier aleatoire k (l'engagement),
dans l'intervalle [0,n-1] et calcule K = k^2 (mod n)";
printf "Elle transmet K a Bob";

k := Random(Zn);
K := Zn!k^2;

//2e echange
printf "Bob choisit un booleen r (le defi) au hasard (0 ou 1)
et le transmet a Alice";
r := Random(0,1);

//3e echange
printf "Alice calcule y = k * a^r (mod n) (la reponse) et
le transmet a Bob";

```

```

y := Zn!(k*a^r);

//Bob calcule
printf "Bob calcule K * A^r (mod n):";
y2 := Zn!(K*A^r);

//Verification
printf "Et finalement il verifie que y^2 = ? K * A^r (mod n)";
if y^2 eq y2
then printf "Test reussi";
else printf "Test non reussi";
end if;

```

***Remarques :***

- $p := \text{RandomPrime}(128)$  : Génère un nombre premier de 128 bits,
- $Z_n := \text{ResidueClassRing}(n)$  : Crée la classe des nombres modulo  $n$ ,
- $a := \text{Random}(Z_n)$  : Génère  $a$  de la classe des nombres modulo  $n$  donc  $a \in \mathbb{Z}/n\mathbb{Z}$ ,
- $A := \text{Zn}!a^2$  : Prend la racine carrée de  $a$  et la réduit modulo  $n$  donc  $A \in \mathbb{Z}/n\mathbb{Z}$ ,

Comme tous les nombres (sauf  $p$  et  $q$ ) sont dans  $\mathbb{Z}/n\mathbb{Z}$ , le temps des calculs est plus court, si on travaillait dans  $\mathbb{Z}$  et qu'on réduisait les nombres modulo  $n$  après, on devrait travailler avec des nombres énormes et cela ralentirait notre protocole.

### 3.2.2 Protocole de Schnorr

```

//structures
Z := IntegerRing();

//Generation de p et q premiers tels que q divise p-1

```

```

//Generation de q
q:=RandomPrime(128);

b:=Random(1,q);

//Cette fonction prend comme parametre le q qui a ete genere
//et il trouve un p (premier) tel que q divise p-1

trouvons_p := function(q)
for n in [2..b] do
if IsPrime(n*q+1) then
p := n*q+1;
return p;
end if;
end for;
end function;

//generation de p avec la fonction trouvons_p
p := trouvons_p(q);

//Declaration des structures Zp et Zq
Zp := ResidueClassRing(p);
Zq := ResidueClassRing(q);

//generation de g tel que  $g^q = 1 \pmod p$ 
h := PrimitiveElement(Zp);
d := (p-1) div q;
g := h^(d);

//Le secret de Alice => a dans [0,q-1]
a := Random(0,q-1);

```

```

//La donnee publique => A = g^(-a)mod p
A := Zp!(g^(-a));

//Protocole
//1er echange
//Alice choisit un engagement aleatoire k dans [0,q-1]
//puis elle calcule K = g^(k) mod p elle le transmet a Bob
k := Random(0,q-1);

K := Zp!(g^(k));

//2e echange
//Bob choisit un defi r dans [0,q-1] et le transmet a Alice
r := Random(0,q-1);

//3e echange
//Alice calcule la reponse y = k + ar mod q et le transmet a Bob
y := Zq!(k + a*r);

//Verification
//Bob verifie que g^(y)*A^(r) = K mod p
if g^(Z!y)*A^(r) eq Zp!K
then printf "\nTest reussi";
else printf "Test non reussi";
end if;

```

**Remarques :**

- `trouvons_p := fonction(q) :` Cette fonction retourne un nombre premier, qui est divisible par  $q - 1$ ,
- `IsPrime(n*q+1) :` Regarde si  $n \times q + 1$  est premier ou pas, si oui il envoie ceci comme  $p$ .

- $h := \text{PrimitiveElement}(\mathbb{Z}_p)$  ; : On génère  $h$ , un élément primitif de  $\mathbb{Z}/p\mathbb{Z}$
- $d := (p-1) \text{ div } q$  ;  $g := h^d$  ; : On prend la puissance  $(p-1)/q$  de ce nombre  $h$  pour obtenir un générateur  $g$  tel que  $g^q \equiv 1 \pmod{p}$ ,

### 3.2.3 Protocole de Guillou-Quisquater

