

Rapport de Projet : CryptoHack

Victor Bailleul Sébastien Leglise

Université de Caen Normandie / ENSICAEN

Année 2025-2026



UNIVERSITÉ
CAEN
NORMANDIE



Plan de la présentation

- 1 Introduction
- 2 Challenge Diffie-Hellman
- 3 Challenge RSA
- 4 Bilan

Introduction

La plateforme CryptoHack

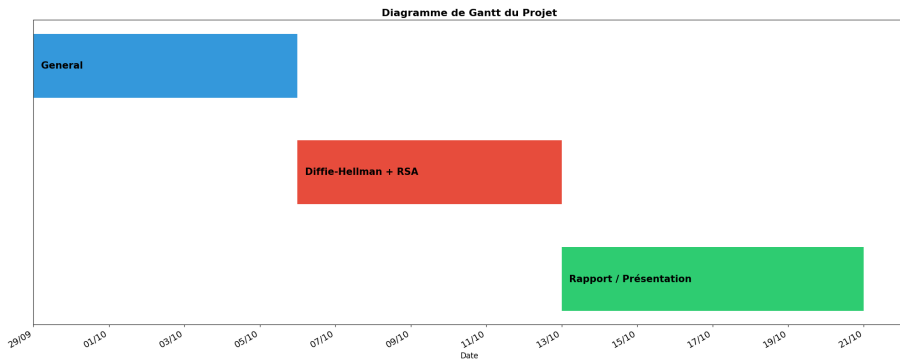
- Plateforme d'apprentissage dédiée à la cryptographie moderne.
- Approche pratique : résolution de défis à difficulté croissante.
- Objectif : enseigner les concepts fondamentaux et avancés.

L'intérêt des challenges de type CTF (*Capture The Flag*)

- **Principe** : *Gamification* de l'apprentissage en cybersécurité.
- **Bénéfices** :
 - Ancrage des connaissances par la pratique.
 - Développement de compétences techniques (analyse, résolution de problèmes).

Organisation du projet

Planification



Organisation du projet

Travailler en binôme

Répartition des tâches individuelles

	<i>Catégorie de challenges abordées</i>
<i>Victor</i>	General/ Encoding , General/ XOR , Diffie-Hellman
<i>Sébastien</i>	General/ Mathematics , General/ Data Formats , RSA

Travail commun et collaboration

L'ensemble du projet a été géré via un dépôt Git partagé sur GitHub. Cette approche nous a permis de :

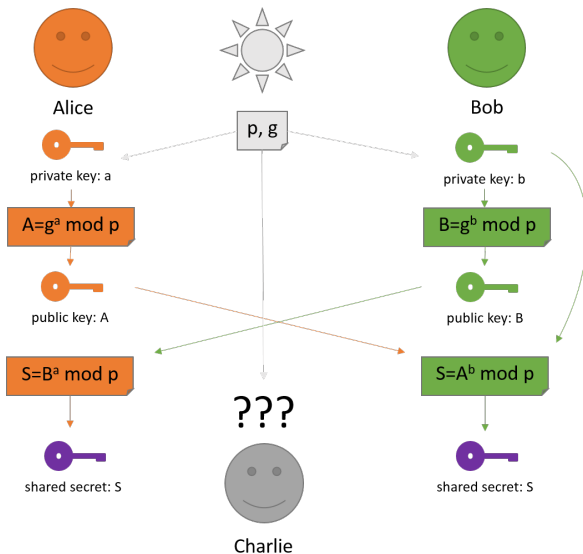
- **Centraliser le code** et les documents du projet.
- **Suivre les versions** pour éviter les conflits et les pertes de données.
- **Collaborer de manière asynchrone** sur les différentes parties du rapport et du code.

Challenge Diffie-Hellman

Man-in-the-middle

Diffie-Hellman : *Man-in-the-middle* / *Export grade*

Objectifs du challenge



Diffie-Hellman : *Man-in-the-middle* / *Export grade*

Méthode de résolution

```
➔ ~ nc socket.cryptohack.org 13379
```

Diffie-Hellman : *Man-in-the-middle* / *Export grade*

Méthode de résolution

```
➔ ~ nc socket.crytohack.org 13379
```

```
Intercepted from Alice: {"supported": ["DH1536", "DH1024", "DH512", "DH256", "DH128", "DH64"]}  
Send to Bob: {"supported" : ["DH64"]}
```

Diffie-Hellman : *Man-in-the-middle* / *Export grade*

Méthode de résolution

```
➔ ~ nc socket.cryptohack.org 13379
```

```
Intercepted from Alice: {"supported": ["DH1536", "DH1024", "DH512", "DH256", "DH128", "DH64"]}
```

```
Send to Bob: {"supported": ["DH64"]}
```

```
Intercepted from Bob: {"chosen": "DH64"}
```

```
Send to Alice: {"chosen": "DH64"}
```

