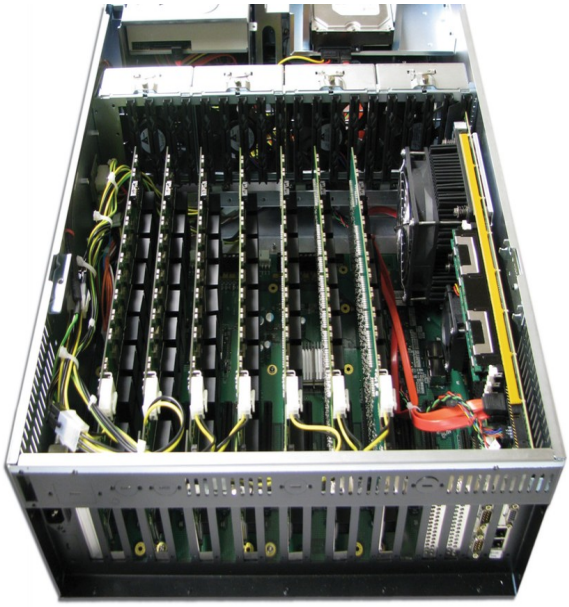
h e p i a
Haute école du paysage, d'ingénierie
et d'architecture de Genève

Hes·SO GENÈVE
Haute Ecole Spécialisée
de Suisse occidentale

Bruteforce password attack on FPGAs



Thèse de Bachelor présentée par

Abivarman KANDIAH

pour l'obtention du titre Bachelor of Science HES-SO en

Informatique et systèmes de communication avec orientation
Systèmes informatiques embarqués

Septembre 2024

Professeur-e HES responsable

Andres UPEGUI POSADA

Mandant

ELCA Security

Légende et source de l’illustration de couverture :

https://militaryembedded.com/cyber/encryption/accelerating-cryptography-fpga-clusters

TABLE DES MATIÈRES

La table des matières doit reprendre tous les niveaux de titre et sous-titre du mémoire, y compris les pages initiales (page des remerciements, énoncé du sujet, résumé, table des annexes et autres tables), ainsi que les références documentaires, etc.

< *Insérez ici votre dédicace* > (facultatif)

REMERCIEMENTS

< Formulez ici vos remerciements aux personnes qui vous ont aidé dans la réalisation de votre travail. >

ÉNONCÉ DU SUJET

< Insérez ici la page d'énoncé complété et signé
par l'enseignant-e responsable (cf. feuille de style fournie) >

(obligatoire)

Attention : Tout l'énoncé doit tenir sur une seule page

RÉSUMÉ

< Insérez ici la page d'énoncé complété et signé
par l'enseignant-e responsable (cf. feuille de style fournie) >

(obligatoire)

Attention : Tout l'énoncé doit tenir sur une seule page

LISTE DE ACRONYMES

FPGA Field-Programmable Gate Array. 2

IC Integrated Circuit. 2

SBOX Substitution boxes. 7

VHDL Very High Speed Integrated Circuit Hardware Description Language. 2

LISTE DES ILLUSTRATIONS

Références des URL

- URL01 ce-site.ch/bla/bli/blo/blou.html
- URL03 ce-site.ch/blou/bli/bla.html
- URL04 https://commons.wikimedia.org/w/index.php?curid=906980
- URL06 ce-site.ch/monrapportdestage.pdf

LISTE DES TABLEAUX

N.B. Si vous avez peu de tableaux, vous pouvez les intégrer à la table des illustrations.

Références des URL

— URL02 ce-site.ch/bli/bla/blo/blou

— URL05 ce-site.ch/publications/documents/rapports/rapportsdestage/monrapportdestage.pdf

LISTE DES ANNEXES

INTRODUCTION

Votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte.

CHAPITRE 0 : BASE TECHNIQUE

Ce chapitre a pour but d'introduire et expliquer les différents aspects techniques clés de ce projet de Bachelor. Je vais notamment expliquer brièvement ce qu'est un Field-Programmable Gate Array (FPGA) et le principe d'une fonction de hachage.