```
// Les structures independantes
Z := IntegerRing();

//fonction qui genere des nombres inversibles
geninv := function(n)
answer := false;
while not answer do
E := Random(1,n-1);
if Gcd(E,n) eq 1 then
answer := true;
end if;
end while;
return E;
end function;

//L'autorite de Confiance genere p et q premiers
print "\n\nPour ce protocole l'Autorite de Confiance genere deux nombres
aleatoires p et q (premiers)";
p :=RandomPrime(128);
q :=RandomPrime(128);

//n = p * q
printf "\n\nLa multiplication de ces 2 nombres n est public";
printf "\n\nMais comme p et q sont tres grands, c'est tres
difficile de factoriser n\n";
```

```

n := p*q;

//structures dependantes
Zn := ResidueClassRing(n);

//on calcule phi
phi := (p-1)*(q-1);

//On genere E avec la fonction geninv
// (comme ca on est sur il est inversible)
E := geninv(phi);

//e l'inverse de E mod phi
e:=(E^(-1));

//Alice possede un secret a
a := geninv(n);

//et sa contrepartie A=a^(-E)mod n
A:= (Zn!a)^(-E);

//Protocole
//1er echange
//engagement : Alice choisit k dans Zn
k:=Random(Zn);

//elle envoie K=k^(E) mod n a Bob
K:=Zn!(k^(E));

//2e echange
//Bob choisit un defi r dans [0,E-1] et l'envoie a Alice

```

```

r:=Random(0,Z!(E-1));

//3e echange
//Alice calcule y = k*a^r mod n et l'envoie a Bob
y:=(Zn!k)*(Zn!a)^(r);

