L’Université Galatasaray

Faculté d’Ingénierie et de Technologie

Génie Informatique

INF315 - Mathématiques Discrètes

Projet de Session

Doğa Yağmur Yılmaz

19401852

Contenu

3)

Chiffrement de Vernam…….

Principe…….

Exemple Théorique……..

Défauts…….

Ressources…….

Attachements :

RSA.ipynb …….(Code en Python)

Diffie-Hellman-Merkle.ipynb ……(Code en Python)

Vernam.ipynb ……..(Code en Python)

Chiffrement RSA

Le RSA est un système cryptographique très utilisé dans le commerce électronique. Il est souvent utilisé pour la sécurisation de données confidentielles comme l'authentification et l'échange de clés symétriques ainsi que dans les cartes bancaires, en particulier lorsqu'elles sont transmises sur un réseau peu sûr comme Internet.

L'algorithme, basé sur les nombres premiers et l'arithmétique modulaire, a été décrit en 1977 par Ron Rivest, Adi Shamir, Leonard Adleman et a été breveté par MIT (Massachusetts Institute of Technology) en 1983.

Cette algorithme RSA (du nom de ses inventeurs) est utilisé pour la cryptographie à clé publique et est basé plus clairement sur le fait qu'il est facile de multiplier deux grands nombres premiers mais difficile de factoriser le produit. C'est l'exemple le plus courant de cryptographie asymétrique, toujours considéré comme sûr, avec la technologie actuelle, pour des clés suffisamment grosses (1024, 2048 bits).

Fonctionnement Général et Descriptif

Le chiffrement RSA est également appelé comme un chiffrement asymétrique étant donné qu'il utilise deux clés différentes mais mathématiquement liées : clé publique et clé privée. L'un de ces deux clés est utilisé pour chiffrer et l'autre est utilisé pour déchiffrer le message. La clé publique peut être partagée avec quiconque tandis que la clé privée doit rester secrète.

De nos jours, RSA est devenu l'algorithme asymétrique le plus répandu du fait qu'il offre une méthode permettant d'assurer la confidentialité, l'intégrité, l'authenticité des communications électroniques et du stockage de données.

« De nombreux protocoles, tels que SSH, OpenPGP, S/MIME et SSL/TLS reposent sur RSA pour leurs fonctions de chiffrement et de signature numérique. Cet algorithme est également utilisé dans des logiciels : les navigateurs en sont un exemple flagrant, car établir une connexion sécurisée sur un réseau peu sûr comme Internet ou valider une signature numérique font partie de leurs attributions. La vérification de signature RSA constitue l'une des opérations les plus couramment réalisées en informatique. » (Citation de la site :

« https://whatis.techtarget.com/fr/definition/cryptographie-asymetrique-cryptographie-a-cle-publique »)

RSA est basé sur la factorisation d'entiers en termes de difficulté de résolution. Il utilise le processus de factorisation tout en chiffrant le texte. Si une personne extérieure essaie de factoriser la valeur donnée sans la clé privée -pour le déchiffrer-, c'est un processus qui peut prendre des années pour déchiffrer même un texte court.

Il existe 3 manières de communiquer avec RSA :

1)

Les deux clés sont créées par une personne (A) qui souhaite de recevoir des données confidentielles par la personne (B). Puis, la personne (A) rend la clé publique accessible et cette clé est utilisée par la personne (B) pour chiffrer son message. Par contre la clé privée est réservée à la personne (A), et lui permet de déchiffrer le message de la personne (B).

2)

Les deux clés sont créées par une personne (A) qui souhaite d'envoyer son message à la personne (B). Après que la personne (A) a chiffré son message avec la clé privée, elle rend la clé publique accessible et cette clé est utilisée par la personne (B) pour déchiffrer le message de la personne (A).

Cette manière peut aussi être expliqué par la e-signature. La clé privée est utilisée par la personne (A) pour signer une donnée qu'elle envoie et la clé publique permet à la personne (B) de vérifier la signature.

3)

Cette manière peut être décrire comme l'union de ces 2 premières manières. Premièrement les deux personnes (A et B) créent leur clés privées et publiques. Puis elles rendent leurs clés publiques accessible. La personne (B) qui souhaite d'envoyer son message, chiffre ses données avec sa clé privée et aussi avec la clé publique de la personne (A). Ainsi la personne (A) utilise sa clé privée et la clé publique de la personne (B) pour déchiffrer le message.

