h e g

Haute école de gestion
Genève

La cryptographie

Travail de Bachelor réalisé en vue de l'obtention du Bachelor HES

par :

Daniel LAMAS

Conseiller au travail de Bachelor :

Peter DAEHNE, Professeur HES

Genève, le 5 juin 2015

Haute École de Gestion de Genève (HEG-GE)

Filière Informatique de Gestion

Hes·so GENÈVE
Haute Ecole Spécialisée
de Suisse occidentale

Déclaration

Ce travail de Bachelor est réalisé dans le cadre de l'examen final de la Haute école de gestion de Genève, en vue de l'obtention du titre de Bachelor en Informatique de Gestion.

L'étudiant a envoyé ce document par email à l'adresse remise par son conseiller au travail de Bachelor pour analyse par le logiciel de détection de plagiat URKUND, selon la procédure détaillée à l'URL suivante : http://www.urkund.fr/student_gorsahar.asp.

L'étudiant accepte, le cas échéant, la clause de confidentialité. L'utilisation des conclusions et recommandations formulées dans le travail de Bachelor, sans préjuger de leur valeur, n'engage ni la responsabilité de l'auteur, ni celle du conseiller au travail de Bachelor, du juré et de la HEG.

« J'atteste avoir réalisé seul le présent travail, sans avoir utilisé des sources autres que celles citées dans la bibliographie. »

Fait à Genève, le 5 juin 2015

Daniel LAMAS

Remerciements

Je tiens tout d'abord à remercier Monsieur Peter DAEHNE pour m'avoir suivi tout au long de l'élaboration de ce travail. Ces conseils et remarques pertinentes m'ont beaucoup aidé. Je tiens également à remercier Monsieur Michel KUHNE pour ses explications à propos des algorithmes de multiplication, ainsi que pour les diverses informations à propos d'Alan Turing.

Je remercie également ma famille qui m'a toujours soutenue tout au long de mes études. Finalement, je remercie particulièrement ma sœur Paula LAMAS, Michael LARANJO et Julien FLUBACHER pour leur relecture ainsi que Szabolcs ZAKANY pour ses explications concernant les corps finis.

Résumé

La cryptographie est la discipline qui permet de protéger des messages. Si un message est intercepté, il devrait ne pas être compris ou ne pas être déchiffré facilement. Les premières traces de la cryptographie remontent au XVI^{ème} siècle avant J.-C. Depuis cette époque, elle n'a fait qu'évoluer. Si à ses débuts il était question de chiffrement symétrique, c'est-à-dire, avec une clé permettant de chiffrer et déchiffrer un message, depuis le milieu des années 70, une nouvelle méthode révolutionnaire de partage de messages est apparue avec des clés publiques : le cryptage asymétrique. Actuellement, ces deux types de cryptage sont utilisés conjointement. Les algorithmes asymétriques pour transmettre des clés de chiffrement et les algorithmes symétriques afin de chiffrer les données à protéger.

Ce travail de Bachelor parle tout d'abord de l'histoire de la cryptographie. Ensuite, il décrit plusieurs algorithmes symétriques et asymétriques, ainsi que leur implémentation.

Table des matières

Liste des tableaux

Liste des figures

1. Introduction

Actuellement, le réseau Internet est omniprésent. Que ce soit sur un smartphone, un ordinateur ou même sur un objet comme une montre, Internet est partout. Il possède des caractéristiques fascinantes qui n'ont pas besoin d'être décrites tellement son utilisation est rentrée dans les mœurs.

Toutefois, face à cette ouverture, certaines questions doivent se poser concernant la sécurité des données. Que se passe-t-il lorsqu'un individu achète un billet d'avion sur Internet et doit rentrer des informations sensibles comme un numéro de carte de crédit ? Comment les mots de passe d'un utilisateur sont-ils enregistrés dans les différents sites auxquels il est inscrit. En effet, après les scandales liés à Wikileaks (1), la divulgation d'informations sensibles du Gouvernement Américain par Edward Snowden (2) ou encore le vol de données confidentielles de l'entreprise Sony (3), il est normal de se demander comment les données sont protégées ? Les différents gouvernements et les grandes entreprises dépensent énormément d'argent pour assurer la sécurité de leurs données. Malgré cela, les données arrivent à être dérobées.

Au vu de cette problématique, la cryptographie va être l'objet de ce travail. Si des données sont interceptées mais qu'elles sont chiffrées et donc incompréhensibles, alors ces données sont inutilisables. En revanche, si une méthode de cryptage pas très robuste a été utilisée, une personne malveillante pourrait retrouver les données d'origine. En partant du principe qu'Internet est un réseau ouvert où n'importe quelle communication peut être interceptée, la cryptographie prend alors tout son sens.

Ce travail va tout d'abord parler de l'histoire de la cryptographie, afin de voir son évolution. La partie suivante va porter sur l'étude de différentes méthodes de cryptage. Pour finir, certains algorithmes vont être programmés afin de voir plus en détail leur fonctionnement.

2. Définition

Le terme cryptographie provient des deux mots grecs anciens « Kruptos » qui signifie « cacher » et « graphein » qui signifie « écrire » (4). Ce qui signifie littéralement, « cacher l'écriture ». Le Petit Larousse donne la définition suivante : « *Ensemble des techniques de chiffrement qui assurent l'inviolabilité de textes et, en informatique, de données.* » (5)

3. Histoire de la cryptographie

L'être humain a toujours eu le besoin de cacher des informations. Que ce soit un secret ne devant pas être divulgué dans son entourage qui compromettrait des individus ou encore d'informations tactiques lors des différentes batailles et guerres ayant marqué l'Histoire. Dissimuler des informations a toujours été une nécessité. Voici une liste non exhaustive de différentes techniques utilisées au fil des siècles qui marquent l'évolution de la cryptographie à travers les âges.

Les premières traces de cryptographie remontent à l'Antiquité, plus précisément aux alentours du XVI^{ème} siècle avant J.-C. Un potier en Irak avait gravé sur une table en argile sa recette en supprimant les consonnes et en modifiant l'orthographe des mots. (6)

Par la suite, entre le X^{ème} et le VII^{ème} siècle avant J.-C., les Grecs utilisaient des scytales, des sortes de bâtons en bois. Quand l'émetteur voulait communiquer, il enroulait une bande de cuir sur la scytale et y inscrivait le message (une lettre par bout de bande). Une fois la bande déroulée, les lettres n'étaient plus ordonnées et n'avaient donc plus aucun sens. Le seul moyen de pouvoir comprendre le message était d'enrouler la bande sur une scytale de même diamètre pour que les lettres puissent s'aligner correctement. (6) (7) (8)

Figure 1 : Une scytale

Source : (8)

Au I^{er} siècle avant J.-C., le cryptage de César faisait son apparition. Ce chiffrement était utilisé par Jules César pour communiquer de façon secrète. C'est un des premiers chiffrements par substitution. Son principe est simple, il suffit de substituer chaque

caractère du message d'origine par un autre dans l'alphabet, qui se trouve toujours à une distance fixe. (9)

Au fur et à mesure que le temps passe, comme les méthodes utilisées précédemment deviennent de plus en plus faciles à comprendre et à détourner, d'autres mesures ont dû être prises afin de pouvoir empêcher le décryptage des messages. Au XVI^{ème} siècle, le chiffre de Vigenère fit son apparition. Il s'agit également d'un chiffrement par substitution, mais il est plus avancé que le chiffrage de César. Plutôt que d'utiliser un décalage fixe, le chiffre de Vigenère se base sur une clé qui va déterminer le décalage pour chaque caractère. (10)

A partir du siècle dernier, la cryptographie a joué un rôle clé lors des différentes guerres. La technologie a commencé à être utilisée avec par exemple, les messages radio ou les télégrammes. Les informations pouvaient être beaucoup plus facilement interceptées que dans le passé. Ce fut le cas du ministre des affaires étrangères allemand Arthur Zimmermann, le 16 janvier 1917. Il a envoyé un télégramme à l'ambassadeur allemand au Mexique pour lui proposer un complot afin de garder les Etats-Unis hors de la guerre. Cependant, le message a été intercepté par les Britanniques car les Allemands ont utilisé le « code diplomatique 0075 » qui était un code de substitutions comprenant 1000 substitutions possibles (voir Annexe 1). (11)

Figure 2 : Machine Enigma

Source : (20)

Après la première Guerre Mondiale, la machine Enigma a été créée et les Allemands, ayant compris l'importance que peuvent avoir les informations sensibles, ont investi dans une version militaire plus complexe de cette machine. Toutefois, bien que le fonctionnement de la machine soit complexe, des chercheurs polonais ont étudié le fonctionnement de la machine pour tenter de décrypter les messages. Par la suite, le célèbre mathématicien, cryptologue et informaticien britannique Alan Turing a collaboré au décryptage des messages codés par la machine. Grâce à cela, les Britanniques ont pu déchiffrer des messages, ce qui a été un sérieux avantage qui leur a permis de gagner la guerre. Il a même été estimé que la découverte de Turing a raccourci la deuxième guerre mondiale de deux ans (12).

En 1977, le standard de chiffrement DES[1] est proposé comme standard par le NIST[2]. C'est un chiffrement par bloc de textes de 56 bit (en réalité 64, mais un bit est utilisé pour contrôler la parité). Il est resté comme standard jusqu'à la fin des années 1990. En effet, à la fin des années 70, les ordinateurs n'étaient pas suffisamment puissants pour déchiffrer un message codé avec DES, mais avec la multiplication de leur puissance, le déchiffrage est devenu possible dans un temps raisonnable. Une information chiffrée en DES peut être déchiffrée en 30h avec 1000 PC en parallèle cadencés à 1GHz. C'est pourquoi, à la fin des années 90, le NIST a lancé un concours pour remplacer le DES et définir un nouveau standard de cryptage qui deviendra AES[3]. Les gagnants sont les deux chercheurs belges Joan Daemen et Vincent Rijmen avec leur chiffrement Rijndael. (13)

A l'époque du DES, plus précisément en 1976, les deux cryptologues Whitfield Diffie et Martin Hellman publient un article « New Directions in Cryptography » (14) qui propose une nouvelle façon de crypter les données. C'est à partir de leur article que le cryptage asymétrique est né avec le chiffrement de Diffie-Hellman. Jusque-là, tous les cryptosystèmes étaient symétriques. Bien que ce type de cryptage fonctionne bien, garder la clé secrète ou bien tout simplement la communiquer peut devenir contraignant. Avec leur système, le problème de clé secrète n'est plus un problème car le fonctionnement se base sur une clé pouvant être connue de tout le monde.

[1] Data Encryption Standard

[2] National Institute of Standards and Technology

[3] Advanced Encryption Standard

4. La cryptographie

4.1 Cryptographie symétrique

La cryptographie symétrique (ou cryptographie à clé secrète) est la forme la plus ancienne de cryptographie (15). Ce chiffrement fonctionne en principe avec une clé secrète, bien qu'il existe certains chiffrements symétriques qui n'utilisent pas de clé, comme par exemple le chiffre de César. Dans le cas des chiffrements avec clé, le principe est le suivant : L'émetteur du message chiffre les données grâce à une clé. Cette clé est généralement une chaîne de caractères. Le message est chiffré et sans la clé il est quasi impossible (le niveau d'impossibilité dépend du niveau de protection du chiffrement utilisé ainsi que de la complexité de la clé utilisée) de retrouver le message d'origine. L'émetteur doit donc transmettre la clé aux personnes à qui il désire transmettre le message s'il veut que son message puisse être lu.

Figure 3 : Schéma de fonctionnement de la cryptographie symétrique

4.1.1 Principe de Kerckhoffs

En 1883, le cryptologue hollandais Auguste Kerckhoffs publiait un essai intitulé « La cryptologie militaire ». Cet essai présente une liste des six règles à respecter en cryptographie afin d'assurer un système confidentiel. Bien que ce soit des règles anciennes (elles font par exemple référence à la correspondance par télégramme), elles sont toujours applicables de nos jours. (16)

Tableau 1 : Principe de Kerckoffs

1. Le système doit être matériellement, sinon mathématiquement, indéchiffrable ;
2. Il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi ;

3. La clé doit pouvoir en être communiquée et retenue sans le secours de notes écrites, et être changée ou modifiée au gré des correspondants ;
4. Il faut qu'il soit applicable à la correspondance télégraphique ;
5. Il faut qu'il soit portatif, et que son maniement ou son fonctionnement n'exige pas le concours de plusieurs personnes ;
6. Enfin, il est nécessaire, vu les circonstances qui en commandent l'application, que le système soit d'un usage facile, ne demandant ni tension d'esprit, ni la connaissance d'une longue série de règles à observer.

Source : (16)

4.1.2 Méthodes de chiffrement

4.1.2.1 Chiffrement par flot

Les algorithmes basés sur le principe de chiffrement par flot chiffrent ou déchiffrent un message à la volée. Leur fonctionnement se base sur un générateur de nombres pseudo-aléatoires et un mécanisme de substitution bit à bit. Les algorithmes se basant sur ce principe sont réputés rapides. (13)

4.1.2.2 Chiffrement par blocs

Le chiffrement par blocs fonctionne différemment. Au lieu de prendre chaque bit un par un, les messages sont découpés en blocs (la taille des blocs dépend de la clé). Ensuite, chaque bloc est additionné à la clé et un traitement de type permutation, opération XOR ou autre est appliqué à chaque bloc. (13)

4.1.3 Algorithmes

4.1.3.1 ROT13

Le chiffrement ROT13 (Rotate by 13 places) est une variante du chiffre de César, précédemment vu dans le chapitre consacré à l'histoire de la cryptographie. C'est un chiffrement à décalage fixe, et comme son nom l'indique, le décalage est de 13 positions. C'est donc un chiffrement par flot. Cela signifie que pour chiffrer ou déchiffrer un message, il faut substituer la lettre de base par celle qui se trouve 13 positions plus loin, comme dans le tableau de correspondance des caractères (Tableau 2) ci-dessous. (17)

Tableau 2 : Chiffrement ROT13

Position dans l'alphabet	0	1	2	3	4	5	6	7	8	9	10	11	12
Alphabet	A	B	C	D	E	F	G	H	I	J	K	L	M
	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Position dans l'alphabet	13	14	15	16	17	18	19	20	21	22	23	24	25

Ainsi, le chiffrement du message « La cryptographie » deviendra « Yn pelcgbtencuvr »

L	A		C	R	Y	P	T	O	G	R	A	P	H	I	E
Y	N		P	E	L	C	G	B	T	E	N	C	U	V	R

Pour décrypter le message, il suffit de refaire la même opération. En effet, l'alphabet est coupé en deux et contient 26 lettres (donc est un chiffre pair). En additionnant ou en soustrayant de 13 (soit la moitié), le résultat sera le même.