//Verification
//y^E * A^r =? K mod n
if (Zn!y)^(E) * (Zn!A)^(r) eq Zn!K
then printf "\nTest reussi";
else printf "Test non reussi";
end if;

```

**Remarques :**

- `geninv := fonction(n)` : Cette fonction trouve un nombre modulo  $n$  inversible, on génère un entier modulo  $n$  aléatoirement puis on regarde si le pgcd de ce nombre avec  $n$  est 1 ou pas. Si c'est 1 cela veut dire que le nombre généré est inversible. La génération aléatoire dans  $\mathbb{Z}/n\mathbb{Z}$  sans connaître la factorisation de  $\varphi(n)$  est un problème délicat.
- `A := (Zn!a)^(-E)` : Ici nous calculons  $a^{-E} \pmod n$  mais en fait au lieu de faire tout d'abord le calcul et de réduire la réponse modulo  $n$ , il vaut mieux réduire  $a \pmod n$  et puis de prendre la puissance  $(-E)$  de cette façon nous restons toujours dans la classe  $\mathbb{Z}/n\mathbb{Z}$  et donc nous ne rencontrons pas des nombres très grands. Si on ne fait pas ça en faisant ce calcul nous aurons un nombre de 128 bits à la puissance un nombre de 128 bits...

### 3.3 Tests et Résultats

Après avoir implémenté les protocoles, nous avons fait quelques tests sur ceux-ci. En fait savoir factoriser  $n$  revient à savoir casser plusieurs systèmes de cryptographie (Voir Cryptosystème R.S.A. Section 1.2, page 6). Donc en premier nous avons essayé

de voir le temps qu'il faut pour pouvoir factoriser  $n$  pour des couples  $(p, q)$  de tailles différentes. Nous avons fait ces tests avec un ordinateur qui a un processeur Pentium Centrino, 1,4 GHz.

Le deuxième test que nous avons fait était pour voir le temps qu'il faut pour trouver les racines carrées d'un nombre  $A = a^2 \pmod n$ , en fait c'est le  $a$  qui est le secret d'Alice dans le protocole de Fiat-Shamir. Celui qui sait retrouver  $a$  à partir de  $A$ , peut très facilement se faire passer pour Alice vu qu'il possède son secret.

Voilà les résultats obtenus (les temps exprimés sont en secondes) :

Tableau (3.1) Résultats des tests

Nombre de Bits des premiers	Factorisation de $n$	Trouver les racines carrées de $A$
32	0.11	0.11
45	0.28	0.26
56	0.33	0.32
64	1.041	1.041
75	3.004	2.974
85	6.219	6.399
95	28.14	28.331
100	102.367	105.001
110	133.742	132.37
128	2559.991 (42.7 min)	2512.523 (41.9 min)

Nous avons arrêté nos tests à 128 bits comme le temps augmente énormément pour des nombres plus grands. En fait le problème de la factorisation est exponentielle donc l'augmentation se voit très clairement...

Ce que nous avons également pu constater, est que le temps que l'ordinateur met pour factoriser  $n$  et pour trouver les racines carrées de  $A$  sont quasiment identiques (ce qui confirme les résultats obtenus de manière théorique).

## Conclusion

L'informatique et l'Internet ont tellement évolué que dans le monde contemporain, ils sont devenus une partie indispensable de notre vie. Au lieu d'utiliser le téléphone ou la poste, nous communiquons avec nos amis par Internet ; au lieu d'aller faire nos courses nous mêmes, nous nous connectons sur les sites des super-marchés, nous faisons tous nos achats en ligne, et nous faisons encore beaucoup de choses à distance. Certes, cela nous facilite beaucoup la vie mais en même temps toutes ces interactions peuvent être écoutées par des pirates et ces derniers peuvent apprendre des choses qui peuvent leur servir à s'identifier en notre nom et de faire des choses à notre place.