Fonctionnement Détaillé (et en Mathématiques)

"Ronald Rivest, Adi Shamir et Leonard Adleman ont publié leur chiffrement en 1978 dans *A Method for Obtaining Digital Signatures and Public-key Cryptosystems*. Ils utilisent

les congruences sur les entiers et le petit théorème de Fermat, pour obtenir des fonctions à sens unique, avec brèche secrète (ou porte dérobée).” (Citation de la site: https://fr.wikipedia.org/wiki/Chiffrement_RSA#:~:text=Le%20chiffrement%20RSA%20(nom%C3%A9%20par,des%20donn%C3%A9es%20confidentielles%20sur%20Internet.)

L'algorithme RSA comporte trois étapes : la création et la distribution de la clé, le chiffrement (cryptage) et le déchiffrement (décryptage).

Le premier principe de RSA est d’observer les 3 grands entiers positifs « e, d et n » tels qu’avec une exponentiation modulaire pour tous les entiers m (le message transmis)

(avec $0 \leq m < n$) :

$$(m^e)^d \equiv m \pmod{n}$$

En outre, pour certaines opérations, il est pratique que l'ordre des deux exponentielles puisse être modifié et que cette relation implique également :

$$(m^d)^e \equiv m \pmod{n}$$

Comme on a parlé, le chiffrement RSA implique une clé publique et une clé privée. La clé publique peut être partagée avec quiconque et elle est utilisée pour le cryptage des messages. L'objectif est que les messages cryptés avec la clé publique ne puissent être décryptés dans un délai raisonnable qu'en utilisant la clé privée. La clé publique est représentée par les nombres entiers ‘n’et ‘e’ ; la clé privée par le nombre entier ‘d ’ (bien que n soit également utilisé au cours du processus de décryptage, de sorte qu'il pourrait être considéré comme faisant partie de la clé privée).

La Création de la Clé

1)On choisit deux nombres premiers distincts « p » et « q ».

Pour des raisons de sécurité, les nombres entiers p et q doivent être choisis par hasard, et doivent être similaires en magnitude mais différer en longueur de quelques chiffres pour rendre la factorisation plus difficile.
« p » et « q » sont gardés secrets.

2) Calculons « n= p.q »

« n » est utilisé comme modulo pour les clés publiques et privées. Sa longueur, généralement exprimée en bits, est la longueur de la clé. « n » est publié comme faisant partie de la clé publique.

3)Calculons la fonction de Totient « $\phi(n) = (p-1).(q-1)$ »

C'est la valeur de l'indicatrice (ou l'indicateur d'Euler ou fonction phi) en n.

$\phi(n)$ est gardé secret.

4) Choisissons un entier naturel « e » premier avec $\phi(n)$ et strictement inférieur à $\phi(n)$, appelé ***exposant de chiffrement***.

$1< e < \phi(n)$

« e » est publié dans le cadre de la clé publique.

5) Calculons l'entier naturel « d » grâce à « $d.e \equiv 1 \mod (\phi(n))$ »

C'est l'inverse modulaire*(2) de « e » modulo $\phi(n)$ appelé ***exposant de déchiffrement***

« d » peut se calculer efficacement par l'algorithme d'Euclide étendu*(1).

La valeur de « d » va être la clé privée.

En conséquence, notre clé publique se compose de (n, e) tandis que notre clé privée se compose de (n, d).

*(1) l'Algorithme d'Euclide Etendu

L'algorithme d'Euclide étendu est une variante de l'algorithme d'Euclide qui permet, à partir de deux entiers a et b, de calculer non seulement leur plus grand commun diviseur (pgcd), mais aussi un couple de coefficients de Bézout, c'est-à-dire deux entiers u et v tels que

- $a.u + b.v = pgcd(a,b)$

-(1.a) l'Algorithme d'Euclide

La notion principale de cette partie et celle du plus grand commun diviseur (pgcd) de deux entiers.
Le pgcd de deux entiers a et b non nuls est le plus grand entier qui divise à la fois a et b.

Puisqu'on choisit de grands nombres dans l'algorithme RSA, la factorisation ne soit pas facile par le calcul. Alors on utilise l'algorithme d'Euclide par convention. Cet algorithme est basé sur une observation simple.