1.1. FPGA

Un FPGA est un Integrated Circuit (IC) dans lequel on peut programmer et interconnecter des circuits logiques. Contrairement à un processeur, qui est limité par un certain nombre d'instructions et exécute les instructions de manière séquentielle, un FPGA permet d'exécuter de nombreux circuits logiques en parallèle.

Pour programmer un FPGA, on utilise généralement des langages de description matériel tels que le Very High Speed Integrated Circuit Hardware Description Language (VHDL) et le Verilog. Dans ce projet, j'ai personnellement travaillé avec le VHDL.

Le processus de programmation d'un FPGA est séparé en deux étapes : la synthèse et l'implémentation. La synthèse consiste à traduire le code VHDL en registres et portes logiques. L'implémentation lui consiste à placer les différents composants logiques sur le FPGA et à les interconnecter. Ces différents processus vont prendre généralement beaucoup de temps.

Lorsque l'on souhaite tester un programme VHDL, il est possible d'utiliser des outils de simulation afin de vérifier le fonctionnement souhaité. Il est aussi possible d'automatiser la phase de simulation à l'aide de fichier que l'on appelle testbench. La simulation va permettre de valider une première fois le programme afin d'éviter de perdre du temps à reprogrammer le FPGA.

Durant ce travail, j'ai utilisé Vivado qui est le logiciel qui m'a permis la simulation et la programmation des FPGA qui ont été utilisés.

1.2. FONCTION DE HACHAGE

Une fonction de hachage est une fonction qui va prendre en entrée une donnée a taille variable et va ressortir une donnée de taille fixe.

Une des propriétés fondamentales d'une fonction de hachage est qu'il n'existe pas de fonction mathématique permettant de retrouver la donnée originale à partir d'un hash généré. Même une petite modification apportée à la donnée en entrée conduira à un hash totalement différent

en sortie. Cette particularité est essentielle pour sécuriser le stockage des mots de passe, car même si des hash venait à être compromises, il est extrêmement difficile de retrouver les mots de passe originaux à partir de leurs hachages.

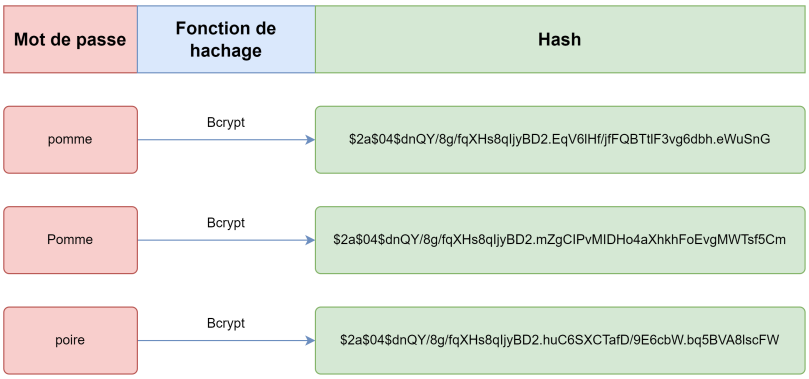


ILLUSTRATION 1.1 – Fonction de hachage. Source : réalisé par Kandiah Abivarman

a. Salt

Certaines fonctions de hachage tel que le bcrypt utilisent ce qu'on appelle un salt (sel en français), qui est une valeur générée aléatoirement qu'on va donner avec notre mot de passe. Le salt va permettre d'avoir un hash différent, même si deux personnes utilisent le même mot de passe, ajoutant ainsi une couche supplémentaire de sécurité.