Comme la sécurité de ces systèmes à distance est très importante, il a fallu trouver des méthodes pour la garantir et pendant cette recherche sont nés la cryptographie et les protocoles d'identification : la cryptographie, pour chiffrer les messages confidentiels qui sont envoyés à distance, afin de les protéger des pirates éventuels ; et les protocoles d'identification pour garantir que celle qui veut se connecter en un système est bien la personne qu'elle prétend être.

Ces deux domaines, d'une importance primordiale dans notre vie, ont été et continuent à être des sujets de recherche pour les mathématiciens et les informaticiens. Dans ce projet, nous avons vu l'importance de ces concepts en montrant l'état de l'art de ces recherches et en étudiant les différents protocoles d'identification. Par la suite, nous avons étudié sur quoi reposent la sécurité de ces protocoles, en les implémentant à l'aide du langage Magma. Ce projet m'a donc permis de découvrir une des nombreuses applications de la cryptographie. J'ai ainsi pu apprendre, comprendre et mettre en pratique, ce que j'ai lu à travers des livres et Internet, et ce que mon responsable de projet, Monsieur Riboulet, m'a enseigné.

Ce domaine de la cryptographie, que j'appréciais déjà avant de réaliser ce projet, me plaît encore plus aujourd'hui. J'envisage de plus en plus sérieusement me spécialiser dans ce domaine.

## Bibliographie

- [1] "Introduction à la Cryptographie",  
<http://www.madchat.org/crypto/I2.cryptologie.pdf>.
- [2] "Protocoles à Divulgateion Nulle",  
[http://perso.crans.org/martin/Doc/Divers/TER/HTML/TER-compte\\_rendu06.htm](http://perso.crans.org/martin/Doc/Divers/TER/HTML/TER-compte_rendu06.htm).
- [3] Lazard "Calcul Formel : Tendances et progrès récents",  
<http://www-calfor.lip6.fr/~dl/tsi.pdf>.
- [4] Pecquet "Introduction au système de calcul formel Magma",  
[http://www.math.univ-montp2.fr/FODESIT/SemJuin2004/Pecquet/2004\\_MontpellierMathSem.pdf](http://www.math.univ-montp2.fr/FODESIT/SemJuin2004/Pecquet/2004_MontpellierMathSem.pdf).
- [5] Jackson "Linux Shadow Password HOWTO" :  
<http://www.tldp.org/HOWTO/Shadow-Password-HOWTO-2.html>.
- [6] "Le chiffre de César",  
<http://zmaster007.free.fr/cryptocesar.htm>.
- [7] Müller "Le chiffre de Vigenère",  
<http://www.apprendre-en-ligne.net/crypto/vigenere/>.
- [8] Zémor "Cours de Cryptographie" (2000).
- [9] "Le chiffrement AES",  
[http://www.egs-howto.com/fr/securite/cryptographie\\_aes.php](http://www.egs-howto.com/fr/securite/cryptographie_aes.php).
- [10] Diffie, Hellman, "New directions in cryptography, IEEE Transactions on Information Theory" (1976).
- [11] Rivest, Shamir, Adleman, "A method for obtaining digital signatures and public-key cryptosystems", (1978).

- [12] Mascle "Identité de Bézout",  
<http://membres.lycos.fr/masclejp/spe.html>.
- [13] Wikipédia "Théorie de la Complexité",  
[http://fr.wikipedia.org/wiki/Th%C3%A9orie\\_de\\_la\\_complexit%C3%A9](http://fr.wikipedia.org/wiki/Th%C3%A9orie_de_la_complexit%C3%A9).
- [14] Fiat, Shamir "How to prove yourself : Practical solutions of identification and signature problems" Springer-Verlag, (1987).
- [15] Quisquater, Guillou "A practical zero-knowledge protocol fitted to security microprocessor minimising both transmission and memory" Springer-Verlag (1988).
- [16] Schnorr, "Efficient identification and signatures for smart-cards", Springer-Verlag, (1990).
- [17] Izmitli, Kayis, Kocaer "La Cryptographie" (2003).
- [18] Papadimitriou, "Computational Complexity" (1994).
- [19] Needham, Schroeder, "Using Encryption for Authentication in Large Networks Computers", (Dec. 1978)
- [20] "The Magma Computational Algebra System" ,  