Toutefois, la méthode « visuelle » a été présentée. Mathématiquement parlant, ce qui a été expliqué reviendrai à dire cela (avec l'exemple de « La cryptographie ») :

La lettre L se trouve en 11ème position dans l'alphabet. En lui additionnant 13, le résultat deviendra 24, ce qui va donc correspondre à la lettre de l'alphabet située en 24ème position, c'est-à-dire Y. Pour les lettres A et C qui suivent, le même traitement est appliqué. En revanche, pour la lettre R, qui est à la 17ème position dans l'alphabet, si on lui additionne 13, le résultat sera 30, or l'alphabet ne contient que 26 lettres. Pour combler à ce problème il suffit d'utiliser l'opérateur modulo, qui permettra d'avoir le reste de la division par 26 et donc de retomber sur la bonne lettre.

Tableau 3 : Fonction de calcul pour ROT13

(Position de la lettre + 13) mod 26 = Position de la lettre chiffrée

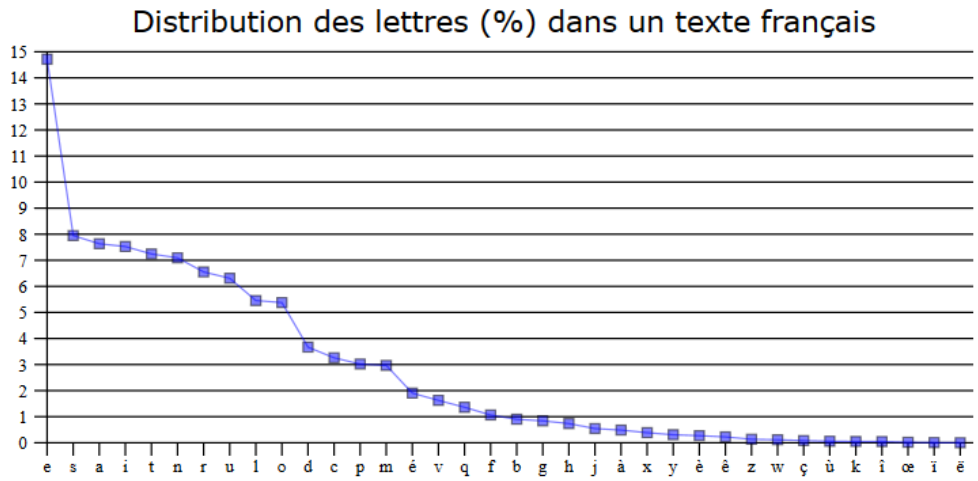
Ainsi, pour la lettre A, le calcul (0+13) mod 26 = 13. Comme 13 (0+13) est inférieur à 26, il n'y aura pas de reste pour la division entière 13/26.

Pour la lettre R, le calcul (17+13) mod 26 = 4. Comme 30 (17+13) est supérieur à 26, le reste de la division entière 30/26 est égal à 4.

Cette façon de calculer fonctionne dans les deux sens (chiffrement/décryptage) et ne comprend aucun test au moment du chiffrement. En effet, les tests pourraient nuire aux performances de l'algorithme.

Il est à noter que le tableau commence à l'indice 0 et non à 1. En effet, si le tableau commençait à 1, le cas numéro 13 poserait problème. En effet, (13+13) modulo 26 donne 0, alors que la valeur chiffrée aurait dû être 26. C'est pourquoi le tableau commence à 0 et donne un indice maximum de 25. De cette façon, le cas numéro 12 donne un résultat de 25 (donc Z) et le cas numéro 13 donne un résultat de zéro (26 mod 26 = 0) et donc donne A comme résultat. Cet algorithme a le mérite d'être très simple à mettre en place (une opération arithmétique pour chaque caractère), mais il ne respecte pas le principe de Kerckhoffs (Principe numéro 2), car il est déchiffrable très facilement. En effet, c'est un algorithme de substitution mono alphabétique (18), ce qui signifie qu'un caractère aura toujours une autre même valeur quand il sera chiffré. A cause de ce problème, une personne malintentionnée désirant décrypter le message pourrait alors analyser les occurrences de chaque lettre dans le texte et avoir des informations qui lui permettraient de retrouver le texte original. Par exemple, il y a une occurrence moyenne de presque 15% pour la lettre E dans un texte français (19). En analysant un texte chiffré avec ROT13, la lettre E pourrait être repérée facilement et ainsi il serait aisé de connaître le décalage des lettres.

Figure 4 : Occurrences des lettres dans un texte en Français



Source : (19)

4.1.3.2 Chiffrement de Vernam

Le chiffre de Vernam ou encore l'algorithme à masque jetable est une méthode de chiffrement qui est théoriquement indéchiffrable. Toutefois, pour que cet algorithme fonctionne de façon optimale, les trois contraintes suivantes doivent absolument être respectées :

1. La clé doit être absolument aléatoire
2. La clé doit être de la même longueur que l'information à chiffrer
3. La clé ne doit être utilisée qu'une seule et unique fois (d'où le nom « chiffrement à masque jetable »). (16)

Cet algorithme est une version améliorée des algorithmes de décalage simple (ou monoalphabétique). En effet, il inclut la notion de clé. Il est aussi une version améliorée du chiffrement de Vigenère qui fonctionne de la même façon, à la différence qu'il n'a pas les trois contraintes décrites précédemment, ce qui rend l'algorithme vulnérable.
Plutôt que d'avoir un seul décalage fixe, chaque caractère de la clé définit le décalage du caractère à chiffrer. On parle alors de chiffrement par substitution polyalphabétique. C'est pour cette raison que la clé doit avoir la même taille que le message à chiffrer et donc le chiffrement est réalisé par flot. Ainsi, un caractère n'aura pas la même valeur une fois chiffré, grâce à la clé. (16)

Prenons l'exemple du message « LA_CRYPTOGRAPHIE » chiffré avec la clé « FRIYL_FTYBDGZWF_ ». La première lettre du message à chiffrer est L (douzième lettre de l'alphabet) et la première lettre de la clé est F (sixième lettre de l'alphabet). Cela signifie que la nouvelle valeur de L dans ce cas-là sera R (L(12) + F(6) = R(18)).
Il est assez aisé de chiffrer un message en s'aidant d'une matrice contenant les caractères de la clé et du message comme dans le tableau qui suit.

Tableau 4 : Matrice de chiffrement de Vernam

	_	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
_	_	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	_
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	_	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	_	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	_	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	_	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	_	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	_	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	_	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	_	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	_	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	_	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	_	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	_	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	_	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	_	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	_	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	_	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	_	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	_	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	_	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	_	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	_	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	_	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	_	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	_	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	_	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Source : (16)

Finalement, le message codé deviendra « RSIACYVMMIVHODOE »

Tableau 5 : Message chiffré avec le chiffrement de Vernam

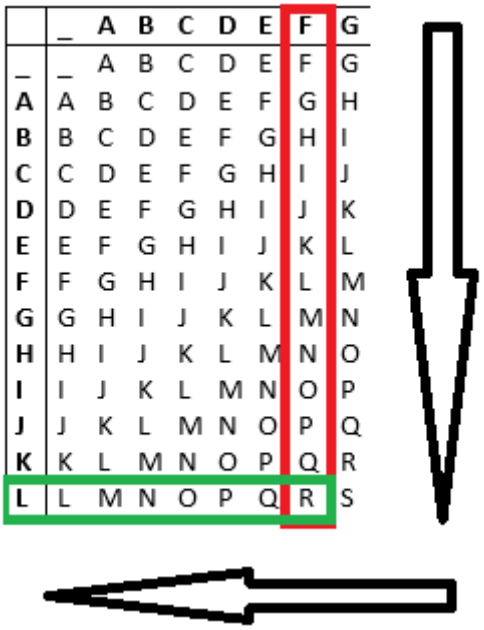
Message	L	A	_	C	R	Y	P	T	O	G	R	A	P	H	I	E
Clé	F	R	I	Y	L	_	F	T	Y	B	D	G	Z	W	F	_
Message chiffrée	R	S	I	A	C	Y	V	M	M	I	V	H	O	D	O	E

On peut constater que les lettres « T » et « O » ont la même valeur une fois chiffrées alors qu'initialement ce ne sont pas les mêmes caractères. De ce fait, une personne malveillante voulant casser la clé ne pourra pas utiliser une méthode de nombre d'occurrences de caractères, comme pour l'algorithme ROT13.

Une fois le message transmis, la personne qui reçoit le message (et qui a bien évidemment en sa possession la clé) devra réaliser l'opération inverse pour déchiffrer le

message. Toujours en utilisant la matrice des caractères, la personne devra chercher dans la colonne du caractère de la clé (dans ce cas-là, pour le premier caractère « F »), le caractère chiffré (R). Une fois le caractère repéré, il faut consulter le caractère de ligne, qui dans ce cas est la lettre L. (16)

Figure 5 : Déchiffrage d'un message avec Vernam



	_	A	B	C	D	E	F	G
_	_	A	B	C	D	E	F	G
A	A	B	C	D	E	F	G	H
B	B	C	D	E	F	G	H	I
C	C	D	E	F	G	H	I	J
D	D	E	F	G	H	I	J	K
E	E	F	G	H	I	J	K	L
F	F	G	H	I	J	K	L	M
G	G	H	I	J	K	L	M	N
H	H	I	J	K	L	M	N	O
I	I	J	K	L	M	N	O	P
J	J	K	L	M	N	O	P	Q
K	K	L	M	N	O	P	Q	R
L	L	M	N	O	P	Q	R	S

Pour résoudre ce problème de façon mathématique, le fonctionnement est similaire à celui de ROT13 mais comme le décalage est variable, au lieu d'additionner 13 comme dans ROT13, la valeur de la clé va déterminer le décalage, comme dans la formule suivante :

(Position du caractère du texte + Position du caractère de la clé) mod 27 = Position de la lettre chiffrée

Ce calcul est applicable en utilisant la matrice du Tableau 4 qui représente les 26 lettres de l'alphabet ainsi qu'un caractère d'espace. Toutefois, il est tout à fait envisageable d'appliquer cet algorithme avec la table ASCII étendue (256 possibilités) et donc le modulo ne serait plus de 27 mais de 256.

Pour déchiffrer le message, il faut tout d'abord soustraire la lettre chiffrée à la lettre de la clé correspondante. Si le résultat est supérieur ou égal à zéro, cela signifie que le résultat obtenu, correspond à la position du caractère (par exemple 5 = E). Si le résultat

est négatif, il faut alors additionner le nombre de caractères de l'alphabet (dans notre cas, 27) pour enfin tomber sur la position du caractère dans l'alphabet.

1. Si position de la lettre chiffrée – position du caractère >= 0 :
(Position de la lettre chiffrée - Position caractère de la clé) =
Position du caractère du texte
2. Si position de la lettre chiffrée – position du caractère < 0 :
(Position de la lettre chiffrée - Position caractère de la clé) + 27 =
Position du caractère du texte

4.1.3.3 Enigma

La machine Enigma est une machine mécanique aux allures d'une machine à écrire. Dotée d'un clavier, elle permet de chiffrer des messages. Lorsqu'une lettre est tapée sur le clavier, un courant électrique traverse les différents composants de la machine pour ainsi allumer la lettre chiffrée correspondante. Si la même lettre est tapée à la suite, une autre lampe sera allumée. C'est un autre type de cryptage par substitution (polyalphabétique) sauf que dans ce cas, il est fait automatiquement par la machine. Cet automatisme a pu être possible grâce à l'utilisation de rotors[4]. Généralement, elle en est dotée de trois mais il existe des versions à quatre, cinq voire six rotors. Les rotors sont alignés les uns à côté des autres et ont 26 contacts électriques (un pour chaque lettre de l'alphabet) sur chaque face du rotor, comme dans la Figure 6 ci-dessous. (20).

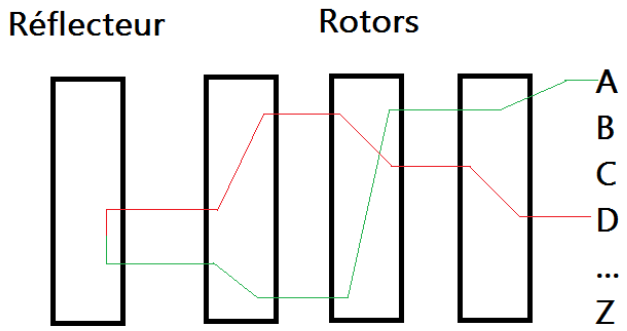
Figure 6 : Contacts électriques sur les rotors d'une machine Enigma

Source : http://upload.wikimedia.org/wikipedia/commons/d/d8/Enigma_rotors_and_spindle_showing_contacts_rachet_and_notch.jpg

[4] Partie tournante d'une machine (Définition du petit Larousse Illustré)

De plus, ils sont câblés électriquement à l'intérieur pour transmettre le courant d'une face à l'autre. Ainsi, quand une lettre est frappée, elle passe par un point d'entrée dans le premier rotor et ressort par une autre sortie qui est reliée au deuxième rotor. Le deuxième rotor est câblé d'une certaine façon et fera ressortir le courant par une autre sortie et ainsi de suite, jusqu'à arriver au réflecteur[5]. En effet, le réflecteur complexifie encore le chiffrement du caractère car il refait passer le courant une deuxième fois par les rotors selon le même principe que précédemment et va allumer une autre lettre que celle qui a été frappée à la base (21). En théorie, le chemin parcouru pourrait ressembler au schéma de la Figure 7.