Si a et b sont deux entiers tels que a>b, alors

- pgcd(a,b) = pgcd(b, a mod b)

Exemple:
Calculer le pgcd(243,198).
>243=1.198+45
>198=4.45+18
>45=2.18+9
>18=2.9+0
Alors pgcd(243,198)= 9.

Donc,

>pgcd(243, 198) = pgcd(198, 243 mod 198)
>pgcd(198, 45) = pgcd(45, 198 mod 45)
>pgcd(45, 18) = pgcd(18, 45 mod 18)
>pgcd(18, 9) = pgcd(9, 18 mod 9)
>pgcd(9,0)= 9.

-(1.b) La Théorème de Bézout

Idéntité: Soient a,b∈Z , il existe deux entiers u,v ∈ Z tels que a.u+b.v = pgcd(a,b).

Théorème : Soient a,b∈Z , les entiers a,b sont premiers entre eux si et seulement s'il existe deux entiers u et v tels que a.u+b.v = 1.

Exemple: On appliquera l'algorithme d'Euclide étendu pour a=r0=243 et b=r1=198. Pour cela, on effectuera à gauche les étapes de l'algorithme d'Euclide (calcul des restes ri et des quotients qi.) En même temps on écrira à droite les calculs pour ui et vi, tels que
ri= ui.a + vi.b

		ri = [ui].a + [vi].b
243=1.198+45	>	45= [1].243+ [−1].198
198=4.45+18	>	18 = 198 − 4.45 =198+(−4).(243−1.198) = [−4].243+ [5].198
45=2.18+9	>	9 =45−2.18

=45−2.(198−4.45)
=9.45−2.198
=9.(243−198)−2·198
= [9].243+ [−11].198

18=2·9+0

Nous avons alors pgcd(243,198)= 9 = [9].243+ [−11]. 198

*(2) L'Inverse Modulaire

On peut utiliser l'algorithme d'Euclide étendu afin de calculer l'inverse modulaire d'un entier. Avant de continuer, on va démontrer un résultat crucial pour la démarche.
Un entier a est inversible modulo n si et seulement si pgcd(a,n)=1.

Supposons maintenant qu'on veut calculer l'inverse de
a mod n, avec n<a.
On vient de montrer que si cet inverse existe, alors forcément pgcd(a,n)=1. En appliquant l'algorithme d'Euclide étendu on obtient un couple (u,v) tels que a·u+n·v=1. On a alors
a·u+n·v = 1
a·u+0 ≡ 1 mod n
a.u ≡ 1 mod n

La dernière équation est la définition de l'inverse. Ceci vaut dire que u est l'inverse de a:

- u ≡ a^(−1)mod n.

Exemple,

On cherche les coefficients de Bézout.
1=5−2.2
=5−2(7−5)
=−2.7+3.5
=−2.7+3.(12−7)
=3.12−5.7
=3.12−5.(67−5.12)
=−5.67+28.12

Nous avons alors

−5.67+28.12=1
28.12 ≡ 1 (mod 67)
28 ≡ 12^(−1) (mod 67)

Chiffrement du Message

En premier temps, la personne (A) rend sa clé publique (n,e) accessible. Puis la personne (B) chiffrer son message comme la suivante :

$$m^e \equiv c \pmod{n}$$

« m » étant un entier naturel strictement inférieur à n représente le message. Alors que « c » étant aussi choisi strictement inférieur à n, représente le message chiffré.

Déchiffrement du Message

Pour déchiffrer le message chiffré « c », on utilise « d » qui est l'inverse modulaire* de « e » modulo « n » :

$$m \equiv c^d \pmod{n}$$

Exemple théorique :

On choisit petits nombres premiers pour capter l'idée facilement mais en pratiques il faut utiliser de très grands nombres premiers.

-Création de la clé :

1) p=11 et q=17
2) n= p.q (qui est le modulo du chiffrement)
   =187
3) $\phi(n) = (p-1).(q-1)$
   =160 (on le verra comme phi dans l'exemple du code)
4) On choisit e=3 étant premier avec 160 comme exposant de chiffrement
5) On trouve d =107 par l'inverse modulaire de 3 modulo 160 ($e.d = 3.107 \equiv 1 \pmod{160}$)

La clé publique de la personne(A) est (n,e) = (187, 3) et sa clé privée est (n,d)=(187, 107).

-Chiffrement du Message :

La personne(B) chiffre son message m=2 avec la clé publique de la personne(A) :

- $2^3 \equiv 8 \pmod{187}$, donc le message chiffré est c=8 que (B) transmet à (A).