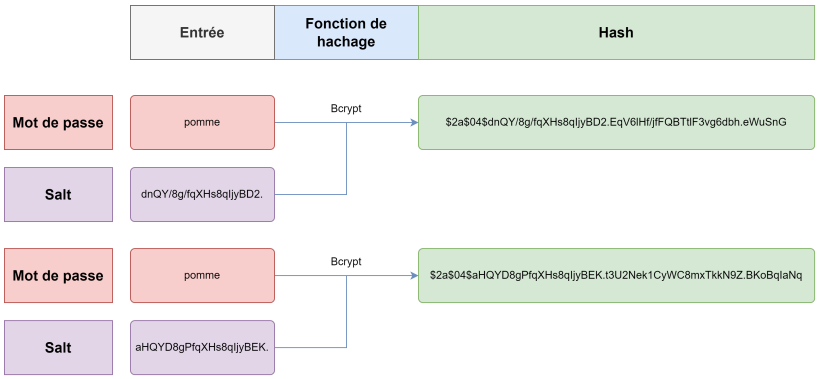


ILLUSTRATION 1.2 – Salt. Source : réalisé par Kandiah Abivarman

b. Attaque par bruteforce

L'attaque par bruteforce consiste à essayer toutes les combinaisons possibles de mots de passe afin de retrouver celui qui correspond au hash compromis. Cette méthode repose sur le fait qu'il est impossible de retrouver directement le mot de passe à partir du hash, obligeant ainsi l'attaquant à tester différentes entrées jusqu'à ce qu'il trouve celle qui génère le hash recherché.

Toutefois, cette méthode peut prendre beaucoup de temps, notamment lorsque les fonctions de hachage utilisées sont conçues pour être lentes à calculer.

CHAPITRE 1 : ANALYSE

2.1. DESCRIPTION DU PROJET

Votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte.

2.2. MÉTHODES DE COMMUNICATION

Votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte.

a. UART

b. PCIe

c. Ethernet

2.3. BCRYPT

Pour ce projet, nous avons décidé de cibler le Bcrypt, car c'est une fonction de hachage qui prend du temps à être calculé.

Le Bcrypt est une fonction de hachage avec comme particularité, un paramètre supplémentaire qui est le cost (coût en français). Ce paramètre va définir le nombre d'itérations que va prendre la fonction de hachage, de ce fait plus le cost est élevé, plus le calcul va prendre du temps.

a. Algorithme

L'algorithme du Bcrypt se base sur l'algorithme de chiffrement Blowfish [1] qui est une fonction de chiffrement à clef symétrique, c'est-à-dire que la même clef est utilisée pour le chiffrement et le déchiffrement. L'algorithme du Bcrypt peut être divisé en deux grandes étapes.

1. *Blowfish Algorithm with Examples*. en-US. Section: Algorithms. Oct. 2019. URL : `https://www.geeksforgeeks.org/blowfish-algorithm-with-examples/` (visité le 21/03/2024).

On a une première étape qui est une phase de mise en place des clés symétriques. Dans cette étape, on va créer les clés de chiffrements à partir des paramètres d'entrée de la fonction de hachage (mot de passe, salt, cost). Cette première étape est la partie la plus coûteuse de la fonction, car la mise en place de la clé va prendre plus ou moins de temps en fonction du cost. Les clés de chiffrement sont composées de Subkeys qui est un tableau de 18 entiers de 32 bits et quatre Substitution boxes (SBOX) qui sont chacun des tableaux de 256 entiers de 32 bits. Avant de calculer ces clés de chiffrements, ils sont tout d'abord initialisés avec les décimales de PI.

Puis il y a la deuxième étape, où l'on va utiliser les clés de chiffrement qui ont été calculées plus tôt afin de chiffrer la phrase magique "OrpheanBeholderScryDoubt", le chiffrement va être fait 64 fois.

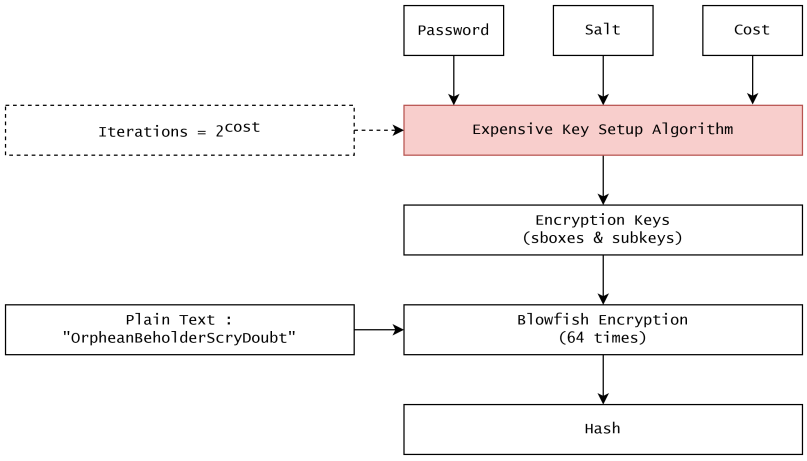


ILLUSTRATION 2.1 – Algorithme Bcrypt. Source : réalisé par Kandiah Abivarman

b. Format du Hash

Le hash généré par la fonction Bcrypt est généralement stocker sous une forme particulière.