Figure 7 : Exemple de chemin pouvant être parcouru par le courant électrique à travers les rotors

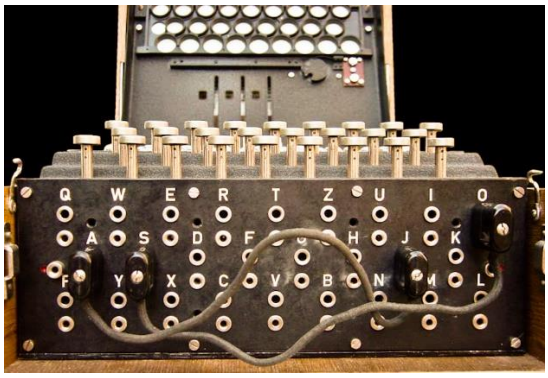


Bien que le câblage ne soit pas linéaire (si le courant passe par la première position, il ne va pas forcément ressortir par la première position), si aucun autre changement n'est effectué, le cryptage des caractères serait monoalphabétique. C'est pour cette raison que le câblage est fait dans des rotors, ainsi, à chaque frappe de clavier, le rotor de droite va se décaler d'une position. Une fois qu'il a tourné 26 fois (pour les 26 lettres de l'alphabet) et a donc effectué un tour complet, le deuxième rotor va se décaler d'un cran. Ensuite, une fois que le deuxième rotor a été décalé de 26 positions, c'est au tour du troisième de se décaler d'une position. Le mécanisme est comparable à celui d'un compteur kilométrique d'une voiture, qui comptabilise les centaines de mètres, les kilomètres, les centaines et les milliers de kilomètres (22). Pour augmenter encore la complexité de la machine, un tableau de connexion a été mis en place. Il est possible de relier une lettre à une autre pour que quand une lettre soit entrée, elle prenne la valeur

[5] Réflecteur : dernière partie de la machine Enigma, qui permet de réfléchir le courant électrique afin qu'il repasse par les rotors.

d'une autre et vice versa. Par exemple, si la lettre A est reliée à la lettre W, quand la lettre W sera rentrée, elle prendra la valeur de la lettre A alors que quand la lettre A sera tapée, elle prendra la valeur de la lettre W.

Tableau 6 : Tableau de connexions



Source : http://en.wikipedia.org/wiki/Enigma_machine#mediaviewer/File:Enigma-plugboard.jpg

Pour qu'un message puisse être chiffré et déchiffré, les machines émettrices et réceptrices doivent avoir strictement la même configuration (Voir Figure 7, si la lettre A est saisie, elle va renvoyer D alors que si D est saisi, elle va renvoyer A). En effet, les rotors peuvent être inter changés, et leur position initiale peut ne pas être à 1. C'est pour cette raison que le tableau de connexions doit aussi être configuré de la même façon sur chaque machine. Le câblage à l'intérieur des rotors varie. Dans la version militaire Allemande (Enigma 1), les rotors étaient configurés de la façon suivante (Tableau 7) :

Tableau 7 : Tableau de correspondance de caractères des rotors

Rotor	Correspondance
I	EKMFLGDQVZNTOWYHXUSPAIBRCJ
II	AJDKSIRUXBLHWTMCQGZNPYFVOE
III	BDFHJLCPRTXVZNYEIWGAKMUSQO
IV	ESOVPZJAYQUIRHXLNFTGKDCMWB
V	VZBRGITYUPSDNHLXAWMJQOFECK

Source : (23)

Initialement, seulement les rotors I, II et III avaient été prévus et les rotors IV et V ont été introduits plus tard, afin de complexifier le cassage d'un code. (23)

Avec toutes ces possibilités de configuration, cela représente 60 (5 x 4 x 3) possibilités de combinaisons avec les rotors. Le premier rotor est sélectionné parmi les 5 (voir tableau 9), ensuite le deuxième parmi les 4 restants et le troisième parmi les 3 restants.

Ensuite, les rotors peuvent être positionnés sur une des 26 possibilités qu'offre chaque rotor. Ce qui donne un total de 17576 (26 x 26 x 26) possibilités. Pour finir, le tableau de connexions rend le code indéchiffrable (en théorie). En effet, sachant qu'il y avait dix câbles qui reliaient vingt lettres entre elles (et que donc six lettres ne changeaient pas), cela permettait de faire 150'738'274'937'250 ($\frac{26!}{10! \times 6! \times 2^{10}}$) possibilités supplémentaires ce qui portait le nombre de possibilités totales à environ 1.59 x 10^{20} (60 x 17576 x 150'738'274'937'250) possibilités (21).

Malgré la complexité du chiffrement, les données ont pu être décryptées par une bombe électromécanique[6], car la machine Enigma comportait une faille, qui au premier abord semblait ne pas avoir d'importance. En effet, comme énoncé au début de ce chapitre, une lettre ne peut pas être chiffrée par elle-même. Si un opérateur tapait la lettre A, n'importe quel autre caractère aurait pu ressortir, sauf le A. C'est donc de cette façon qu'Alan Turing a pu déchiffrer des messages. Les opérateurs allemands envoyaient un rapport météorologique journalier à 6h du matin, qui comportait en en-tête « Wetterbericht » (qui signifie en allemand bulletin météorologique) et signaient le message par « Heil Hitler ». En partant de l'idée qu'une lettre ne peut pas être remplacée par elle-même, Turing et son équipe tentaient de placer ces termes dans un message intercepté, à un endroit où aucune lettre ne correspondait (24), comme dans les Tableau 8 et Tableau 9.

Tableau 8 : Exemple de message intercepté comparé à « Wetterbericht »

Message chiffré	A	E	T	V	K	L	Z	U	D	F	X	Y	A	A	L	M	J	N
Mot	W	E	T	T	E	R	B	E	R	I	C	H	T					

Tableau 9 : Exemple de message intercepté qui n'a aucun caractère en commun

Message chiffré	A	E	T	V	K	L	Z	U	D	F	X	Y	A	A	L	M	J	N
Mot				W	E	T	T	E	R	B	E	R	I	C	H	T		

Grâce à cette découverte, un message qui aurait pris plusieurs millions d'années à être déchiffré a pu l'être en moins de vingt minutes (24).

[6] Machine permettant de tester toutes les combinaisons possibles de la machine Enigma

4.1.3.4 AES

Le chiffrement AES est le standard actuel en termes de cryptographie. Il est pour le moment indéchiffrable à moins d'utiliser une méthode de *force brute*[7]. Une clé de 128 bits est utilisée pour la version standard d'AES. Initialement, le chiffrement de Rijndael (gagnant du concours AES) prévoyait en plus des chiffrements par clés de 192 et 256 bits. Les tailles de clés différentes ne changent pas le fonctionnement de l'algorithme. La seule différence se trouve dans le nombre de fois que les quatre opérations de la deuxième phase sont réalisées. Le Tableau 10 indique le nombre d'itérations (Nr) effectuées. Ce nombre dépend du nombre de colonnes que contient la matrice contenant la clé (Nk) ainsi que de son nombre de lignes (Nb). Ainsi, dans AES 128 bits, le nombre de tours de boucle sera égal à Nr - 1. Son fonctionnement se déroule en plusieurs étapes (généralement appelés « rounds »). (13)

Tableau 10 : Tableau du nombre d'itérations par rapport à la clé

	Nk	Nb	Nr
128	4	4	10
192	6	4	12
256	8	4	14

Le round initial permet de réaliser une opération initiale de clé. Ensuite, quatre opérations sont répétées neuf fois (Voir annexe 3).

1. SubBytes

Cette opération permet de faire une substitution non-linéaire sur la matrice state[8]. Chaque octet est remplacé par un autre octet choisi dans une autre table, appelée S-Box. Cette S-Box est un tableau à deux dimensions avec 16 cases en X et en Y, ce qui représente 256 valeurs distinctes. Prenons l'exemple de la lettre A qui a comme valeur ASCII 65 soit 0100 0001 en binaire. En séparant ces 8 bits en deux groupes de 4 bits, les valeurs sont 4 et 1. Ces deux valeurs correspondent aux indices x et y de la matrice qui pointent sur la nouvelle valeur de A. (13) (25)

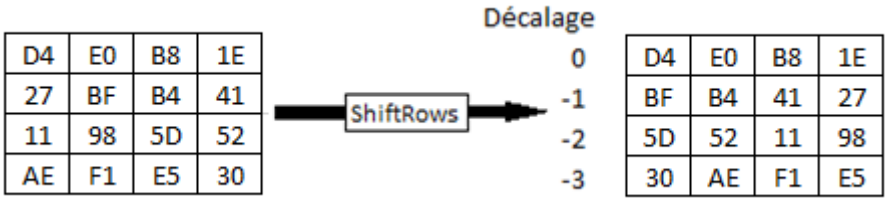
[7] Attaque par force brute : Attaque permettant de casser un code en testant toutes les possibilités.

[8] Matrice de 4x4 cases contenant un bloc de 16 octets provenant du message à chiffrer

2. ShiftRows

Dans cette étape, chaque case du tableau modifié par l'étape précédente est décalée. Si on représente les données de la matrice state sous la forme d'une matrice de 4 cases sur 4 (chaque case contenant 8 bits ce qui fait toujours un total de 128 bits), la première ligne est décalée de 0 positions vers la gauche, la deuxième ligne est décalée d'une position, la troisième ligne de deux positions et la quatrième ligne de trois positions comme dans la Figure 8 ci-dessous. (13) (25)

Figure 8 : Opération ShiftRows



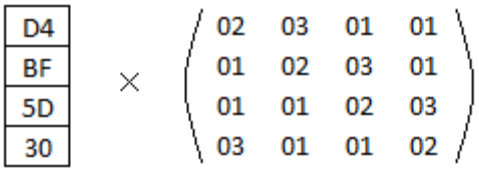
Source : (26)

3. MixColumns

Cette étape calcule le produit matriciel entre chaque colonne de la matrice state et une autre matrice. Mathématiquement parlant, cette autre matrice a été calculée avec des corps finis à 2^8 éléments. En prenant l'exemple de la Figure 9, la colonne de state utilisée est un polynôme a(x) de degré 3. Le polynôme $c(x) = 03x^3 + x^2 + x + 02$. Ensuite, pour calculer les nouvelles valeurs, le calcul suivant est réalisé : $a(x) * c(x) \bmod (x^4 + 1)$. (13) (25) Il est important de souligner que le modulo réalisé n'est pas là par hasard. Il permet de toujours tomber sur un nombre entre 0 et 255. Ensuite, c(x) et $(x^4 + 1)$ doivent être premiers entre eux, sinon le résultat du modulo pourrait donner 0. Si c'était le cas, les données ne pourraient pas être déchiffrées en faisant l'opération inverse. (13)

Informatiquement parlant, ces calculs sont calculés sous forme de produits matriciels. En prenant l'exemple de la Figure 9, le calcul se présenterait de la façon suivante pour obtenir la première valeur de la colonne : 2*D4 + 3*BF + 1*5D + 1*30. Ensuite le même calcul est appliqué avec la deuxième ligne de la matrice et ainsi de suite, pour ainsi donner les nouvelles valeurs. (13) (25)

Figure 9 : Calcul de MixColumns

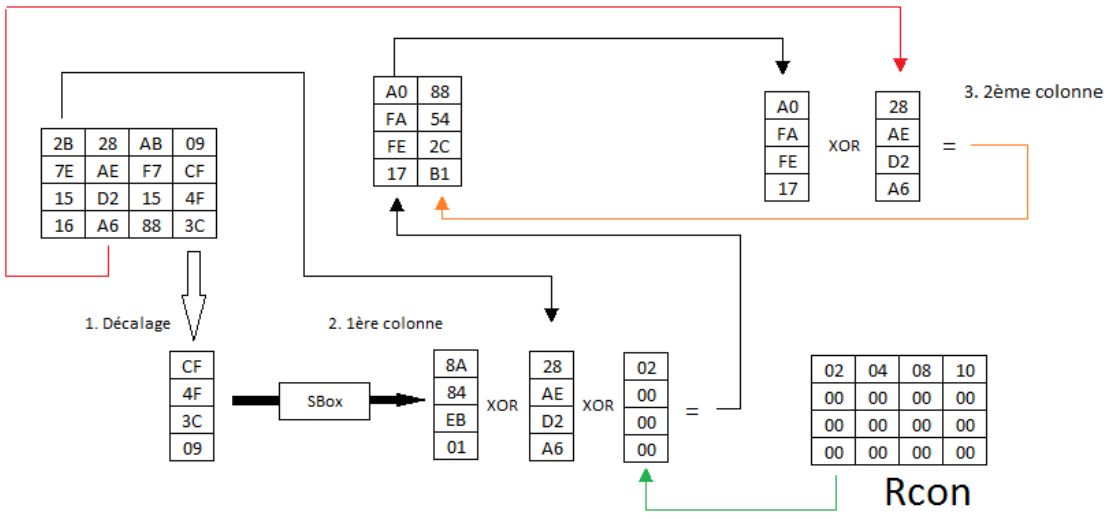


Source : (26)

KeyExpansion

Avant de passer à l'étape suivante, à savoir AddRoundKey, la clé doit être modifiée. La première colonne de la clé une fois modifiée va correspondre à la dernière colonne, décalée du bas vers le haut. Ensuite, elle va être modifiée avec la SBox, puis une opération xor va être réalisée entre le résultat obtenu, la première colonne de la clé d'origine et la colonne x (x correspond au numéro du round) de la matrice Rcon (tableau fixe de constantes de tours). Ensuite, les trois autres colonnes restantes seront des opérations xor entre la colonne de la clé d'origine et la dernière colonne ajoutée à la nouvelle clé. (13) (26)

Figure 10 : Schéma de fonctionnement de l'étape KeyExpansion



Source : (26)

4. AddRoundKey

Cette étape va réaliser une simple opération xor entre la clé et le state.
Une fois les neuf rounds réalisés, il y a un round final. Ce round reprend les mêmes quatre étapes expliquées précédemment à l'exception de l'étape « MixColumns ».
Une fois le round final terminé, le message est chiffré. (13)

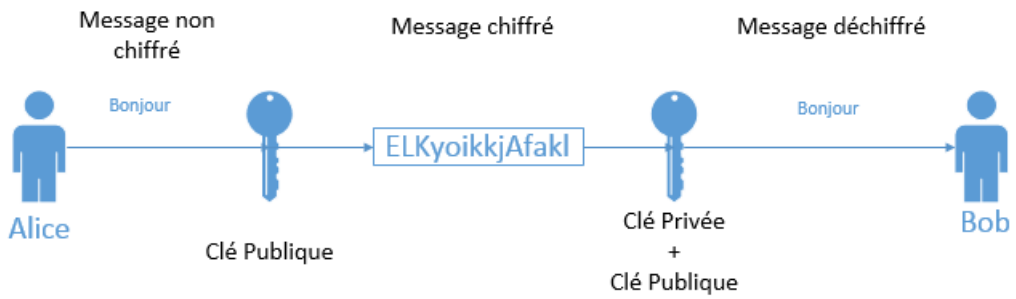
Pour déchiffrer un message, il suffit d'utiliser les fonctions inverses de chaque étape qui sont généralement nommées InvShiftRows, InvSubBytes et InvMixColumns.