-Déchiffrement du Message :

La personne (A) déchiffre le message avec sa clé privée :

- 8^107 ≡ 2 (pour le calcul : « https://www.mtholyoke.edu/courses/quenell/s2003/ma139/js/powermod.html »)

Donc la personne (B) retrouve le message initial m=2.

Le mécanisme de signature, la deuxième manière, en échangeant les clés est analogue.

Désavantages du Chiffrement RSA

L'un des inconvénients les plus importants de la méthode de chiffrer se produit au stade de la recherche de grands nombres premiers. Comme on le sait, il n'est pas facile de savoir si un nombre considéré est premier ou non. Le théorème de Fermat peut être utilisé pour cela.

Chaque utilisateur a la possibilité de décrypter (déchiffrer) tous les textes cachés avec une seule clé en conservant son propre mot de passe. Cela pose de grandes difficultés si la clé est perdue ou si quelqu'un d'autre la récupère.

Comme les clés doivent être échangées, ils doivent utiliser un réseau pour transmettre la clé à l'autre partie. Cela nécessite de prendre des mesures de sécurité supplémentaires sur le réseau.

Par exemple, s'il y a n utilisateurs, il est nécessaire de créer n - 1 mot de passe et il doit être conservé dans ce système. Cela peut également être considéré comme un inconvénient car cela prendra de l'espace mémoire supplémentaire.

Sécurité de RSA

Comme on a indiqué en haut, la sécurité de RSA repose sur la difficulté de la factorisation du produit de deux grands entiers. Cependant, à mesure que la puissance de traitement augmente et que des algorithmes de factorisation plus efficaces sont découverts, il devient possible de factoriser des nombres de plus en plus élevés. La puissance du chiffrement est directement liée à la taille de la clé. De ce fait, un doublement de la longueur de la clé renforce le chiffrement de façon exponentielle au détriment des performances.

Les clés de RSA sont généralement 1024 ou 2048 bits, mais les experts pensent que les clés de 1024 bits pourraient être déchiffrer dans le futur proche. C'est la raison pour laquelle le secteur privé et l'administration commencent à adopter des clés d'une longueur minimale de 2048 bits.

2)

Chiffrement de Diffie-Hellman-Merkle

Chiffrement de Diffie-Hellman-Markle est une méthode d'échange sécurisé de clés cryptographiques sur un canal public et a été l'un des premiers protocoles à clé publique conçu par Ralph Merkle et nommé d'après Whitfield Diffie et Martin Hellman. DH est l'un des premiers exemples pratiques d'échange de clés publiques mis en oeuvre dans le domaine de la cryptographie. Publié en 1976 par Diffie et Hellman, il s'agit du premier ouvrage connu du public qui proposait l'idée d'une clé privée et d'une clé publique correspondante. Traditionnellement, les communications cryptées sécurisées entre deux parties nécessitaient d'abord qu'elles échangent des clés par des moyens physiques sécurisés, tels que des listes de clés papier transportées par un service de messagerie de confiance.