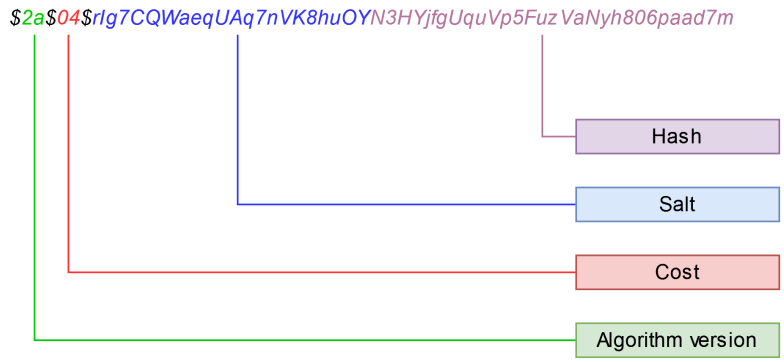


ILLUSTRATION 2.2 – Format du hash Bcrypt. Source : réalisé par Kandiah Abivarman

On va avoir un premier champ qui contient la version de l'algorithme, un deuxième qui contient le cost de la fonction, un troisième avec le salt et le quatrième avec le hash généré. Le salt et le hash sont en base 64, mais il faut faire attention, car c'est une base 64 différente de la norme RFC 4648 [2] qui est couramment utilisé.

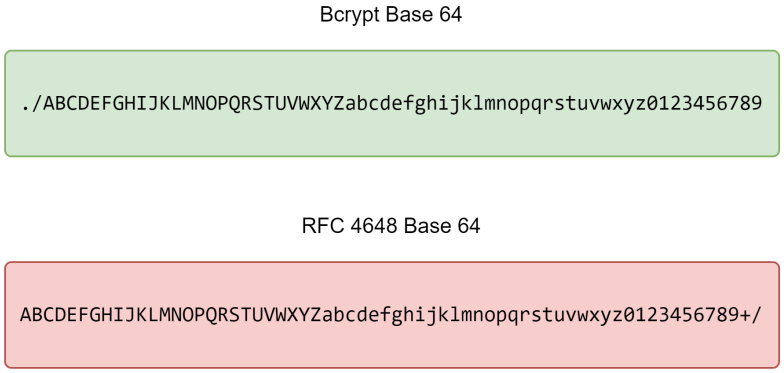


ILLUSTRATION 2.3 – Différence Base 64. Source : réalisé par Kandiah Abivarman

2. Simon JOSEFSSON. *The Base16, Base32, and Base64 Data Encodings*. Request for Comments RFC 4648. Num Pages: 18. Internet Engineering Task Force, oct. 2006. DOI : 10.17487/RFC4648. URL : https://datatracker.ietf.org/doc/rfc4648 (visité le 21/03/2024).

2.4. PROJET DE SEMESTRE

Votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte.

CHAPITRE 2 : SOLUTION UART

3.1. SYSTÈME DE PAQUET

a. Encodage Cobs

Votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte.

b. CRC

Votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte.