4.2 Cryptographie asymétrique

La cryptographie asymétrique ou cryptographie à clé publique fonctionne de façon totalement différente à la cryptographie symétrique. Si l'on peut comparer la cryptographie symétrique à un coffre-fort auquel seules les personnes possédant la clé peuvent accéder, la cryptographie asymétrique pourrait être comparée à une boite aux lettres dans laquelle on peut déposer des informations, et seule la personne possédant la clé peut accéder au contenu de la boîte. La boite aux lettres serait la clé publique (donc accessible à tout le monde), alors que la clé pour ouvrir la boite serait la clé privée. En effet, dans la cryptographie asymétrique, il y a une clé publique et une clé privée. (27)

Les nombres premier sont l'élément clé pour rendre les algorithmes de cryptographie asymétrique indéchiffrables (ou presque). En prenant comme exemple la multiplication de 5 x 7, il est assez facile de répondre instantanément 35. L'opération inverse, à savoir, retrouver 35 à partir de facteurs premiers est assez facile. Cependant, factoriser 1591 par exemple, est beaucoup plus compliqué, alors que calculer 37 x 43 se fait très facilement. C'est sur cette difficulté de factorisation que se reposent les algorithmes asymétriques. (27)

Figure 11 : Schéma de fonctionnement de la cryptographie asymétrique



4.2.1 Algorithmes

4.2.1.1 Diffie-Hellman

Le protocole de Diffie-Hellman a été inventé par les cryptologues du même nom. Leur idée de base était la suivante : si Alice[9] veut partager une clé secrète avec Bob[10] pour pouvoir envoyer des messages chiffrés, cela pourrait se faire de façon aisée. Mais si Alice souhaite partager un message avec dix personnes, cela pourrait toujours être réalisable mais il faudrait tout de même 45 clés différentes (9+8+7+6+5+4+3+2+1 clés). Si cent personnes communiquent entre elles, les choses se compliquent car il faudrait 4950 clés au total (99+98+ …. +3+2+1 clés). C'est là que l'échange de clés de Diffie-Hellman prend tout son sens. (28)

Cet algorithme fonctionne de la façon suivante :

Alice et Bob choisissent une base **g** et un nombre premier **p**. Alice choisit un nombre secret **a**. Elle prend la base **g** et l'augmente à la puissance **a** puis fait un modulo avec le nombre premier **p**, puis elle l'envoie à Bob. Bob choisit un nombre secret **b**, réalise la même opération qu'Alice avec le nombre **b** qu'il a choisi et l'envoie à Alice. Alice n'a plus qu'à réaliser l'opération suivante pour obtenir la clé secrète : $((g^b \bmod p)^a \bmod p)$. Bob réalise la même opération qu'Alice, avec les valeurs qu'il a reçu, pour obtenir la même clé secrète, à savoir $((g^a \bmod p)^b \bmod p)$. (29)

Prenons comme exemple **g** = 37 et **p** = 43. Alice choisit comme nombre **a** = 6 et envoie à Bob 1 (le résultat de $37^6 \bmod 43$). Bob choisit comme nombre **b** = 11 et envoie donc 7 (le résultat de $37^{11} \bmod 43$). Maintenant Alice n'a plus qu'à faire l'opération suivante : $7^6 \bmod 43 = 1$ et Bob doit faire $1^{11} \bmod 43 = 1$. Alice et Bob tombent sur le résultat 1, qui est à présent leur clé secrète.

Maintenant que l'échange de clés a été réalisé, cette clé obtenue pourrait par exemple être utilisée comme clé de chiffrement pour chiffrer les données avec un chiffrement AES. Bien évidemment, pour que la clé secrète obtenue ne soit pas indéchiffrable dans un temps raisonnable, des nombres premiers beaucoup plus grands que ceux qui ont été utilisés dans l'exemple plus haut doivent être utilisés. Idéalement, il faudrait employer des nombres premiers avec plusieurs centaines de chiffres.

[9] Alice : Nom utilisé par convention lorsque l'on parle d'un émetteur

[10] Bob : Nom utilisé par convention lorsque l'on parle d'un récepteur

4.2.1.2 RSA

Le chiffrement RSA a été inventé en 1977 par les mathématiciens Ronald Rivest, Adi Shamir et Leonard Adleman. Les initiales de leur nom ont donné RSA. Même si l'idée de base est la même que celle de Diffie-Hellman, (échanger une clé avec un grand nombre de personnes), son fonctionnement est différent bien qu'il soit basé sur la difficulté à factoriser de très grands nombres premiers. Il est massivement utilisé à travers le monde. En effet, c'est entre autres le chiffrement utilisé lors de connexions sécurisées sur un navigateur Web (voir Annexe 4). Avec tous les utilisateurs du réseau Internet, il serait inimaginable d'utiliser un chiffrement symétrique. C'est pour cette raison que la clé privée est calculée avec RSA. Comme les algorithmes asymétriques sont plus lents que les symétriques, RSA ne calcule que la clé qui servira à chiffrer les données avec un chiffrement symétrique tel qu'AES.

Pour générer la clé publique qui se compose des nombres **n** et **e**, Alice doit choisir deux très grands nombres premiers distincts **p** et **q** d'une taille d'au moins 2^{512} bits chacun dans le but de former une clé de 2^{1024} bits afin d'être suffisamment sûre (30). Ensuite, elle devra calculer **n** qui est égal à **p** * **q**. Elle doit, après cela, calculer **φ(n)** = (p-1) * (q-1), qui va permettre de calculer la clé de décryptage **d**. Ensuite, elle doit choisir un exposant **e** qui est premier avec **φ(n)**. La clé publique va être constituée de **e** et **n**. Bob pourra ainsi grâce à e et n chiffrer un message m avec le calcul suivant : m^e mod n. Ensuite, pour calculer la clé de décryptage, il faut que **e** * **d** mod **φ(n)** = 1. En résumé, pour trouver **d**, il faut faire le calcul suivant : $\mathbf{e^{-1}}$ mod **φ(n)**. Pour trouver ce résultat, l'algorithme d'Euclide étendu peut être utilisé.

Prenons comme exemple p = 7, q = 11. Pour calculer n, il faut multiplier p et q, ce qui nous donne n = 77. Pour calculer **φ(n)**, il faut faire (7-1) * (11-1) ce qui donne 60. Maintenant, il faut choisir l'exposant e, qui doit être premier avec φ(n). Prenons donc par exemple e = 13 (PGCD (13, 60) = 1 donc premiers entre eux). Voici donc la clé publique e = 13 et n = 77.

Pour calculer d, il suffit de faire 13^{-1} mod 60, ce qui donne 37 (Voir Annexe 5 pour voir le détail du calcul (31)).

Pour vérifier que d est correct, il suffit de calculer e * d mod φ(n) = 1. Dans cet exemple cela revient à calculer 13 * 37 mod 60 = 1.

4.3 Hachage

Les fonctions de hachage permettent de chiffrer un message sous la forme d'une chaîne de caractères de taille fixe, peu importe la taille du message d'origine, généralement entre 128 et 512 bits. Elles sont comparables à une empreinte, car un message aura toujours la même empreinte en appliquant une fonction de hachage. Elles sont à sens unique, ce qui signifie qu'il est facile de hacher un message, mais qu'il n'est en principe pas possible de calculer son inverse, à moins d'utiliser une méthode de force brute. Pour qu'une fonction de hachage soit sûre, elle doit être résistante aux collisions. En partant du principe qu'il y a une infinité de messages possibles pouvant être chiffrés, il est évident que plusieurs messages vont donner le même résultat une fois hachés. La résistance est le fait que les collisions ne puissent pas être retrouvées. (28)

4.3.1 MD5

La fonction de hachage MD5 (Message Digest 5) a été développée par Ronald Rivest, l'un des créateurs de RSA. Elle utilise des blocs de 512 bits et génère des messages chiffrés de 128 bits. Chaque bloc est découpé en 16 sous-blocs de 32 bits (A, B, C et D). Les quatre calculs suivants sont réalisés 64 fois. (13)

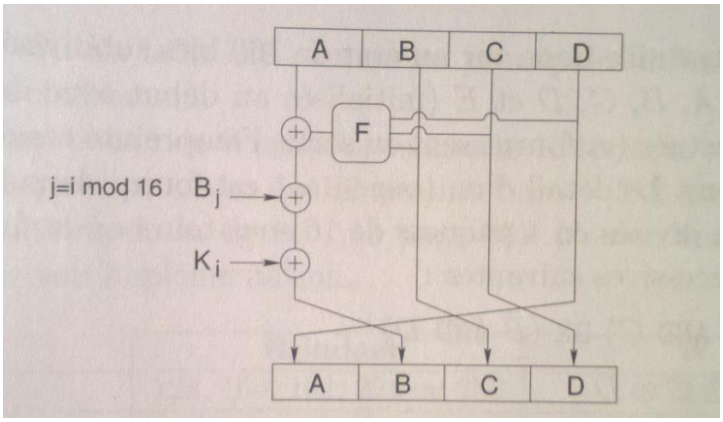
1. F = (B AND C) OR (¬(B) AND D)

2. F = (D AND B) OR (¬(D) AND C)

3. F = B XOR C XOR D

4. F= C XOR (B OR ¬(D))

Figure 12 : Schéma représentant un tour dans MD5



Source : (13)

Le résultat final obtenu donnera le message haché. MD5 est actuellement considéré comme une méthode de hachage non sûre. En réalisant une attaque par force brute, cela permet de retrouver un message original assez rapidement. La complexité du message haché est de 2^{64} calculs, or certains algorithmes permettent de réduire les calculs à 2^{30} opérations.

4.3.2 SHA-1

La fonction de hachage SHA-1 (Secure Hash Algorithm) a été développée par le NIST en 1995. SHA-1 utilise des blocs de 512 bits et produit des messages hachés de 160 bits. Les blocs sont découpés en 16 sous-blocs comme pour MD5 de 32 bits chacun. Ils sont ensuite étendus en 80 blocs. Pour produire les 80 blocs, les 4 opérations suivantes sont réalisées 80 fois.

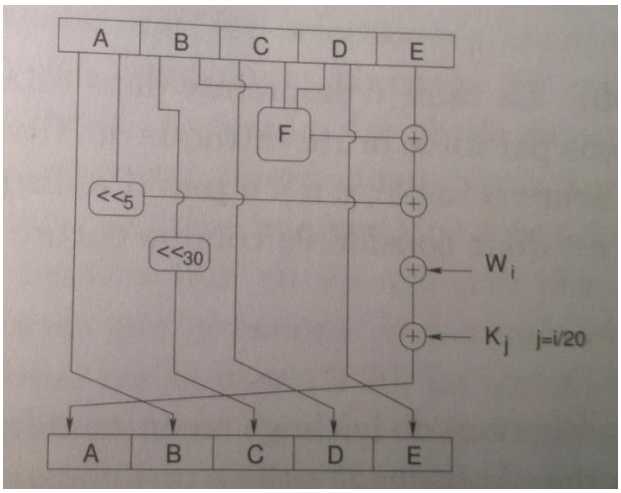
1. F = (B AND C) OR (¬(B) AND D)

2. F = B XOR C XOR D

2. F = (D AND C) XOR (B AND D) XOR (C AND D)

4. F= B XOR C XOR D

Figure 13 : Schéma représentant un tour dans SHA-1



Source : (13)

A ce jour, SHA-1 est théoriquement cassable par force brute. Toutefois, il est encore très difficile de le casser facilement. Un mot de passe simple de type « 12345 » haché avec SHA-1 pourra être retrouvé facilement. En revanche un mot de passe contenant des caractères aléatoires prendra beaucoup plus de temps à être retrouvé. Malgré cela, il reste très utilisé. (13)

4.4 Cryptographie quantique

La cryptographie quantique se base sur les propriétés de la mécanique quantique. Plutôt que d'essayer de chiffrer des données afin qu'elles soient incompréhensibles, la communication est arrêtée si un message a été intercepté. C'est donc de cette façon que fonctionne la cryptographie quantique. Selon le principe d'incertitude d'Heisenberg, si un objet quantique (dans ce cas, un photon circulant dans une fibre optique) a été observé alors il a été modifié et donc la communication est coupée. (32)

Le papier « La cyptographie quantique en bref » (32) résume ce type de cryptage avec un exemple : un joueur de tennis lance avec sa raquette, des balles contenant chacune un message. Si une personne souhaite intercepter les balles, elle pourrait le faire. Mais si au lieu de lancer des balles, le joueur de tennis lançait des bulles en savon ? Dans ce cas-là, si quelqu'un tente de les intercepter, elles exploseraient sur le coup car elles sont fragiles.