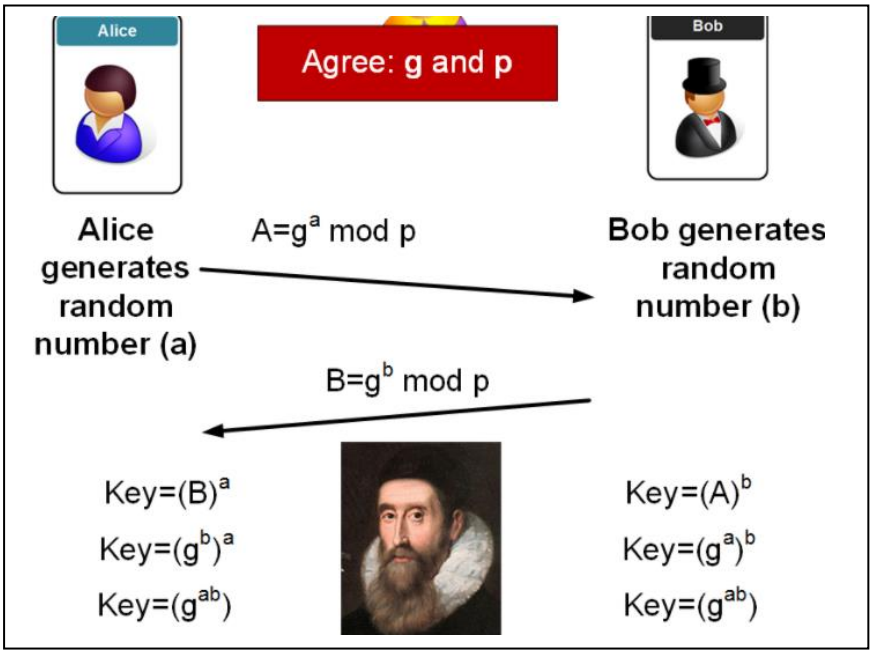
Fonctionnement Générale

La méthode d'échange de clés Diffie-Hellman permet à deux parties qui ne se connaissent pas préalablement d'établir conjointement une clé secrète partagée sur un canal non sécurisé. Cette clé peut ensuite être utilisée pour chiffrer les communications suivantes à l'aide d'un chiffrement à clé symétrique. Diffie–Hellman est utilisé pour sécuriser une variété de services Internet. Cependant, des recherches publiées en octobre 2015 suggèrent que les paramètres utilisés pour de nombreuses applications Internet DH à cette époque ne sont pas assez solides pour empêcher la compromission par des attaquants très bien financés, tels que les services de sécurité de certains pays.

Fonctionnement Détaillée

L'échange de clés Diffie-Hellman, également connu sous le nom d'échange de clés exponentiel, est une méthode de chiffrement numérique s'appuyant sur des nombres élevés à des puissances données de manière à obtenir des clés de déchiffrement sur la base de composantes jamais transmises directement afin de rendre mathématiquement insurmontable toute tentative de décryptage du code. L'échange de clés Diffie-Hellman définit une clé secrète partagée entre deux parties, qui peut être utilisée pour communiquer secrètement et échanger des données via un réseau public. Cette méthode fait appel à des techniques de clé publique pour échanger une clé de chiffrement privée.

L’Algorithme de Diffie-Hellman-Merkle



Alice et Bob choisissent une base g et un nombre premier p. Alice choisit un nombre secret a. Alice calcule la valeur publique (A) en prenant la base g et l’augmente à la puissance a puis fait un modulo avec le nombre premier p. Elle envoie sa clé publique (A) à Bob. Bob choisit un nombre secret b et calcule la valeur publique (B) en prenant la base g et l’augmente à la puissance b puis fait un modulo avec le nombre premier p. Il envoie sa clé publique (B) à Alice. Alice réalise l’opération suivante pour obtenir sa clé secrète [SA= ((g^b mod p)^a mod p)]. Bob réalise la même opération qu’Alice, avec les valeurs qu’il a reçu, pour obtenir sa clé secrète [SB= ((g^a mod p)^b mod p)] qui est égale SA.

Exemple Théorique

Imaginons qu'Alice utilise le nombre premier « 3 »et Bob le base « 5 ». Alice sélectionne un nombre aléatoire privé(a), par exemple « 2 », et calcule « 5^2 mod 3 », qui est égal à « 1 », et envoie le résultat(A) publiquement à Bob. Bob sélectionne alors son nombre aléatoire privé(b), par exemple « 3 », calcule « 5^3 mod 3 », qui est égal à « 2 », puis envoie le résultat(B) publiquement à Alice. Le cœur de la stratégie est le calcul suivant. Alice utilise le

résultat public de Bob (B) « 2 » et calcule « 2^2 mod 3 ». Le résultat obtenu « 1 » est leur clé secrète partagée. Bob utilise alors le résultat public d'Alice (A) « 1 » et calcule « 1^3 mod 3 », qui correspond en effet à la même clé secrète « 1 ». Désormais, Alice et Bob peuvent communiquer à l'aide de l'algorithme symétrique de leur choix et de la clé secrète partagée, qui n'a jamais été transmise via un canal non sécurisé.

Si une tierce partie interceptait l'échange, il lui serait difficile en termes de puissance de calcul de déterminer la clé secrète. En pratique, si de grands nombres sont utilisés, cette action demanderait d'énormes ressources aux supercalculateurs modernes pour obtenir le résultat dans un délai raisonnable.