c. Transmission et Réception de Paquet

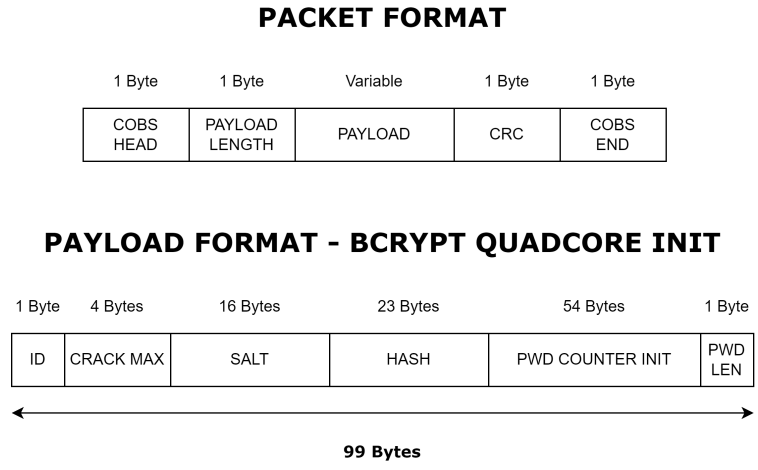


ILLUSTRATION 3.1 – Format de paquet. Source : réalisé par Kandiah Abivarman

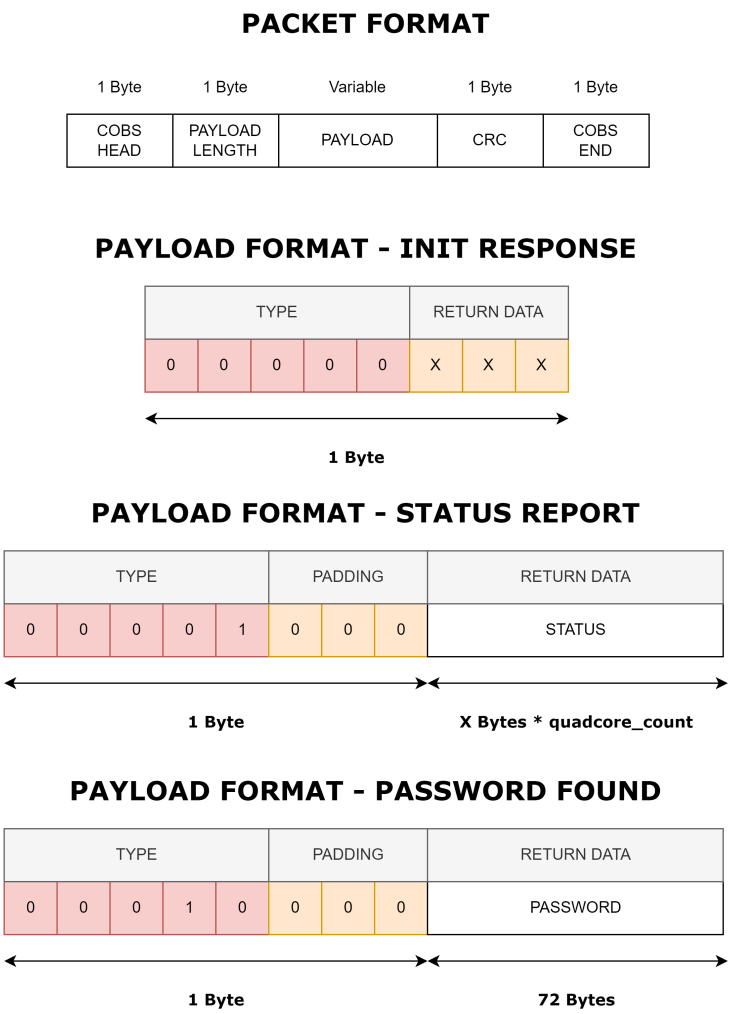


ILLUSTRATION 3.2 – Format de paquet. Source : réalisé par Kandiah Abivarman

3.2. IMPLÉMENTATION SUR FPGA

a. Architecture Logique

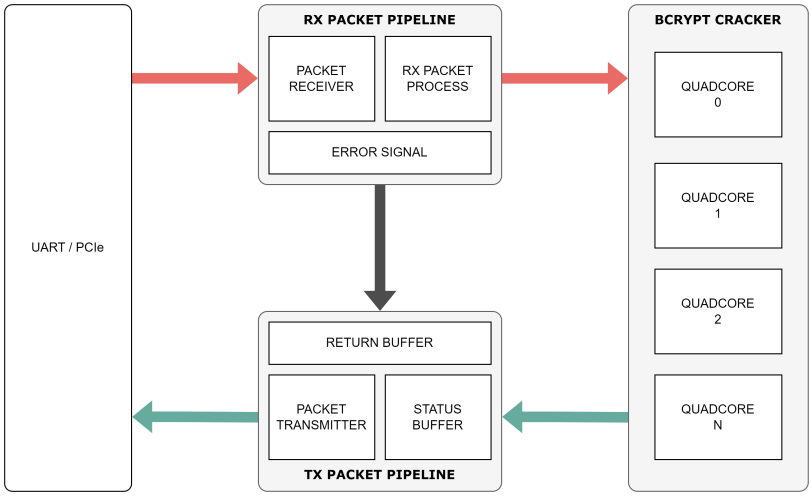


ILLUSTRATION 3.3 – Schéma système UART. Source : réalisé par Kandiah Abivarman

b. UART

c. Modifications Bcrypt Cracker

d. Implémentation - MOSI

d.1. Module - Packet Receiver

d.2. Module - RX Packet Process

d.3. Module - RX Packet Pipeline

e. Implémentation - MISO

e.1. Module - Packet Transmitter

e.2. Module - TX Packet Pipeline

f. Tests

f.1. Simulations

f.2. Vérification Hardware

3.3. INTERFACE UTILISATEUR

CHAPITRE 3 : SOLUTION PCIE

CHAPITRE 4 : MESURES ET PERFORMANCES

5.1. MESURES FPGA

5.2. MESURES CPU

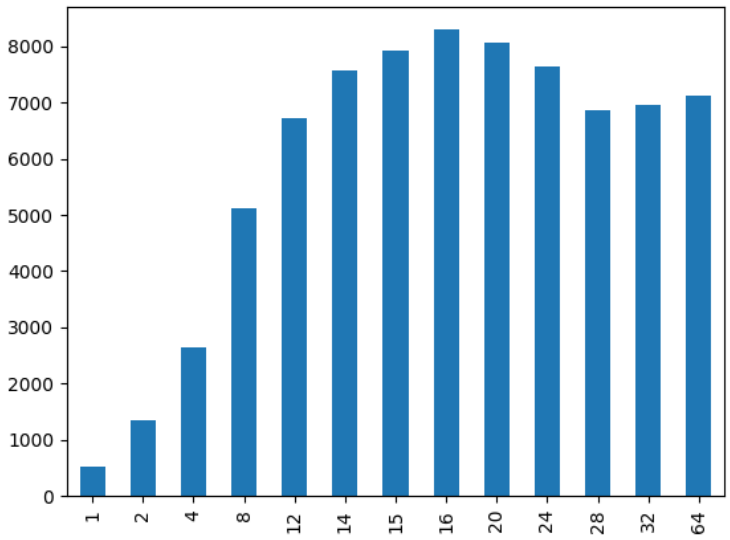


ILLUSTRATION 5.1 – Mesures Bcrypt CPU. Source : réalisé par Kandiah Abivarman

5.3. MESURES GPU

CONCLUSION

Votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte, votre texte.

ANNEXES

Imprimer idéalement cette page sur une page de couleur. Chaque annexe doit commencer sur une nouvelle page et doit être numérotée : Annexe 1 puis Annexe 2, etc.

ANNEXE 1 - REPO GITLAB

Lien du répértoire Gitlab :

`https://gitedu.hesge.ch/abivarma.kandiah/fpga_bruteforce_attack.`

RÉFÉRENCES DOCUMENTAIRES

Blowfish Algorithm with Examples. en-US. Section: Algorithms. Oct. 2019. URL : https://www.geeksforgeeks.org/blowfish-algorithm-with-examples/ (visité le 21/03/2024).

JOSEFSSON, Simon. *The Base16, Base32, and Base64 Data Encodings*. Request for Comments RFC 4648. Num Pages: 18. Internet Engineering Task Force, oct. 2006. DOI : 10.17487/RFC4648. URL : https://datatracker.ietf.org/doc/rfc4648 (visité le 21/03/2024).