Figure 14 : Illustration du fonctionnement de la cryptographie quantique

Source : (32)

5. Les algorithmes en pratique

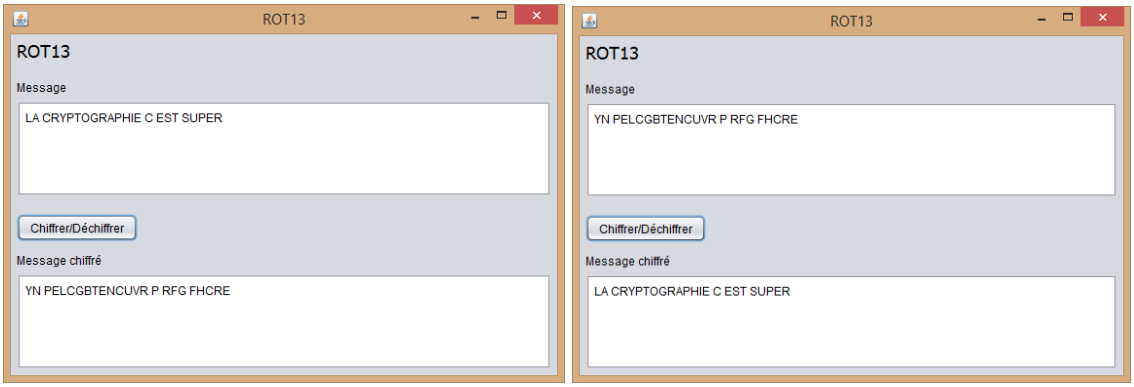
5.1 Présentation du logiciel

Le logiciel se présente sous forme de plusieurs petits projets, réalisés en Java. Ils mettent en pratique les différents algorithmes étudiés précédemment.

5.2 Les différents algorithmes utilisés

5.2.1 ROT13

Figure 15 : Interfaces graphiques pour le chiffrement/déchiffrement (ROT13)

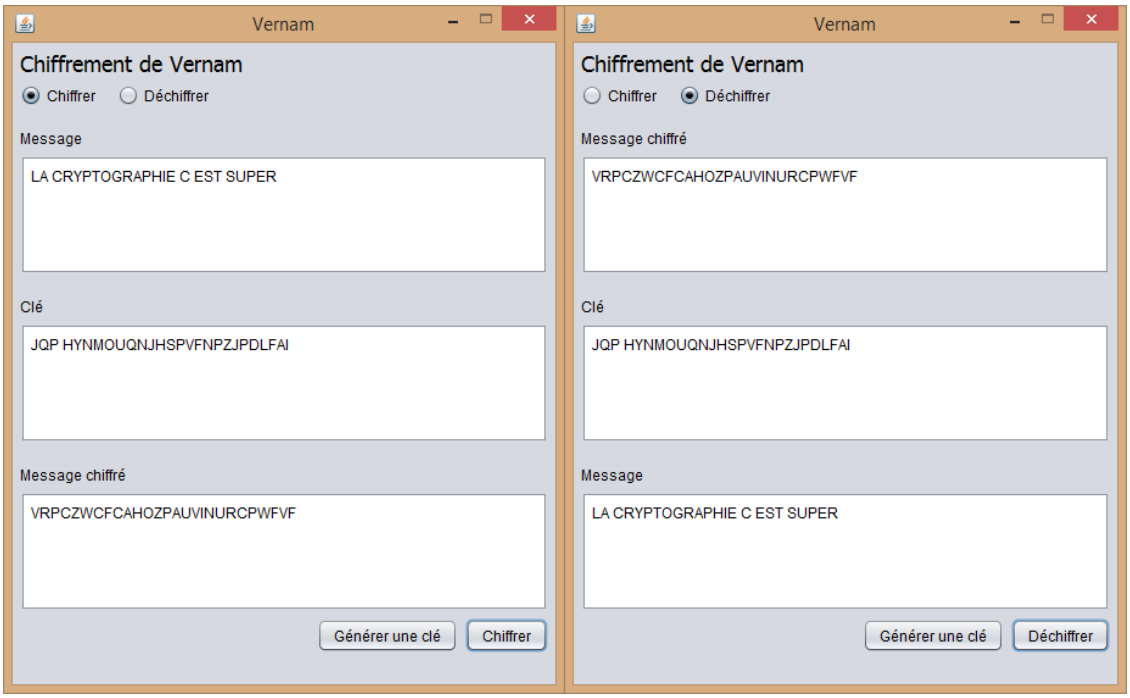


L'application de chiffrement ROT13 contient deux champs textes. Le premier permet de rentrer le texte reçu (chiffré ou original) et va l'afficher sous sa forme inverse : s'il est chiffré, il sera déchiffré alors que s'il est déchiffré il sera chiffré.

Pour mettre en place l'algorithme, la formule décrite dans le chapitre sur le chiffrement ROT13 a été utilisée. Toutefois, quelques traitements préalables ont été effectués. Une fois que le bouton « Chiffrer / Déchiffrer » a été cliqué, la chaîne de caractères va être tout d'abord transformée en lettres majuscules, pour éviter toute ambiguïté. Ensuite, les accents vont être enlevés. Ces traitements sont nécessaires, car cet algorithme ne fonctionne qu'avec les 26 lettres de l'alphabet.

5.2.2 Chiffrement de Vernam

Figure 16 : Interfaces graphiques pour le chiffrement/déchiffrement (Vernam)



L'application permet de chiffrer et de déchiffrer un message. Dans les deux cas, il fonctionne de la même façon. Il suffit d'indiquer si le message va être chiffré ou déchiffré dans la radio bouton en dessous du titre. Une première zone de texte permet de rentrer le message d'origine. Dans le cas d'un chiffrement, le message non codé, et dans le cas d'un déchiffrement, le message codé. Puis, la deuxième zone de texte sert à renseigner la clé. Il est à noter qu'un générateur de clés pseudo aléatoires[11] a été développé. Il génère une clé contenant le même nombre de caractères que le message d'origine et il peut être utilisé via le bouton « Générer une clé » tout en bas de l'interface.

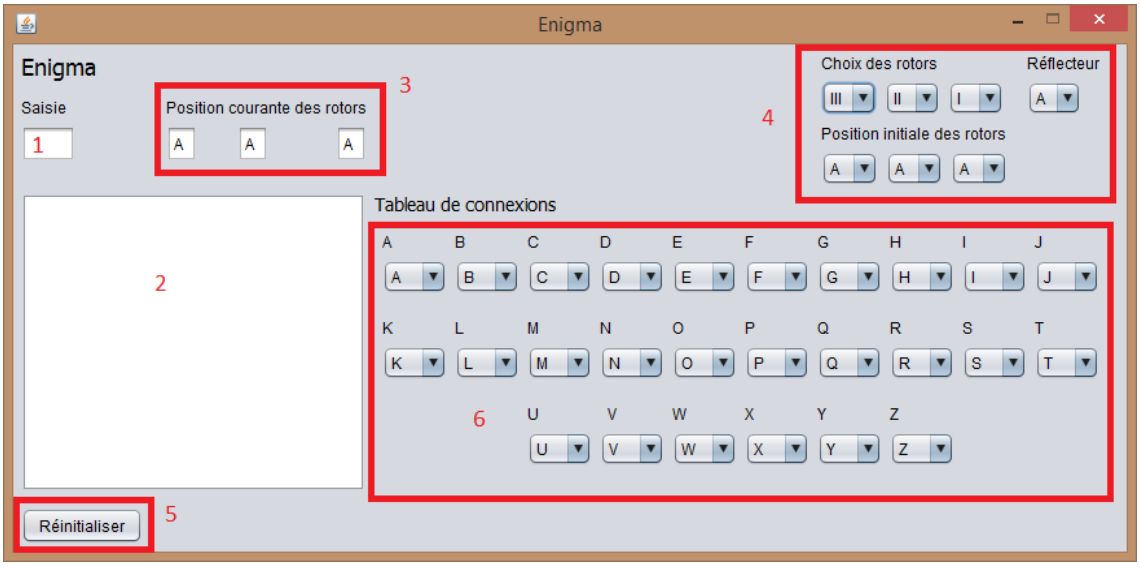
Concernant la mise en place des algorithmes de chiffrage et déchiffrage, les recherches effectuées lors de l'analyse de l'algorithme ont été mises en place ; soit deux fonctions servant à chiffrer et déchiffrer. Pour reproduire fidèlement ce qui a été expliqué, certains traitements ont dû être effectués sur les messages. Afin de faciliter le fonctionnement du programme et pour qu'il n'y ait aucune ambiguïté, tous les caractères sont automatiquement convertis en majuscules. Ensuite, le chiffrement et déchiffrement ne traitent que les caractères entre A et Z. Si un autre caractère est entré, il n'est pas pris en compte mais est quand même affiché.

[11] Une clé aléatoire est impossible à réaliser avec une simple fonction « random »

5.2.3 Enigma

Enigma étant une machine physique, un simulateur de la machine a été réalisé. Il permet d'utiliser la version allemande utilisée pendant la 2ème Guerre Mondiale avec cinq rotors à disposition dont trois sont utilisables. Les réflecteurs A, B et C (Tableau 7) ont eux aussi été implémentés. L'interface se présente de la façon suivante :

Figure 17 : Simulateur de la machine Enigma



Une zone de texte permet de saisir un caractère à la fois [1] et est automatiquement affiché dans la grande zone de texte en dessous [2]. La position courante des rotors est affichée instantanément lors d'une saisie [3]. Dans la partie en haut à droite de l'interface [4], les réglages du choix des rotors, du réflecteur ainsi que des positions initiales des rotors s'effectuent à cet endroit. En dessous [6], se trouve le tableau de connexions, qui permet d'intervertir un caractère par un autre. Si par exemple la lettre B est intervertie avec la lettre P, le champ correspondant à la lettre P se modifiera automatiquement avec la lettre B. A chaque modification de configuration, il faut réinitialiser la machine [5] pour que les nouveaux paramètres soient pris en compte.

Du point de vue du code Java, une classe « Rotor » a été créée et elle simule le comportement d'un rotor via ses différentes méthodes. Au moment de l'initialisation d'un rotor, le constructeur reçoit comme paramètres un tableau de nombres entiers compris entre 0 et 25[12] (A correspondant à 0, B à 1 et ainsi de suite) ainsi que la position initiale

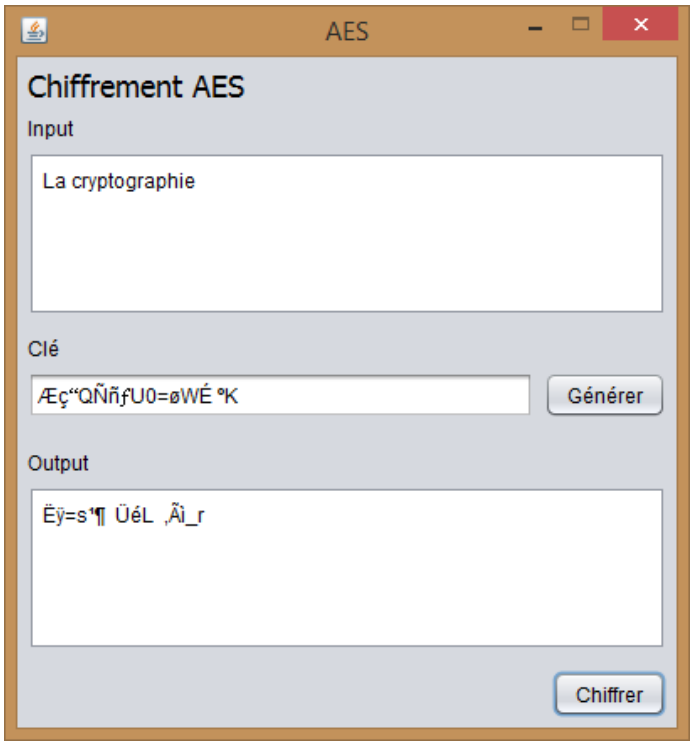
[12] Selon le Tableau 7

du rotor, configurée par l'utilisateur ([4] sur la Figure 17). Pour simuler le fonctionnement de l'intérieur de rotors, deux tableaux ont été mis en place. Le premier contient les chiffres de 0 à 25 (c'est la lettre qui sera affichée dans la position des rotors [3]) et le deuxième, contient le tableau reçu en paramètre lors de l'initialisation du rotor. Ainsi, l'indice 0 du tableau d'entrée contiendra par exemple la lettre A et l'indice 0 du tableau de sortie contiendra par exemple K. Au moment où il y a un décalage, les deux indices sont décalés d'une case dans le tableau (de façon circulaire), ainsi, peu importe le nombre de décalages qu'il y a, ce couple de lettres sera toujours le même, seul l'indice pour y accéder changera.

Pour faire le premier parcours depuis le clavier jusqu'au réflecteur, une méthode getSortie() retourne la position de sortie pour pouvoir accéder au rotor suivant. Une fois arrivé au réflecteur (représenté par la classe « Reflector » et qui ne fait que renvoyer l'indice du tableau du troisième rotor), c'est la méthode getEntree() qui renvoie l'entrée du réflecteur. Cette fois le traitement est différent : une boucle parcourt le tableau des entrées et cherche la valeur correspondante pour pouvoir rentrer dans le rotor suivant. Pour finir, une fois arrivée au premier rotor, la méthode getChrFinal() retourne l'indice de sortie et non le contenu du tableau d'entrée. De cette façon, en supposant que le nombre soit 5, cela correspondra à la lettre F (F étant la cinquième lettre de l'alphabet, en partant de 0).

