Arnaud Guilhamat
Guia2612 10/10/2022

Rapport Laboratoire : Attaques cryptographiques

Question 1)

La représentation numérique d'un caractère permet d'implémenter une translation de caractère facilement car on peut faire les opérations d'addition et de soustraction de la clé k. On peut également utiliser le modulo sur des valeur numériques. Cette représentation numérique des caractères nous permet de faire des opérations numériques qu'on ne peut pas faire sur des lettres.

L'idée va donc être de convertir les caractères en valeur numérique, de faire les opérations nécessaires pour chiffrer ou déchiffrer puis de reconvertir en charactères pour obtenir le message chiffré/déchiffré.

Un algorithme qui fait une translation d'un caractère pourrait être le suivant :

Entrées : caractère char compris entre 'a', …, 'z'

Clé k ∈ ℤ

Sorties : caractère char translaté

Fonction translation(char, k) :

i = ord(char) – ord('a') // on fait une translation de la valeur numérique du caractère pour
// avoir une valeur comprise entre 0 et 26 pour pouvoir faire le modulo

i = (i + k) % 26 // On fait le modulo de 26 pour toujours avoir une lettre, même quand on
// sort de la range [0, 26], ce qui permet que la lettre après le z soit le a et la lettre d'avant le a soit le z

return chr(i + ord('a')) // On refait la translation pour retourner dans le référentiel numérique
// de la table ASCII puis on retourne le caractère correspondant

Question 4)

Pour une clé k = 3, et un message m = "HeLlo wOrld99*&*&*@", l'algorithme retourne le même résultat que pour la clé k' = 29, (3+ 26), soit « khoor zruog99*&*&*@ ».

Le nombre de clés différentes qui fournissent un résultat différent est de 25.

Il est donc très facile d'attaquer le code de Caesar avec une approche de force brute si on sait que l'algo utilisé est le code de Caesar, qu'on a intercepté le chiffré, et même si on ne connait pas la clé k car il n'y a que très peu de possibilités possible (25) !

Il suffirait donc de tester toutes ces possibilités et de vérifier quel message décodé semble être cohérent.

Arnaud Guilhamat
Guia2612
10/10/2022

Question 5)

Voici le dictionnaire que nous retourne l'algorithme pour le message : 'fytgpcdtep op dspcmczzvp'

{
"0": "fytgpcdtep op dspcmczzvp",
"1": "exsfobcsdo no croblbyyuo",
"2": "dwrenabrcn mn bqnakaxxtn",
"3": "cvqdmzaqbm lm apmzjzwwsm",
"4": "bupclyzpal kl zolyiyvvrl",
"5": "atobkxyozk jk ynkxhxuuqk",
"6": "zsnajwxnyj ij xmjwgwttpj",
"7": "yrmzivwmxi hi wlivfvssoi",
"8": "xqlyhuvlwh gh vkhueurrnh",
"9": "wpkxgtukvg fg ujgtdtqqmg",
"10": "vojwfstjuf ef tifscspplf",
"11": "universite de sherbrooke",
"12": "tmhudqrhsd cd rgdqaqnnjd",
"13": "slgtcpqgrc bc qfcpzpmmic",
"14": "rkfsbopfqb ab peboyollhb",
"15": "qjeranoepa za odanxnkkga",
"16": "pidqzmndoz yz nczmwmjjfz",
"17": "ohcpylmcny xy mbylvliiey",
"18": "ngboxklbmx wx laxkukhhdx",
"19": "mfanwjkalw vw kzwjtjggcw",
"20": "lezmvijzkv uv jyvisiffbv",
"21": "kdyluhiyju tu ixuhrheeau",
"22": "jcxktghxit st hwtgqgddzt",
"23": "ibwjsfgwhs rs gvsfpfccys",
"24": "havirefvgr qr fureoebbxr",
"25": "gzuhqdeufq pq etqdndaawq"
}

Le seul message cohérent en celui correspondant à la clé 11.
On peut donc en déduire que les valeurs sont très probablement clear_text = « universite de sherbrooke » et k ≡ 11 (mod 26)

Question 6)

UTF-8 est un schéma de codage. Il ne suffit pas d'ajouter simplement les octets bruts ensemble, on doit d'abord supprimer les parties encodées, puis concaténer (pas ajouter) les bits restants ensemble.

En prenant le cas du caractère ‘é’, dont l’Unicode point est U+00E9 (LATIN SMALL LETTER E WITHACUTE), on doit prendre la deuxième ligne de Table 2 pour U+00E9, ce qui veut dire qu’on utilise 2 octets, avec en byte 1 : 110xxxxx et en byte 2 : 10xxxxxx avec les 1 et les 0 qui doivent apparaître tels quels dans les octets encodés et les x qui vont être les bits originels du Unicode point.

On a donc :

11000000 10000000

OR 00000011 00101001

== 11000011 10101001 = 0xC3 0xA9

La valeur décimale du code ASCII étendu associé au caractère ‘é’ est de 233.

Question 7)

À chaque lettre on a n (le nombre de lettres de l’alphabet) permutations possibles mais comme une clé ne peut être reprise deux fois, on a :

Le nombre de combinaisons possibles, m = 1 * 2 * 3 * … * n-1 * n = n!

Pour notre alphabet de 26 lettres, m = 26! ≈ 2^88 < 2^128

La recommandation est d’avoir une sécurité d’un grand minimum de 128 bits soit 2^128 combinaisons possibles pour rendre le brute force plus ou moins inefficace, ou alors trop lent pour être considéré comme efficace.