<http://magma.maths.usyd.edu.au/magma/>.

## Appendice A

### Algorithme d'Euclide

– **Théorème** : [17] Soit  $c = \text{pgcd}(a, b)$  et soit  $n \in \mathbb{Z}$  alors

$$\text{pgcd}(a, b + na) = \text{pgcd}(a, b).$$

***Démonstration*** : Soit  $d = \text{pgcd}(a, b + na)$  ; puisque  $c = \text{pgcd}(a, b)$ ,

alors  $c \mid a$  et  $c \mid b$ , d'où  $c \mid b + na$ ,

$(c \mid a \text{ et } c \mid b + na) \Rightarrow c \mid d$ , mais aussi

$(d \mid a \text{ et } d \mid b + na) \Rightarrow d \mid a \text{ et } d \mid b$  c'est à dire  $d \mid \text{pgcd}(a, b) = c$

donc  $c \mid d$  et  $d \mid c$  on en déduit que  $c = d$ .

– **Algorithme d'Euclide simple** :

Soit  $a, b \in \mathbb{N}$  où  $a > 0$ , on applique successivement la division euclidienne et on obtient la suite d'équations :

$$b = aq_1 + r_1 \quad 0 < r_1 < a$$

$$a = r_1q_2 + r_2 \quad 0 < r_2 < r_1$$

$$r_1 = r_2q_3 + r_3 \quad 0 < r_3 < r_2$$

...

...

$$r_{j-2} = r_{j-1}q_j + r_j \quad 0 < r_j < r_{j-1}$$

$$r_{j-1} = r_jq_{j+1}$$

***Théorème*** : Si  $\text{pgcd}(a, b) = c$  alors  $c = r_j$

**Démonstration :** D'après le théorème précédent, on a successivement :

$$\text{pgcd}(a, b) = \text{pgcd}(a, r_1) = \text{pgcd}(r_1, r_2) = \dots = \text{pgcd}(r_{j-1}, r_j) = r_j$$

– Algorithme d'Euclide étendu :

**Théorème :** Si  $c = \text{pgcd}(a, b)$ , il existe  $x, y \in \mathbb{Z}^2$  tels que  $c = xa + yb$ .

**Démonstration :** On écrit  $r_j = r_{j-2} - q_j r_{j-1}$ .

En remplaçant  $r_j$  par son combinaison linéaire à partir de l'équation précédent,

$$\text{on a } r_j = r_{j-2} - q_j(r_{j-3} - q_{j-1}r_{j-2}) = (1 + q_j q_{j-1})r_{j-2} + (-q_j)r_{j-3}$$

En continuant de cette façon, on arrive à exprimer  $r_j$  comme une combinaison linéaire de  $a$  et  $b$ .

**Exemple :**

1. Calculer  $\text{pgcd}(480, 126)$ .

$$480 = (126 \times 3) + 102$$

$$126 = (102 \times 1) + 24$$

$$102 = (24 \times 4) + 6$$

$$24 = (6 \times 4)$$

d'après l'algorithme d'Euclide simple, on trouve que  $\text{pgcd}(480, 126) = 6$ .

2. Exprimer ce nombre comme une combinaison linéaire de 480 et 126.

$$6 = 102 - (24 \times 4) = 102 - (126 - 102) \times 4 = (5 \times 102) - (4 \times 126)$$

$$= (480 - 126 \times 3) \times 5 - (4 \times 126) = (5 \times 480) - (19 \times 126).$$

La complexité de ces calculs est  $O(\max(\ln(n1), \ln(n2))^2)$ .

## Appendice B

### Théorème de Bézout

- **Identité de Bézout** [12] : Soient  $a$  et  $b$  des entiers non nuls, et  $d$  leur *pgcd*, il existe alors deux entiers  $u$  et  $v$  dans  $\mathbb{Z}$  tels que :

$$au + bv = d$$

**Démonstration** : Supposons  $a$  et  $b$  strictement positifs et reprenons l'algorithme d'Euclide pour retrouver le *pgcd* de ces deux nombres :

$$b = aq_1 + r_1 \quad 0 < r_1 < a$$

$$a = r_1q_2 + r_2 \quad 0 < r_2 < r_1$$

$$r_1 = r_2q_3 + r_3 \quad 0 < r_3 < r_2$$

...

$$r_{j-2} = r_{j-1}q_j + r_j \quad 0 < r_j < r_{j-1}$$

$$r_{j-1} = r_jq_{j+1}$$

Montrons par récurrence que tous les restes  $r_1, r_2, r_3, \dots$  s'écrivent sous la forme

$$r_i = u_i a + v_i b.$$

Cela est vrai pour  $r_1$  :  $r_1 = b - q_1 a$

et pour  $r_2$  :  $r_2 = a - (b - q_1 a)q_2 = -b + (1 + q_1 q_2)a$ .

Supposons que cela soit vrai pour un reste  $r_i$  :  $r_i = bu_i + av_i$  et aussi pour le reste précédent  $r_{i-1}$  :  $r_{i-1} = bu_{i-1} + av_{i-1}$

On a donc

$$\begin{aligned} r_{i+1} &= r_{i-1} - r_i q_{i+1} = (bu_{i-1} + av_{i-1}) - (bu_i + av_i)q_{i+1} \\ &= (u_{i-1} - u_i q_{i+1})b + (v_{i-1} - v_i q_{i+1})a \end{aligned}$$