5.2.4 AES

Figure 18 : Interface graphique pour le chiffrement AES



L'application permet de chiffrer un message avec le chiffrement AES (128 bits). Il est possible d'entrer un texte dans la partie Input. Comme la clé est une chaîne de 128 bits aléatoires, un générateur de nombres pseudo aléatoires a été mis en place. Ce qui apparaît dans la zone de texte est l'équivalent en valeur ASCII de chaque nombre généré. Comme AES est un chiffrement par blocs, chaque bloc chiffré doit avoir 16 octets (chaque lettre est codée sur 8 bits, ce qui donne un total de 128 bits). Si le nombre de caractères entré ne permet pas d'avoir le dernier bloc restant de 16 caractères, une fonction a été codée afin d'ajouter x nombres de fois le caractère « 0 » pour compléter le dernier bloc à chiffrer. Afin de traiter chaque bloc séparément, au moment où le bouton « Chiffrer » est cliqué, le message rentré est découpé en blocs. Chaque bloc est instancié avec une classe bloc qui contient un tableau de 16 caractères, puis est stocké dans un ArrayList afin de pouvoir appliquer l'algorithme AES sur chaque bloc facilement.

Bien que dans les exemples donnés dans la partie AES de ce travail, le message, la clé, et toutes les autres matrices étaient représentées sous formes de matrices (visuellement cela représente un tableau à deux dimensions), les différents tableaux ont été

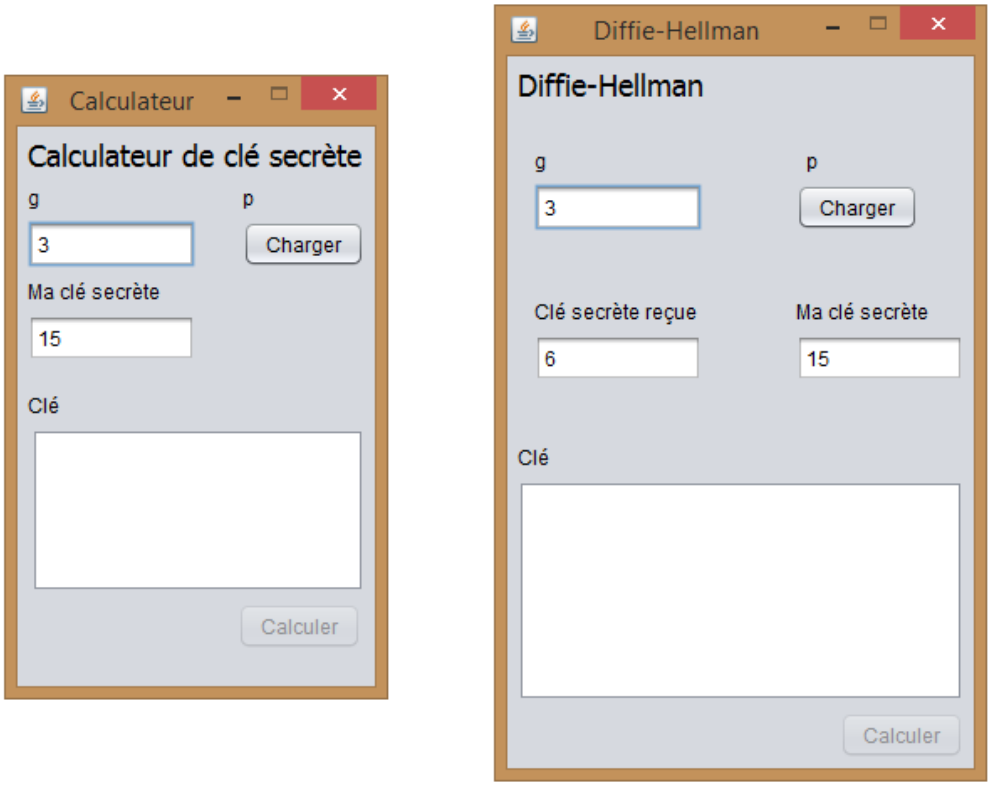
implémentés sous la forme de tableaux à une dimension. Cela n'influe pas sur le résultat final.

En reprenant l'exemple de l'étape « SubBytes », la lettre A correspond à la valeur 41 en hexadécimal (donc la valeur de la [4ème ligne ; 1ère colonne]). Sa valeur décimale est 65, or 41 hexadécimal vaut 65 décimal. Finalement, le résultat est le même. En voulant accéder à la 65ème case du tableau à une dimension, cela donnera le même résultat que si un tableau à deux dimensions avait été utilisé. Toutefois, si la manière pédagogique expliquée dans la partie AES avait été utilisée, il aurait fallu convertir chaque valeur en hexadécimal. Ainsi, des simples Integer on suffit pour pouvoir accéder aux différentes cases des différents tableaux. Ce principe s'applique à l'accès à toutes les matrices de l'algorithme AES.

Le déchiffrage n'a pas été implémenté. Pour pouvoir déchiffrer un message, il suffirait de coder les matrices inverses nécessaires au déchiffrement. En principe, comme AES se base sur des fonctions inversibles, en faisant les mêmes opérations que pour le chiffrement mais avec les matrices inversées, le déchiffrement devrait retourner le message d'origine.

5.2.5 Diffie-Hellman

Figure 19 : Interfaces pour le protocole d'échange de clés Diffie-Hellman



Cette application a été réalisée en deux parties. La première (la fenêtre de gauche dans la Figure 19) permet de calculer la clé publique calculée au moyen de la clé qui va être envoyée à la personne avec qui l'ont souhaite communiquer. Dans la deuxième fenêtre (la fenêtre de droite dans la Figure 19) se trouve l'application qui permet de calculer la clé privée finale, grâce à la clé secrète reçue.

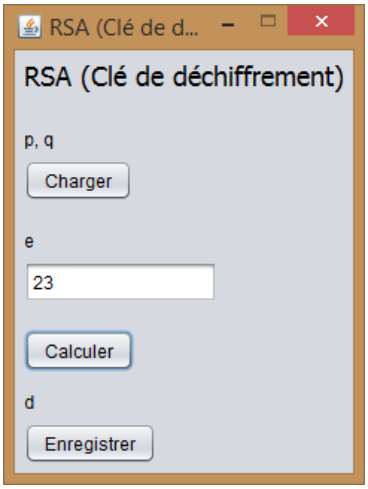
Les deux fenêtres ont un bouton qui permet de charger le nombre premier p. En effet, pour que le chiffrement soit efficace, ce nombre premier est composé de plusieurs centaines de milliers de chiffres. Il serait compliqué de tous les taper à la main dans une zone de texte. C'est pourquoi, le nombre premier se trouve dans un fichier « prime.txt »[13], qui se trouve dans le même dossier que le fichier exécutable.

Comme cette application est amenée à utiliser de très grands nombres premiers, les variables de type Integer ou encore Long n'étaient pas assez grandes. Afin de calculer ce que Bob reçoit et le résultat final. Des BigInteger ont été utilisés, qui proviennent de la librairie java.math.BigInteger.

[13] Ces nombres premiers proviennent du site Internet du GIMPS (28)

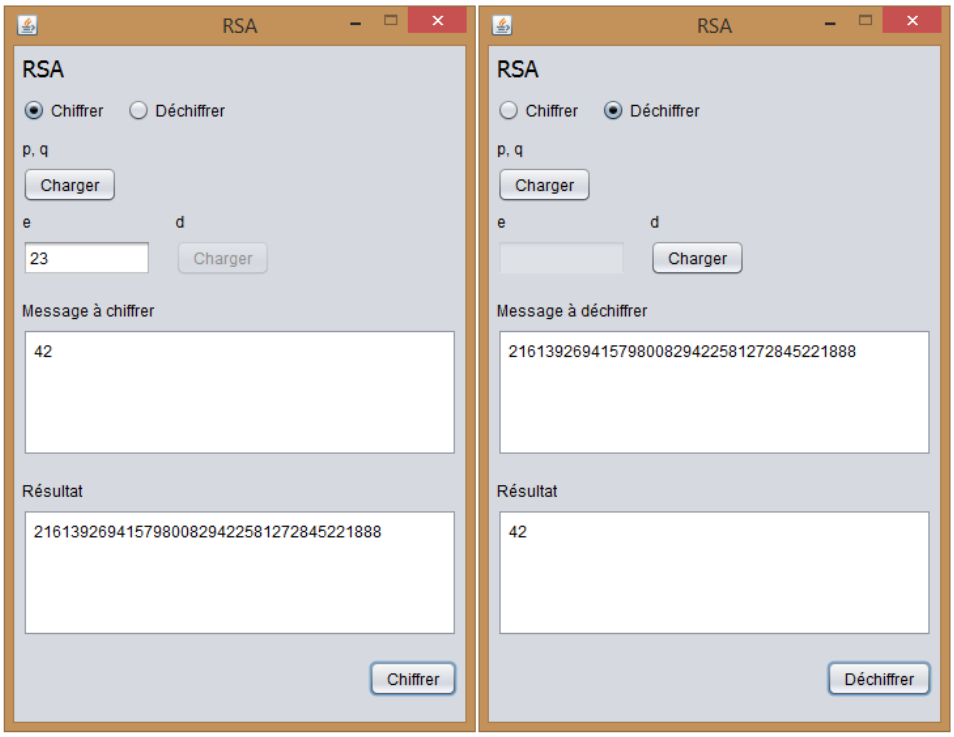
5.2.6 RSA

Figure 20 : Génération de clé de décryptage



L'application RSA se décompose comme pour Diffie-Hellman en deux parties. La première partie permet de générer la clé pour déchiffrer un message. Tout d'abord, les clés privées p et q doivent être chargées en cliquant sur le bouton « Charger ». Une zone de texte plus bas permet de rentrer l'exposant e. Pour calculer d, il suffit de cliquer sur le bouton « Calculer ». Une fois le calcul réalisé, le bouton enregistrer devient actif. En cliquant dessus, un fichier texte « d.txt » est créé (ou écrasé si une autre version existait auparavant) à l'emplacement du programme, avec la clé de décryptage.

Figure 21 : Chiffrement et déchiffrement de messages



La seconde application permet de chiffrer et de déchiffrer des messages. Il y a deux boutons radio permettant à l'utilisateur de choisir s'il souhaite chiffrer ou déchiffrer un message. En fonction de son choix, les différents labels des différents composants sont

modifiés afin qu'ils soient cohérants avec le choix de l'utilisateur. Dans le cas du chiffrement, l'utilisateur doit charger les nombres premiers p et q qui se trouvent dans les fichiers « prime1.txt » et « prime2.txt » dans le même dossier que le fichier exécutable du programme. Ensuite, il doit entrer l'exposant e (le même exposant e utilisé lors de la création de la clé de déchiffrement d expliqué précédemment). Après cela, il ne manque plus qu'à taper un message à chiffrer dans la zone de texte correspondante puis à cliquer sur le bouton « Chiffrer ». Le message chiffré apparaîtra dans la zone de texte « Résultat ».

Pour le déchiffrement, le fonctionnement est le même, sauf pour la clé de déchiffrement d. En effet, les clés de déchiffrements sont généralement très longues, donc le fichier « d.txt » créé précédemment va être chargé au moment où l'utilisateur clique sur le bouton « Charger ». Ensuite le déroulement est le même que pour le chiffrement. Une fois le bouton « Déchiffrer » cliqué, le résultat apparaît dans la zone de texte « Résultat ».

Afin de gérer au mieux ces opérations, une classe D a été créée dans les deux applications. Elle varie un peu d'une application à l'autre, mais elle s'occupe dans les deux cas la gestion de la clé de déchiffrage. Une classe E a été aussi implantée dans la deuxième application afin de gérer la gestion du chiffrement d'un message.

Comme pour Diffie-Hellman, de très grands nombres premiers sont utilisés (les nombres p et q et les résultats calculés à partir d'eux), des BigInteger (java.math.BigInteger) sont utilisés.

6. Conclusion

La complexité des algorithmes de cryptage n'a cessé d'augmenter à travers les âges. A ses débuts, un bâton permettait de chiffrer des données. Bien qu'actuellement ce procédé paraisse rudimentaire, il fonctionnait très bien car à cette époque, beaucoup de personnes étaient illettrées (ce qui était déjà une barrière pour déchiffrer un message codé) et la cryptographie était une science non connue. Par la suite, des méthodes manipulant les caractères d'un message ont commencé à prendre place. D'abord avec des chiffrements monoalphabétiques et par la suite avec des chiffrements polyalphabétiques. Toutefois, bien que certaines méthodes polyalphabétiques soient jugées encore sûres de nos jours, la cryptographie a pris un sérieux tournant avec le début de la technologie. Les décalages de caractères des anciennes méthodes ont laissé place à des méthodes ingénieuses combinant les caractères dans tous les sens. Toutefois, certaines de ces méthodes modernes ont atteint leurs limites. Le code Enigma paraissait indéchiffrable et pourtant il a été déchiffré. Le chiffrement DES a aussi connu le même sort. Il paraissait indéchiffrable et pourtant le NIST a dû lancer le concours AES afin de trouver un successeur suffisamment sûr.

La technologie étant de plus en plus présente, les experts en cryptographie ont réfléchi à une nouvelle manière de pouvoir communiquer des clés de chiffrement. En effet, transmettre une clé à une personne peut être assez facile. Transmettre une clé à beaucoup de personnes est plus compliqué. De plus, les messages peuvent être interceptés. C'est pour cette raison que des protocoles tels que Diffie-Hellman ont fait leur apparition. Il est à noter que contrairement aux autres algorithmes symétriques cités plus haut dans cette conclusion, cette nouvelle façon, asymétrique, de chiffrer les données a été dès le départ beaucoup plus complexe que les scytales jadis utilisées. Ces cryptologues avaient déjà la connaissance du domaine et ont cherché à combler le problème de transmission de clés à grande échelle.