Code en Python :

```python
from random import randint

if __name__ == '__main__':

#choisissons une base g et un nombre premier p
    g = 5
    p = 3


    print('le base g = %d'%(g))
    print('le nombre premier p = %d'%(p))


    #Alice choisit un nombre secret a
    a = 2
    print('la clé privée de Alice :%d'%(a))

    # Bob choisit un nombre secret b
    b = 3
    print('la clé privée de Bob :%d'%(b))


    # Alice et Bob calculent les valeurs publiques
    A= int(pow(g,a,p))
    B = int(pow(g,b,p))


    # la clé secrète pour Alice : ((g^b mod p)^a mod p)
    SA = int(pow(B,a,p))

    # la clé secrète pour Bob : ((g^a mod p)^b mod p)
    SB = int(pow(A,b,p))

    print('la clé secrète pour Alice : %d'%(SA))
    print('la clé secrète pour Bob : %d'%(SB))
```

```
le base g = 5
le nombre premier p = 3
la clé privée de Alice :2
la clé privée de Bob :3
la clé secrète pour Alice : 1
la clé secrète pour Bob : 1
```

Comparaison de RSA et DH

Les algorithmes de chiffrement à clé publique RSA et Diffie-Hellman sont tous deux suffisamment robustes pour un usage commercial, car ils se basent sur des problèmes supposés insurmontables, respectivement la difficulté de factorisation des grands nombres et l'élévation à la puissance et l'arithmétique modulaire. Avec leurs clés de 1 024 bits, les deux algorithmes dépassent la longueur de clé minimale recommandée pour les systèmes de chiffrement, à savoir 128 bits. Tous deux ont été étudiés en détail par des mathématiciens et des cryptographes, mais assurent une sécurité équivalente, pourvu qu'ils soient mis en œuvre correctement.

Toutefois, la nature de l'échange de clés Diffie-Hellman le rend plus susceptible aux attaques de l'homme du milieu (HDM), car il n'authentifie aucune des parties prenantes lors de l'échange. La manœuvre HDM peut également créer une paire de clés et usurper les messages entre les deux parties, qui pensent à tort communiquer entre elles. C'est pourquoi l'algorithme Diffie-Hellman est employé conjointement à une autre méthode d'authentification, en règle générale des signatures numériques.

Contrairement à l'algorithme Diffie-Hellman, l'algorithme RSA peut servir à signer des signatures numériques et assurer l'échange de clés symétriques, mais exige d'abord un échange de clés publiques. Cependant, une étude récente a démontré que même les clés RSA de 2 048 bits peuvent être effectivement rétrogradées via une attaque de l'homme du milieu ou de type « padding oracle ». Le rapport suggère que la contre-mesure la plus sûre consiste à abandonner l'échange de clés RSA et migrer vers des échanges de clés Diffie-Hellman (à courbe elliptique).

3)

Chiffrement de Vernam

Le chiffre de Vernam (ou masque jetable) est considéré comme une amélioration significative du chiffre de Vigenère car cet algorithme de chiffrement est théoriquement impossible à casser. Cependant, aucuns systèmes de chiffrement n'est parfait, et même si ce dernier semble sécurisé, il n'est pas toujours facile de remplir les conditions pour avoir un message chiffré totalement sûr.

Principe

Le chiffre de Vernam est un chiffrement symétrique utilisant, comme le chiffre de Vigenère, une substitution poly-alphabétique. Plutôt que d'avoir un seul décalage fixe, chaque caractère de la clé définit le décalage du caractère à chiffrer. Son efficacité réside dans le choix de la clé de chiffrement, qui doit respecter plusieurs règles fondamentales :

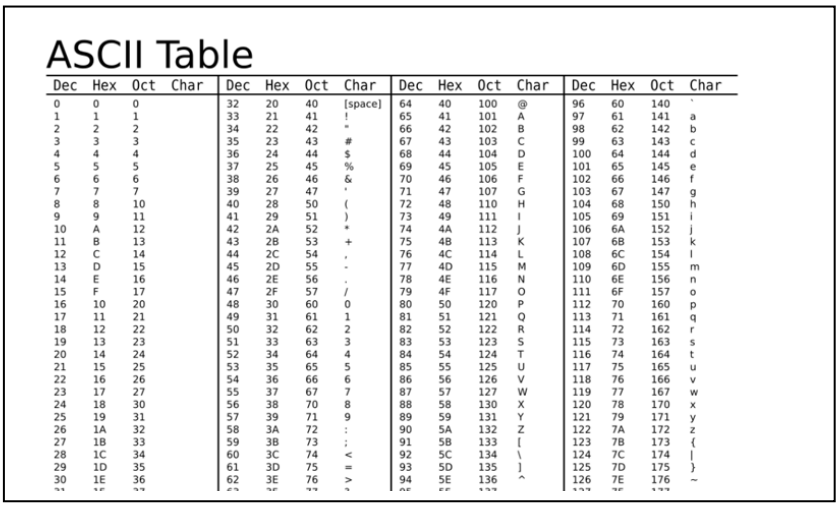
Chaque clé est unique, et cette dernière ne doit pas être réutilisée (d'où le nom de masque jetable).