Par récurrence, le résultat est prouvé pour tous les restes et en particulier pour le reste  $r_j$  qui est justement le *pgcd* de  $a$  et  $b$ .

- **Théorème de Bézout** : Pour deux entiers  $a$  et  $b$ , premiers entre eux et non nuls, on peut dire qu'il existe deux entiers  $u$  et  $v$  tels que  $au + bv = 1$

**Démonstration** : Nous savons que le *pgcd* de deux nombres premiers entre eux est 1 donc la démonstration citée ci-dessus est valable pour ce théorème.

La complexité de ces calculs est  $O(\max(\ln(n1), \ln(n2))^2)$ .

## Appendice C

### Le Théorème Chinois

#### C.1 Le cas général

Ce théorème s'énonce de 2 manières :

1. Soient  $m_1, m_2, \dots, m_r$  une suite d'entiers positifs premiers entre eux deux à deux. Alors le système de congruences :

$$x \equiv a_1 \pmod{m_1},$$

$$x \equiv a_2 \pmod{m_2},$$

...

$$x \equiv a_r \pmod{m_r}.$$

a une solution unique  $x$  modulo  $M = m_1 \times m_2 \times \dots \times m_r$  :

$$x = a_1 M_1 y_1 + a_2 M_2 y_2 + \dots + a_r M_r y_r$$

avec  $M_i = \frac{M}{m_i}$ ,  $y_i M_i \equiv 1 \pmod{m_i}$

2. Soit  $m = m_1 \times m_2 \times \dots \times m_r$  alors l'application

$$\mathbb{Z}/m\mathbb{Z} \leftrightarrow \mathbb{Z}/m_1\mathbb{Z} \times \mathbb{Z}/m_2\mathbb{Z} \times \dots \times \mathbb{Z}/m_r\mathbb{Z}$$

est **bijective**. Le théorème des restes chinois permet de construire l'application réciproque.

Si  $x = (x_1, x_2, \dots, x_r)$  et  $y = (y_1, y_2, \dots, y_r)$  alors :

$$x + y = (x_1 + y_1, x_2 + y_2, \dots, x_r + y_r)$$

$$x \times y = (x_1 \times y_1, x_2 \times y_2, \dots, x_r \times y_r)$$

En terme de temps de calcul c'est très rapide :

Le calcul de  $\mathbb{Z}/m\mathbb{Z} \leftrightarrow \mathbb{Z}/m_1\mathbb{Z} \times \mathbb{Z}/m_2\mathbb{Z} \times \dots \times \mathbb{Z}/m_r\mathbb{Z}$  consiste en  $r$  divisions euclidiennes donc la complexité est  $O((\ln m)^r)$ .

(En d'autres termes, ceci définit un morphisme d'anneau).

## C.2 Théorème Chinois pour les entiers R.S.A.

Ecrit formellement, le Théorème Chinois dit :

Pour  $n = p \times q$ ,  $\mathbb{Z}/n\mathbb{Z} \rightarrow \mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z}$ .

Donc  $(x \bmod n) \rightarrow ((x \bmod p), (x \bmod q))$ .

Nous savons que  $p$  et  $q$  sont premiers et distincts, le Théorème de Bézout (Voir Appendice B, page 49) dit :  $\exists u, v \in \mathbb{Z} \mid up + vq = 1$ .

Si nous prenons  $((x \bmod p), (y \bmod q))$  alors nous avons :  $(xvq + yup \bmod n)$ .

Prouvons ceci :

$$xvq + yup \bmod p = xvq \bmod p$$

(car  $p$  divise  $yup$ ).

Nous pouvons en plus remplacer  $vq$  par  $1 - up$  car  $up + vq = 1$  :

$$x(1 - up) \bmod p = x - xup \bmod p,$$

et comme  $xup$  est divisible par  $p$ , il ne nous reste que  $x \bmod p$ .

Donc :

$$xvp + yup \bmod p = x \bmod p.$$

Et ce sont les mêmes calculs pour modulo  $q$ , donc nous avons également :

$$xvp + yup \pmod{q} = y \pmod{q}.$$

En fait l'idée principale de ce théorème est la suivante : "si un entier  $n$  se divise en petits facteurs premiers, alors les opérations modulo  $n$  se décomposent en petites opérations modulo chacun des facteurs".

Voyons ceci avec un exemple :

Soient  $p = 11$  et  $q = 17$  donc  $n = 187$ ,

Le théorème de Bézout nous dit que si  $p$  et  $q$  sont premiers entre eux - c'est le cas ici- alors 'il existe 2 entiers  $u$  et  $v$  tels que :  $up + vq = 1$  :

$$u = -3 \text{ et } v = 2$$

Prenons un entier  $z$  au hasard :  $z = 20$ , donc  $z \pmod{n} = 20 \pmod{187}$ .

Nous pouvons dire que  $z \pmod{187} \rightarrow ((z \pmod{11}), (z \pmod{17}))$ .