Actuellement, AES et RSA sont des standards en termes de cryptographie. Cependant, bien qu'ils soient jugés sûrs, l'évolution de la cryptographie a prouvé que les algorithmes ont toujours fini par atteindre leurs limites avec l'évolution de la technologie. Donc finalement, si la technologie permet d'augmenter la complexité des méthodes de chiffrement, mais que parallèlement elle permet de casser des messages, est-ce que la technologie est-elle sa propre ennemie ?

7. Bilan personnel

D'un point de vue personnel, j'ai trouvé que ce travail de Bachelor était très intéressant à réaliser. Après avoir passé de longues heures à rechercher et comprendre les différents algorithmes expliqués au long de ce travail, puis finalement à les débugger, j'ai compris que certains algorithmes tels qu'AES pourraient faire l'objet d'un travail de recherche à eux tous seuls, au vu de leur difficulté. Malgré cela, j'ai eu beaucoup de plaisir à écrire ce mémoire car je pense que la cryptographie est un sujet très captivant. En effet, je n'ai pas cessé d'être étonné de l'ingéniosité de certains algorithmes de cryptage lors de la rédaction.

Bibliographie

1. Le scandale WikiLeaks pousse le renseignement américain à se réformer. monde.fr, Le. 27 01 2012.

2. Edward Snowden. Wikipédia, L'encyclopédie libre. [En ligne] http://fr.wikipedia.org/wiki/Edward_Snowden.

3. Sony Pictures victime d'un «vol de données confidentielles». ATS/Newsnet. 04 12 2014.

4. Cryptographie. Wikitionary. [En ligne] [Citation : 26 Janvier 2015.] http://fr.wiktionary.org/wiki/cryptographie.

5. cryptographie. Larousse. [En ligne] [Citation : 26 Janvier 2015.] http://www.larousse.fr/dictionnaires/francais/cryptographie/20864?q=cryptographie#20742.

6. Histoire de la cryptologie. Wikipédia, L'encyclopédie libre. [En ligne] [Citation : 29 Janvier 2015.] http://fr.wikipedia.org/wiki/Histoire_de_la_cryptologie.

7. Petite histoire de la cryptologie. Apprendre-en-ligne. [En ligne] [Citation : 27 Janvier 2015.] http://www.apprendre-en-ligne.net/crypto/histoire/.

8. Scytale. Wikipédia l'encycplopédie libre. [En ligne] [Citation : 27 Janvier 2015.] http://fr.wikipedia.org/wiki/Scytale.

9. Le chiffre de César. La cryptographie expliquée. [En ligne] [Citation : 26 Janvier 2015.] http://www.bibmath.net/crypto/index.php?action=affiche&quoi=substi/cesar.

10. Le chiffre de Vigenère. La cryptographie expliquée. [En ligne] [Citation : 2 Février 2015.] http://www.bibmath.net/crypto/index.php?action=affiche&quoi=poly/vigenere.

11. L'affaire du télégramme de Zimmermann - Cryptographie et diplomatie. La cryptographie expliquée. [En ligne] [Citation : 4 Février 2015.] http://www.bibmath.net/crypto/index.php?action=affiche&quoi=debvingt/zimmermann.

12. Breaking Germany's Enigma Code. BBC. [En ligne] 17 Février 2011. [Citation : 09 03 2015.] http://www.bbc.co.uk/history/worldwars/wwtwo/enigma_01.shtml.

13. Jean-Guillaume Dumas, Jean-Louis Roch, Eric Tannier, Sébastien Varrette. Théorie des codes, Compression, cryptage, correction. Paris : Dunod, 2013. 978-2-10-059911-0.

14. Whitfield Diffie, Martin Hellman. New Directions in Cryptography. 6 Novembre 1976.

15. Cryptographie symétrique. Wikipédia, L'encylopédie libre. [En ligne] [Citation : 03 Février 2015.] http://fr.wikipedia.org/wiki/Cryptographie_sym%C3%A9trique.

16. Vaudenay, Serge. La fracture cryptographique. Lausanne : Presses polytechniques et universitaires romandes, 2011. 978-2-88074-830-2.

17. ROT13. Wikipédia, L'encyclopédie libre. [En ligne] [Citation : 5 Février 2015.] http://fr.wikipedia.org/wiki/ROT13.

18. Chiffrement par substitution. CommentCaMarche.net. [En ligne] [Citation : 4 Février 2015.]

19. Fréquence d'apparition des lettres en français. Wikipédia, L'encyclopédie libre. [En ligne] [Citation : 13 Février 2015.] http://fr.wikipedia.org/wiki/Fr%C3%A9quence_d%27apparition_des_lettres_en_fran%C3%A7ais.

20. La machine Enigma. apprendre-en-ligne. [En ligne] [Citation : 16 Janvier 2015.] http://www.apprendre-en-ligne.net/crypto/Enigma/.

21. Enigma (Machine). Wikipédia, L'encyclopédie libre. [En ligne] [Citation : 15 Février 2015.] http://fr.wikipedia.org/wiki/Enigma_%28machine%29.

22. The Enigma Machine Explained. [Youtube]

23. Enigma Rotor Details. Wikipedia, The Free Encyclopedia. [En ligne] [Citation : 2 Mars 2015.] http://en.wikipedia.org/wiki/Enigma_rotor_details.

24. Benamran, Bruce. Alan Turing - Enigma, ordinateur et pomme empoisonnée - LPPV.05 - e-penser. [Youtube] 2015.

25. Paar, Christof. Lecture 8: Advanced Encryption Standard (AES) by Christof Paar.

26. AES Rijndael Cipher - Visualization. Youtube. [En ligne] https://www.youtube.com/watch?v=mlzxpkdXP58.

27. Cryptographie - partie 4 : cryptographie à clé publique .

28. Niels Ferguson, Bruce Schneier. Cryptographie en pratique. Paris : Vuibert, 2004. 2-7117-4820-0.

29. Paar, Christof. Lecture 13: Diffie-Hellman Key Exchange and the Discrete Log Problem by Christof Paar .

30. —. Lecture 12: The RSA Cryptosystem and Efficient Exponentiation by Christof Paar.

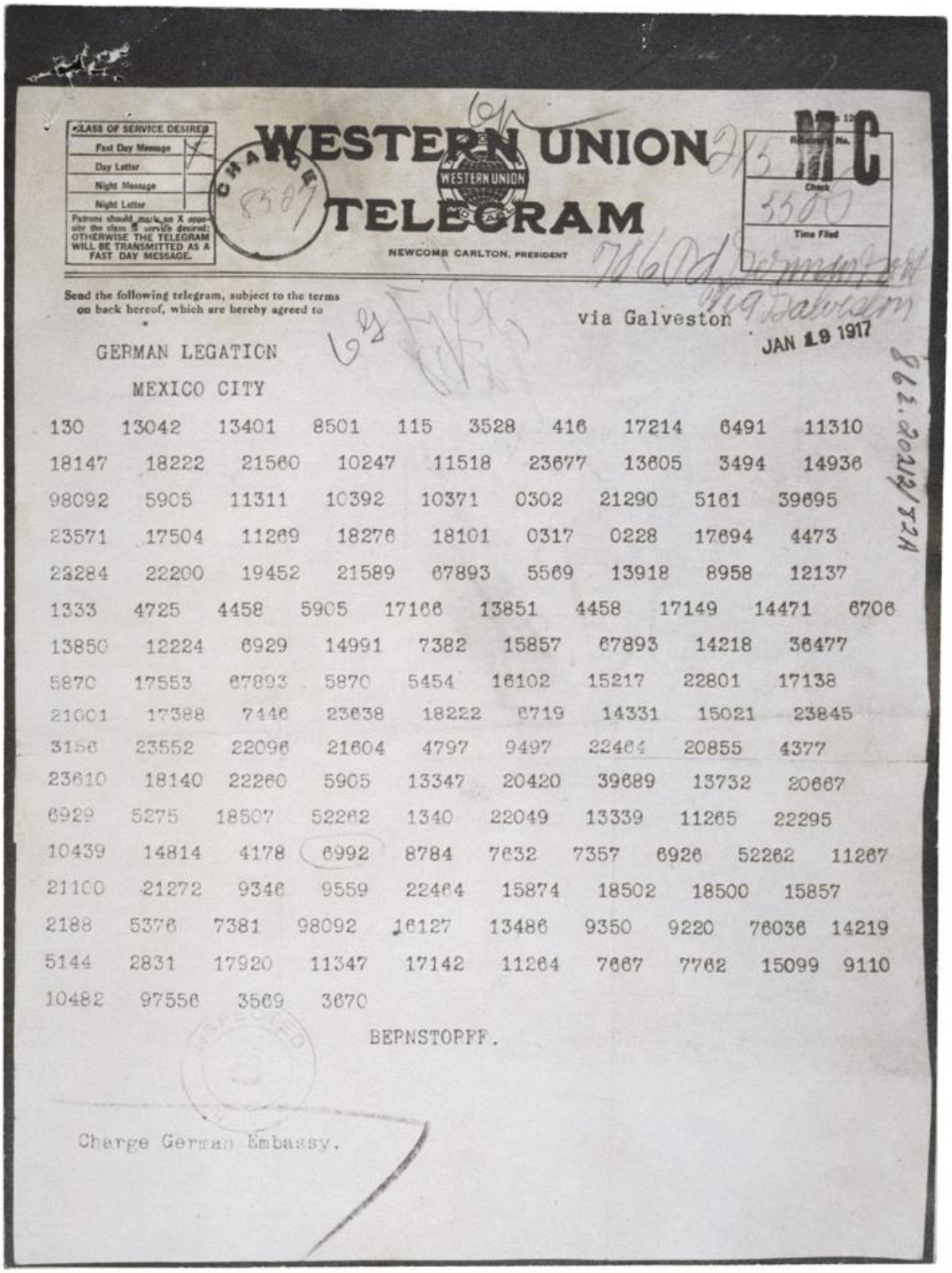
31. Sri-Jayantha, Darren. Princeton University. Modular Inversion. [En ligne] http://www.cs.princeton.edu/~dsri/modular-inversion-answer.php.

32. La cryptographie quantique en bref. Genève, Université de.

33. Thawte. Histoire du chiffrement et de ses métodes, synthèse chronologique du chiffrement à travers les âges. Thawte. [En ligne] [Citation : 27 Janvier 2015.] https://www.thawte.fr/assets/documents/guides/history-cryptography.pdf.

34. List of Known Mersenne Prime Numbers. Great Internet Mesenne Prime Search. [En ligne] http://www.mersenne.org/primes/.

Annexe 1 : Télégramme de Zimmermann



CLASS OF SERVICE DESIRED
Fast Day Message
Day Letter
Night Message
Night Letter
Patrons should mark an X opposite the class of service desired; OTHERWISE THE TELEGRAM WILL BE TRANSMITTED AS A FAST DAY MESSAGE.

CHARGE 8309

WESTERN UNION
TELEGRAM

NEWCOMB CARLTON, PRESIDENT

NC

Check 3380

Time Filed

Send the following telegram, subject to the terms on back hereof, which are hereby agreed to