Or notre algorithme ne respecte pas le nombre minimal de possibilité, il est donc tout à fait attaquable par force brute, en testant toutes les combinaisons possibles jusqu’à trouver un résultat cohérent (comme dans la question 5)

Question 10)

Voici les résultats des dernières questions, déterminant le nombre d'occurrences de chaque charactère pour le texte en clair clear_text_hugo à gauche et le le cipher_text_1 à droite :

```
freq_hugo = [
  (" ", 285),
  ("e", 232),
  ("s", 119),
  ("a", 114),
  ("l", 102),
  ("r", 99),
  ("u", 95),
  ("t", 95),
  ("i", 84),
  ("n", 81),
  ("d", 66),
  ("o", 62),
  ("m", 49),
  ("c", 37),
  ("é", 27),
  ("p", 24),
  ("v", 20),
  ("g", 19),
  ("q", 19),
  ("b", 19),
  ("’", 18),
  (",", 18),
  ("f", 18),
  ("h", 15),
  (".", 11),
  ("y", 9),
  ("x", 6),
  ("-", 5),
  ("è", 5),
  ("j", 5),
  ("\n", 3),
```

```
freq_cipher_text_1 = [
  ("\x96\x97", 285),
  ("\x1e\x0e", 232),
  ("\x06,", 119),
  ("\xf0\x0a", 114),
  (")5", 102),
  ("\x85$", 99),
  ("\x86\xdd", 95),
  ("5\x92", 95),
  ("\x13\x99", 84),
  ("\xad\x0d", 81),
  ("!\xb0", 66),
  ("\xb6\xe6", 62),
  ("\x95\xcc", 49),
  ("\x81\xf5", 37),
  ("@l", 27),
  ("\xab\x9a", 24),
  ("\x9f\xde", 20),
  ("\t\x8e", 19),
  ("\\\x07", 19),
  ("\xaa\xb6", 19),
  ("\x1b\xfd", 18),
  ("\xd6\xc6", 18),
  ("\x8bu", 18),
  ("i\x19", 15),
  ("\xa06", 11),
  ("\xbf\xec", 9),
  ("\x925", 6),
  ("F\xf2", 5),
  ("\x1c\x08", 5),
  ("t\x8c", 5),
  ("z*", 3),
```

On remarque que le nombre d'occurrences de chaque caractère est exactement le même dans les deux textes. La probabilité est quand même extrêmment faible pour que ca ne soit qu'une coincidence ! On peut donc en conjecturer que le clear_text_hugo est le texte en clair qui correspond à cipher_text_1.

Question 12)

La clé qui marchait bien pour la question 11 et qui nous permettait de déchiffrer le texte cipher_text_1 partiellement, le rendant compréhensible, n'est pas la même pour le cipher_text_2. En effet le message déchiffré est incohérent, on ne peut reconnaître aucun mot.

Question 13)

Le début du message en clair retourné est :

Anton Voyl n'arrivait pas à dormir* *l all*ma* *on *a* mar**ait min*it vin*t* *l po*ssa *n pro*ond so*pir* s'assit dans son lit* s'app*yant s*r son polo**on* *l

A noter que j'ai remplacé les caractères manquants dans un premier temps par le caractère « * » pour améliorer la lisibilité.

Si on laisse les octets correspondants, on obtient le message :

Anton Voyl n'arrivait pas à dormirb')\x1d' b'\xf9i'l allb't\xe7'mab')\x1d' b'\x07\xf7'on b'\x93\x16'ab'\xcbV' marb'\xa1\x05'b't\xe7'ait minb't\xe7'it vinb'P\xc5'tb')\x1d' b'\xf9i'l pob't\xe7'ssa b't\xe7'n prob'W\xe5'ond sob't\xe7'pirb'?\xed' s'assit dans son litb'?\xed' s'appb't\xe7'yant sb't\xe7'r son polob'\xab\x92'b'\xdc0'onb')\x1d'

On peut distinguer quelques mots et donc identifier des potentielles correspondances qu'on pourrait ajouter à notre dictionnaire :

Ainsi on remarque les nombreuses occurrences de b')\x1d' qui semble être placé à chaque fois à la fin des phrases (juste après dormir, juste après alluma, juste après vingt, juste après polochon)

On peut donc conjecturer que b')\x1d' correspond au caractère '.'

On remarque également de nombreuses occurrences de b't\xe7' qui par déduction en complétant instinctivement certains mots (alluma, minuit, poussa) nous permet d'assigner b't\xe7' au 'u' et de remplacer toutes ses occurrences dans le text en clair.

Puis après plusieurs étapes de déduction, on peut finalement en déduire le texte en clair (3 premières phrases) :

Anton Voyl n'arrivait pas à dormir. Il alluma. Son *a* marquait minuit vingt.

Cependant, on ne peut pas deviner instinctivement et relativement surement ce que le mot b'\x93\x16'ab'\xcbV' veut dire, et il n'y a pas d'autres occurrences des caractères b'\x93\x16' et b'\xcbV' dans le texte entier…
Une simple recherche sur internet du texte déjà préalablement déchiffré nous permet de trouver un extrait de « la disparition » de Georges PEREC. Le texte déchiffré correspond, et nous permet d'identifier les caractères manquants…

Les trois premières phrases du textes sont donc :

Anton Voyl n'arrivait pas à dormir. Il alluma. Son Jaz marquait minuit vingt.