Notre  $x = z \pmod{p} = 20 \pmod{11} = 9$  et notre  $y = z \pmod{q} = 20 \pmod{17} = 3$ .

$xvp + yup \pmod{n}$  nous donne  $9 \times 2 \times 17 + 3 \times (-3) \times 11 = 207 \pmod{187} = 20 \pmod{187} = z \pmod{n}$ .

Donc nous avons bien vu que nous pouvons décomposer les calculs modulo  $n$  que nous faisons en deux calculs modulo  $p$  et modulo  $q$ . Les calculs  $\mathbb{Z}/n\mathbb{Z}$  sont conservés en passant à  $\mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z}$ .

## Appendice D

### Théorème de Fermat

– **Lemme** [17] : Soit  $p > 0$  un nombre premier et  $a$  et  $b$  des entiers alors :

$$(a + b)^p \equiv a^p + b^p \pmod{p}.$$

**Démonstration** : Pour démontrer le lemme, il suffit de démontrer que :

$$p \mid C_p^k \quad 1 \leq k \leq p - 1$$

car

$$(a + b)^p = a^p + b^p + \sum_{k=1}^{p-1} a^{p-k} b^k.$$

$$C_p^k = \frac{p!}{(p-k)!k!} \Leftrightarrow k! \times C_p^k = p(p-1) \dots (p-k+1).$$

Il est clair que  $p \mid k! \times C_p^k$  et comme  $1 \leq k \leq p - 1$  et  $p$  premier, on conclue donc que  $p \nmid k!$  donc  $p \mid C_p^k$ .

– **Petit Théorème de Fermat** : Soit  $p > 0$  un nombre premier et  $a$  un entier positif.

$$\text{Si } 0 \leq a \leq p - 1 \text{ alors } a^p \equiv a \pmod{p}.$$

**Démonstration** : Pour démontrer ce théorème par récurrence, on a besoin d'une proposition :

$$P(n) : n^p \equiv n \pmod{p}, n \in \mathbb{N}$$

La proposition est vraie pour le premier rang car :

$$P(0) : 0^p = 0$$

On considère que  $P(n)$  est vraie et on va montrer que  $P(n + 1)$  est vraie.

$$(n + 1)^p \equiv n + 1 \pmod{p}.$$

En utilisant le lemme on trouve  $(n + 1)^p \equiv n^p + 1 \equiv n + 1 \pmod{p}$ .  
d'après l'hypothèse de récurrence, on a :

$$n^p \equiv n \pmod{p}.$$

donc on peut remplacer  $n^p$  par  $n$  alors :

$$(n + 1)^p \equiv n^p + 1 \equiv n + 1 \pmod{p}.$$

Conclusion :  $P(n)$  est vraie  $\forall n \in \mathbb{N}$

– **Théorème** :

Soit  $n = pq$  avec  $p$  et  $q$  premiers et soient  $e$  et  $d$  tels que :  $ed \equiv 1 \pmod{\phi(n)}$ .

On a :  $\forall X \in \mathbb{Z}, (X^e)^d \equiv X \pmod{n}$ .

***Démonstration*** :

Montrons d'abord que  $(X^e)^d \equiv X \pmod{p}$  (1)

– **Premier cas** : Si  $p \nmid X$

On sait que  $ed \equiv 1 \pmod{\phi(n)}$  et que  $\phi(n) = (p - 1)(q - 1)$ .

Donc, il existe un entier  $k'$  tel que  $ed = 1 + k'(p - 1)(q - 1)$ ; ou encore  $ed = 1 + k(p - 1)$  avec  $k = k'(q - 1)$ .

On peut écrire maintenant :

$$X^{ed} \equiv X^{k(p-1)} + 1 \pmod{p}$$

Or, le petit théorème de Fermat dit : "si  $\text{pgcd}(X, p) = 1$ , alors  $X^{p-1} \equiv 1 \pmod{p}$ ". . .

$$\text{Donc } X^{k(p-1)} \equiv 1 \pmod{p}$$

$$\text{Finalement, } X^{ed} \equiv X^{k(p-1)+1} \equiv 1 \times X \equiv X \pmod{p}$$

– **Deuxième cas** : Si  $p \mid X$

$$X \equiv 0 \pmod{p}$$

$$\text{donc } (X^e)^d \equiv 0 \pmod{p} \equiv X \pmod{p}$$

Idem pour  $q$  : on obtient  $X^{ed} \equiv X \pmod{q}$

$$(1) \Rightarrow X^{ed} \equiv X \pmod{p}$$

$$(2) \Rightarrow X^{ed} \equiv X \pmod{q}$$

$$(1) + (2) \Rightarrow X^{ed} \equiv X \pmod{pq} \Rightarrow X^{ed} \equiv X \pmod{n} \text{ [cqfd]}$$

## Appendice E

### Fonctions à sens unique

#### E.1 Fonctions à sens unique

La notion de fonction à sens unique (one-way function) [8] a été une grande révolution lorsqu'on s'est rendu compte de son potentiel cryptographique. Une fonction  $f$

$$E \rightarrow F$$

$$x \rightarrow f(x)$$

est dite à sens unique si :

1. Calculer  $f(x)$  à partir de n'importe quel  $x$  est très facile,