-La clé doit être de la même longueur que l'information à chiffrer.

-Chacun des caractères de la clé doit être choisi totalement aléatoirement.

-Afin de garantir une sécurité maximale, la clé ne doit jamais être réutilisée (masque jetable).

Chiffrement



ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~

Pour crypter le message, chaque caractère du texte en clair et la clé devront être convertis en un code numérique. Heureusement, il existe déjà des schémas de codage pour ce faire, et nous pouvons utiliser des codes ASCII standard. Comme vous le savez peut-être déjà, dans le système de codage ASCII, chaque caractère reçoit un code numérique.

	_	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

Exemple Théorique

Prenons l'exemple du message « hell » chiffré avec la clé « abcd ». La première lettre du message à chiffrer est « h » (huitième lettre de l'alphabet) et la première lettre de la clé est « a » (premier lettre de l'alphabet). Cela signifie que la nouvelle valeur de h dans ce cas-là sera ı (h (8) + a(1) = ı(9)).

Défauts

Notre algorithme de chiffrement est donc techniquement impossible à casser si les règles permettant de choisir une clé de chiffrement sont respectées. Cependant, suivre les contraintes peut parfois être difficile :

La création de clés uniques demande une grande organisation et beaucoup de coordination pour mettre à jour une base de données (secrète de préférence) qui contient les clés déjà utilisées, et qui est accessible et commune aux personnes autorisées.

Créer une clé parfaitement aléatoire est difficile, en effet nos ordinateurs ne font que simuler l'aléatoire grâce à des calculs mathématiques et génèrent donc des clés pseudo-aléatoires. Il est possible de savoir à l'avance la clé générée par l'ordinateur si l'on connait suffisamment d'informations dessus (algorithme utilisé, graine d'initialisation, etc.). Par exemple en C, la fonction rand utilise un algorithme de génération de nombre pseudo-aléatoire basé sur des congruences et une fonction linéaire, et il est possible de deviner le résultat si l'on connait la graine initialisée avec srand. Une solution permettant de générer une clé totalement aléatoire serait d'utiliser des phénomènes physiques dont le résultat ne peut être déterminé à l'avance comme des bruits thermiques, des bruits atmosphériques ou encore une réaction radioactive

RESSOURCES

-“Elementary Number Theory
A revision by Jim Hefferon, St Michael’s College, 2003-Dec
of notes by W. Edwin Clark, University of South Florida, 2002-Dec”

-RSA – Bilgisayar Kavramları (bilgisayarkavramlari.com)

-Chiffrement RSA — Wikipédia (wikipedia.org)

-Que signifie cryptographie asymétrique (cryptographie à clé publique)? - Definition IT de Whatis.fr (techtarget.com)

-RSA Şifreleme Algoritması ve Örnek Çözümü - Berk Arat

-https://www.dcode.fr/chiffre-rsa

-https://www.youtube.com/watch?v=J6pgSaFsMAI

https://tr.wikipedia.org/wiki/RSA_(%C5%9Fifreleme_y%C3%B6netimi)#:~:text=RSA%2C%20g%C3%BCvenli%C4%9Fi%20tam%20say%C4%B1lar%C4%B1%20%C3%A7arpanlar%C4%B1na,t%C3%BCr%20A%C3%A7%C4%B1k%20anahtarl%C4%B1%20%C5%9Fifreleme%20y%C3%B6ntemidir.&text=Bir%20RSA%20kullan%C4%B1c%C4%B1s%C4%B1%20iki%20b%C3%BCy%C3%BCk,Se%C3%A7ilen%20asal%20%C3%A7arpanlar%C4%B1%20ise%20saklar.

-https://en.wikipedia.org/wiki/RSA_(cryptosystem)

https://fr.wikipedia.org/wiki/%C3%89change_de_cl%C3%A9s_Diffie-Hellman