via Galveston

JAN 19 1917

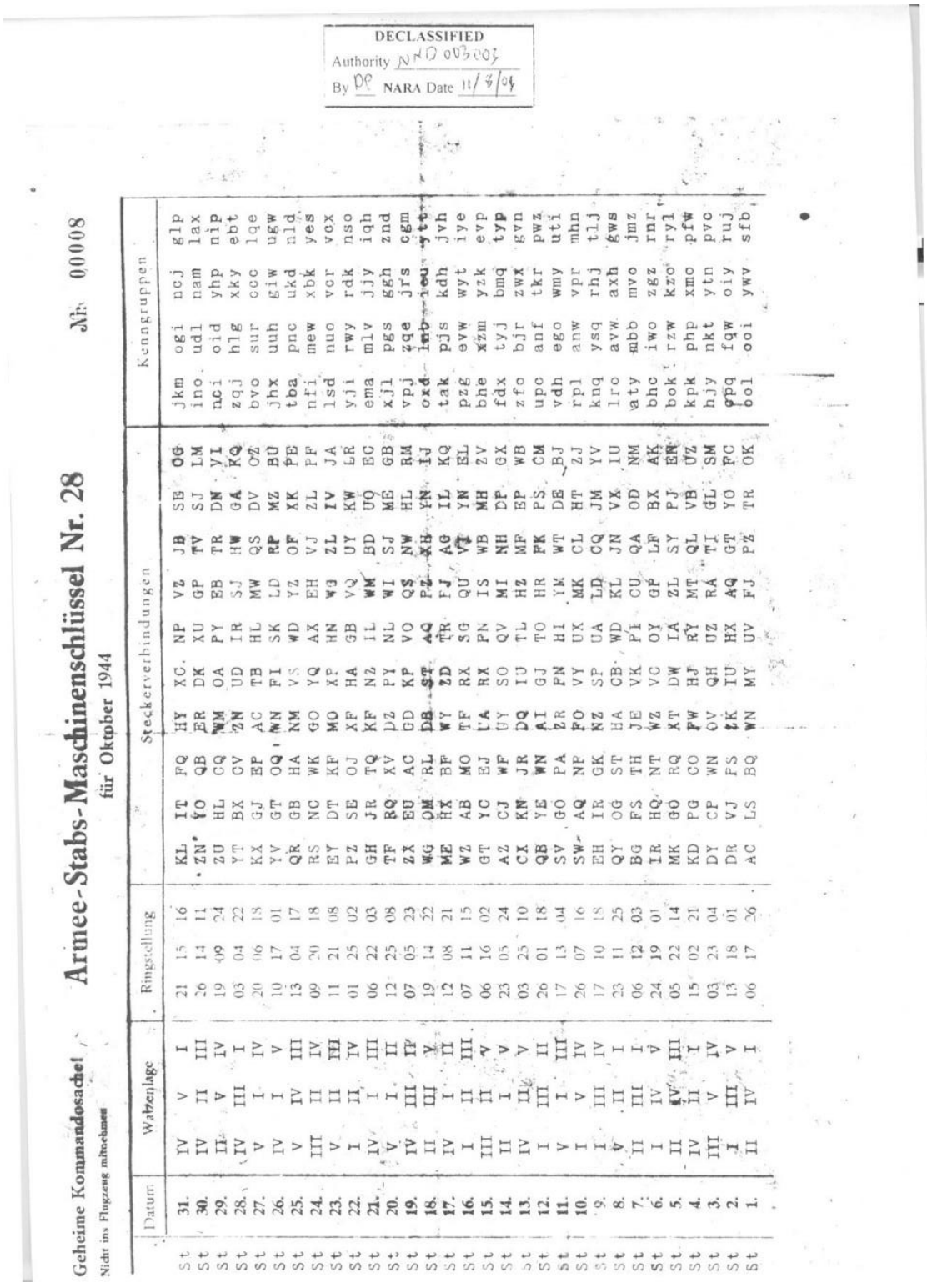
GERMAN LEGATION
MEXICO CITY

130 13042 13401 8501 115 3528 416 17214 6491 11310
18147 18222 21560 10247 11518 23677 13605 3494 14936
98092 5905 11311 10392 10371 0302 21290 5161 39695
23571 17504 11269 18276 18101 0317 0228 17694 4473
23284 22200 19452 21589 67893 5569 13918 8958 12137
1333 4725 4458 5905 17166 13851 4458 17149 14471 6706
13850 12224 6929 14991 7382 15857 67893 14218 36477
5870 17553 67893 5870 5454 16102 15217 22801 17138
21001 17388 7446 23638 18222 6719 14331 15021 23845
3156 23552 22096 21604 4797 9497 22464 20855 4377
23610 18140 22260 5905 13347 20420 39689 13732 20667
6929 5275 18507 52262 1340 22049 13339 11265 22295
10439 14814 4178 6992 8784 7632 7357 6926 52262 11267
21100 21272 9346 9559 22464 15874 18502 18500 15857
2188 5376 7381 98092 16127 13486 9350 9220 76036 14219
5144 2831 17920 11347 17142 11264 7667 7762 15099 9110
10482 97556 3569 3670

BEPNSTORFF.

Charge German Embassy.

Annexe 2 : Réglages journaliers d'une machine Enigma



DECLASSIFIED
Authority NND 003003
By DP NARA Date 11/3/04

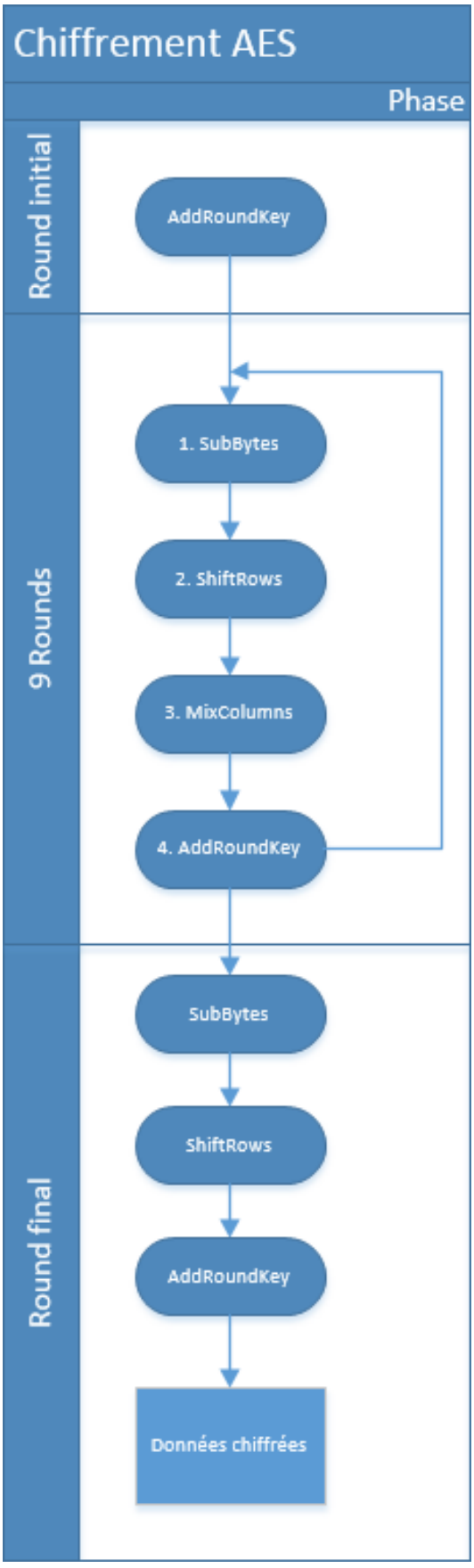
Geheime Kommandosache! Nicht ins Flugzeug mitnehmen

Armee-Stabs-Maschinenschlüssel Nr. 28
für Oktober 1944

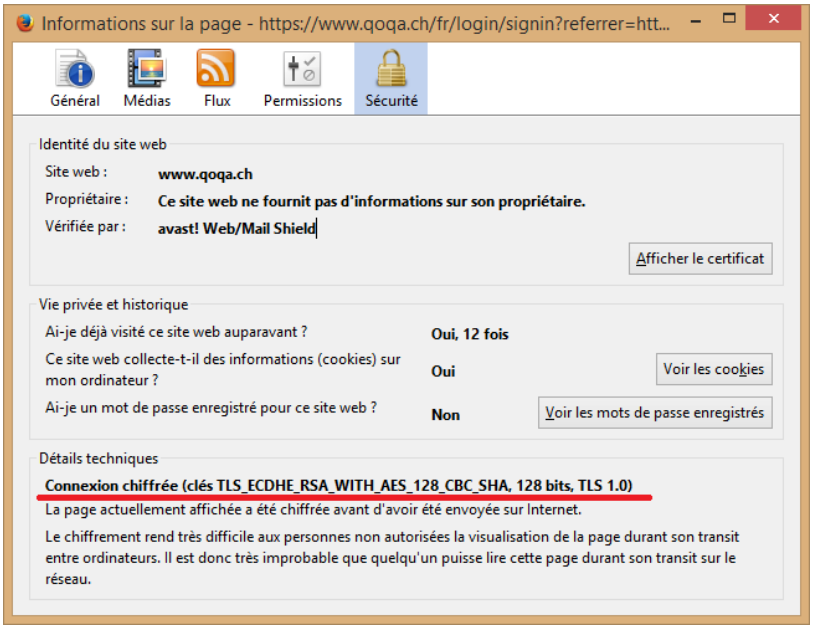
№ 00008

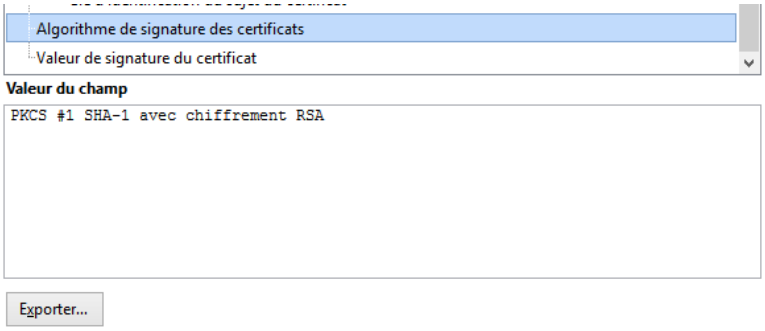
Datum	Walzenlage	Ringstellung	Steckerverbindungen	Kenngruppen
St 31.	IV V I	21 15 16	KL IT FQ HY XC NP VZ JB SB OG	jkm ogi ncj glp
St 30.	IV II III	26 14 11	ZN VO QB ER DK XU GP TV SJ LM	ino udl nam lax
St 29.	II V IV	19 09 24	ZU HL CQ WM OA PY EB TR DN YI	nci oid yhp nip
St 28.	IV III I	03 04 22	YT BX CV ZN UD IR SJ HW GA KQ	zqj hlg xky ebt
St 27.	V I IV	20 06 18	KX GJ EP AC TB HL MW QS DV OZ	bvo sur ccc lqe
St 26.	IV I V	10 17 01	YV GT OQ WN FI SK LD RP MZ BU	jhx uuh giw ugw
St 25.	V IV III	13 04 17	QR GB HA NM VS WD YZ OF XK PE	tba pnc ukd nld
St 24.	III II IV	09 20 18	RS NC WK GO YQ AX EH VJ ZL PF	nfi mew xbk yes
St 23.	V II III	11 21 08	EY DT KF MO XP HN WJ ZL IV JA	lsd nuo vor vcx
St 22.	I II IV	01 25 02	PZ SB OJ XP HA GB VQ UY KW LR	yji rwy rdk nso
St 21.	IV I V	10 17 01	YV GT OQ WN FI SK LD WM BD UQ EC	ema mlv jjy iqh
St 20.	V I III	06 22 03	GH JR TQ KF NZ IL WM BD UQ EC	xjl pgs ggh znd
St 19.	IV II I	12 25 08	TF RQ XV PZ PY NL WI SJ ME GB	vpj zqe jrs cgm
St 18.	II III V	19 14 22	WG QM RL DB ST AQ PZ XH YN IJ	oxd ind ieu ytt
St 17.	IV I II	12 08 21	ME RX BF WY ZD TR FJ AG IL KQ	tak pjs kdh jvh
St 16.	I II III	07 11 15	WZ AB MO TF RX SG QU VJ YN EL	pzg evw wyt iye
St 15.	III II V	06 16 02	GT YC EJ LA RX PN IS WB MH ZV	bhe xzm yzk evp
St 14.	II I V	23 05 24	AZ CJ WF UY SO QV MI NH DP GX	fdx tyj bmq typ
St 13.	IV II V	03 25 10	CX KN JR DQ IU TL HZ MP EP WB	zfo bjr zwx gvn
St 12.	I III II	26 01 18	QB YE WN AI GJ TO HR PK PS CM	upo anf tkr pwz
St 11.	V I III	17 13 04	SV GO PA ZR FN HI YM WT DE BJ	vdh ego wmy uti
St 10.	I V IV	26 07 16	SW AQ NF FO VY UX MK CL HT ZJ	rpl anw vpr mhn
St 9.	I III IV	17 10 18	EH IK GK NZ SP UA LD CQ JM YV	knq ysq rhj tlj
St 8.	V II I	23 11 25	QY OG ST HA CB WD KL JN VX IU	lro avw axh gws
St 7.	II III I	06 12 03	BG FS TH JE VK PI CU QA OD NM	aty mbb mvo jmz
St 6.	I IV V	24 19 01	IR HQ NT WZ VC OY GP LF BX AK	bhc iwo zgz rnr
St 5.	II V III	05 22 14	MK GO RQ XT DW IA ZL SY PJ EN	bok rzw kzo ryl
St 4.	IV II I	15 02 21	KD PG CO FW HJ RY MT QL VB UZ	kpk php xmo pfw
St 3.	III V IV	03 23 04	DY CP WN OV QH UZ RA TI GL SM	hjy nkt ytn pvo
St 2.	I III V	13 18 01	DR VJ FS ZK IU HX AQ GT YO FC	gpq fqw oiy ruj
St 1.	II IV I	06 17 26	AC LS BQ WN MY UV FJ PZ TR OK	ool ooi ywv sfb

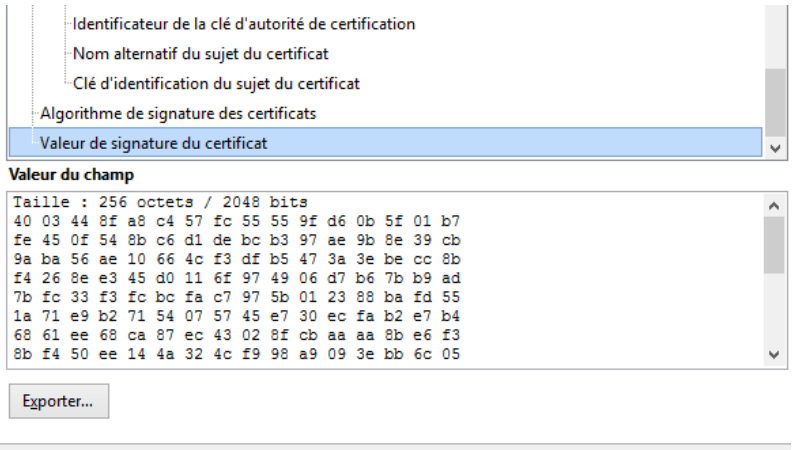
Annexe 3 : Schéma de fonctionnement d'AES



Annexe 4 : Clé de chiffrement RSA dans un navigateur Web







Annexe 5 : Etapes de calcul de modulo inverse en utilisant l'algorithme d'Euclide étendu

Modular inversion

Use the extended Euclidean algorithm to compute a modular multiplicative inverse

---

Computes m for $n^{-1} = m \pmod{p}$, where n and p are coprime.

Displays the steps of the extended Euclidean algorithm.

Set **n** = 13

Set **p** = 60

Submit

---

step	quotient	t_1	t_2	t_3	u_1	u_2	u_3	v_1	v_2	v_3
0					1	0	60	0	1	13
1	4	1	-4	8	0	1	13	1	-4	8
2	1	-1	5	5	1	-4	8	-1	5	5
3	1	2	-9	3	-1	5	5	2	-9	3
4	1	-3	14	2	2	-9	3	-3	14	2
5	1	5	-23	1	-3	14	2	5	-23	1
6	2	-13	60	0	5	-23	1	-13	60	0

The modular inverse is:

37