2. Pour la plupart des  $y \in f(E)$ , il est très difficile, voire impossible, de trouver un  $x$  tel que  $f(x) = y$  à moins de tester un nombre extrêmement grand de  $x$  (ce qui prendra un temps énorme, voire des dizaines ou des centaines d'années) ou d'avoir une chance incroyable qui nous fasse tomber sur le bon  $x$  tout de suite mais il est déraisonnable de compter sur cela.

En fait, dire que  $f(x)$  est facilement calculable veut dire qu'il existe un algorithme qui, pour tout  $x$ , calcule  $f(x)$  en un temps polynomial en la longueur de  $x$ .

L'existence de telles fonctions n'est pas encore vraiment prouvé donc on ne peut pas dire qu'il est impossible de calculer  $x$  à partir de  $f(x)$ , c'est pourquoi on se contente de dire que c'est très difficile. . . Il faut aussi noter qu'il faut prendre un ensemble de départ  $E$  très grand pour que le nombre d'antécédents  $x$  soit important, pour rendre encore plus difficile la chance de tomber sur le  $x$  cherché.

**Exemples :** [18]

–  $f_{MULT}$  :

Considérons la fonction  $f_{MULT}$  qui consiste en la multiplication de deux grands nombres premiers, donc

$$f_{MULT}(p, q) = p \times q.$$

En fait étant donné  $p$  et  $q$ , c'est très facile de calculer leur multiplication, mais factoriser ce produit est très difficile, d'ailleurs on ne connaît pas d'algorithme qui fait ce calcul en temps polynomial.

–  $f_{EXP}$  :

La fonction  $f_{EXP}$  est une autre fonction qu'on considère à sens unique, qui consiste en l'exponentiation modulo un nombre premier donc

$$f_{EXP}(r, x, p) = r^x \pmod{p}$$

avec  $p$  premier. Si nous avons ces données il est très facile de prendre la puissance  $x$  de  $r$  et de la réduire en modulo  $p$  mais quand nous avons en main  $r^x \pmod{p}$  il nous est impossible de retrouver la valeur de  $x$  en temps polynomial. C'est d'ailleurs le problème du logarithme discret.

–  $f_{RSA}$  :

$f_{MULT}$  et  $f_{EXP}$  ne peuvent pas directement être utilisées comme base à un cryptosystème asymétrique mais l'union des deux si. Supposons  $p$  et  $q$  deux nombres premiers, et soit leur produit  $n = p \times q$ . On prend un  $e$  premier avec  $\phi(n) (= (p-1)(q-1))$ . La fonction  $f_{RSA}$  s'écrit de la façon suivante :

$$f_{RSA}(m, e, p, q) = (m^e \pmod{n}, n, e).$$

Plus précisément ce que cette fonction fait c'est juste de calculer  $m^e \pmod n$  et de renvoyer ce résultat avec les valeurs de  $n$  et  $e$ . Il nous est impossible de retrouver la valeur de  $m$ , qu'avec les données de sortie de la fonction  $f_{RSA}$ , c'est donc bien une fonction à sens unique.

On se sert beaucoup des fonctions à sens unique pour faire des cryptosystèmes et des protocoles d'identification.

## E.2 Fonctions à sens unique à trappes

Une fonction à sens unique est dite à trappe ou à brèche secrète si elle est à sens unique sauf pour toute personne connaissant un secret ou une brèche, permettant de calculer un algorithme d'inversion rapide.

### *Exemple :*

Dans le système R.S.A. (voir section 1.2, page 6), nous avons une donné  $d$ , qui est l'inverse de  $e \pmod{\phi(n)}$ , en fait nous pouvons dire que ce  $d$  est la trappe de la fonction  $f_{RSA}$  puisque la connaissance de ce  $d$  nous suffit pour retrouver le  $m$ . Puisque  $e.d \equiv 1 \pmod{\phi(n)}$ ,

$$(m^e)^d = m^{1+k\phi(n)} = m \pmod n$$

Car  $m^{\phi(n)} \equiv 1 \pmod n$  d'après le théorème de Fermat(voir Appendice D, page 54). Donc la connaissance de  $d$  nous amène, au message clair  $m$ .