Diffie-Hellman : *Man-in-the-middle* / Export grade

Méthode de résolution

```
→ ~ nc socket.crytohack.org 13379

Intercepted from Alice: {"supported": ["DH1536", "DH1024", "DH512", "DH256", "DH128", "DH64"]}
Send to Bob: {"supported" : ["DH64"]}
Intercepted from Bob: {"chosen": "DH64"}
Send to Alice: {"chosen": "DH64"}
Intercepted from Alice: {"p": "0xde26ab651b92a129", "g": "0x2", "A": "0x34d2c1aa8641c97b"}
Intercepted from Bob: {"B": "0x801150310e5e819e"}
Intercepted from Alice: {"iv": "cde29bdd9db7363da970aed984ef33fd", "encrypted_flag": "074ce389d"}
```

Diffie-Hellman : *Man-in-the-middle* / Export grade

Méthode de résolution

```
→ ~ nc socket.cryptohack.org 13379
Intercepted from Alice: {"supported": ["DH1536", "DH1024", "DH512", "DH256", "DH128", "DH64"]}
Send to Bob: {"supported": ["DH64"]}
Intercepted from Bob: {"chosen": "DH64"}
Send to Alice: {"chosen": "DH64"}
Intercepted from Alice: {"p": "0xde26ab651b92a129", "g": "0x2", "A": "0x34d2c1aa8641c97b"}
Intercepted from Bob: {"B": "0x801150310e5e819e"}
Intercepted from Alice: {"iv": "cde29bdd9db7363da970aed984ef33fd", "encrypted_flag": "074ce389d"}
```

Étapes clés de l'attaque

- Attaque de type *Man-in-the-middle* pour manipuler la communication.
- Forcer Alice et Bob à utiliser un **paramètre de sécurité faible** (64 bits).
- La faiblesse du paramètre rend la résolution du **logarithme discret** possible, ce qui nous donne accès au secret partagé.

Calcul de la clé secrète partagée

Grâce aux paramètres faibles, nous pouvons résoudre le logarithme discret pour trouver la clé privée d'Alice (a) :

$$a = \log_g(A) \pmod{p}$$

Puis, nous utilisons cette clé privée pour calculer le secret partagé (s) avec la clé publique de Bob (B) :

$$s = B^a \pmod{p}$$

Une fois le secret partagé s obtenu, il ne reste plus qu'à dériver la clé pour obtenir le flag :

`crypto{d0wn6r4d35_4r3_d4n63r0u5}`

Challenge RSA

Vote for Pedro

RSA : *Vote for Pedro*

Objectifs du challenge

Forger une signature RSA valide

```
N = 222666180573747099668189324252168063478925287698094909953312891282141426886989240483258181428723539813  
65579412014454755408382250595146521253104453521750474808668284973168861421563389271626210047747699495342394  
7983931362091470977935268797624752644573955277203987656815646922449168203831499488824798332964121759307658  
27088394780546657887715318520619975956990281083211485881847851847071572686490861748291817283574300353812240  
244016128014948997257285851816837389311516778842184578246761481980413932178576836278864071049153963852833649  
71333310072332846071665082777084028170668148862010444247566819193585999784628222347577  
  
e = 3  
  
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
sock.connect((host, port))  
data = sock.recv(1024)  
print(data)  
  
T = b'VOTE FOR PEDRO'  
c = bytes_to_long(T)  
  
def cube_root_2_pow(c, k_max):  
    s = c % 8  
    for k in range(1, k_max):  
        diff = s**3 - c  
        d = diff // (3*s**2)  
        t = (-d) % 8  
        s = s + t * (3*s**2)  
    return s  
  
s = cube_root_2_pow(c, len(T)*6 + 6)  
sign_hex = long_to_bytes(s).hex()  
  
payload = {  
    "option": "vote",  
    "vote": sign_hex  
}  
  
sock.send(json.dumps(payload).encode())  
flag = sock.recv(1024)  
print(flag)
```

- Voter pour Pedro sans clé privée
- Exploiter l'exposant faible $e = 3$
- Obtenir le flag

Attaque par racine cubique

$$\text{Signature } s = \sqrt[3]{\text{Message}}$$

$$s^3 \equiv \text{Message} \pmod{N}$$

- Message court = "VOTE FOR PEDRO"
- Pas de padding = vulnérabilité

Signature forgée validée !

```
#!/usr/bin/env python3
from Crypto.Util.number import bytes_to_long, long_to_bytes
from utils import listener

FLAG = "crypto{????????????????}"

class Challenge():
    def __init__(self):
        self.before_input = "Place your vote. Pedro offers a reward to anyone who votes for him!\n"

    def challenge(self, your_input):
        if 'option' not in your_input:
            return {"error": "You must send an option to this server"}

        elif your_input['option'] == 'vote':
            vote = int(your_input['vote'], 16)
            verified_vote = long_to_bytes(pow(vote, ALICE_E, ALICE_N))

            # remove padding
            vote = verified_vote.split(b'\x00')[-1]

            if vote == b'VOTE FOR PEDRO':
                return {"flag": FLAG}
            else:
                return {"error": "You should have voted for Pedro"}

        else:
            return {"error": "Invalid option"}

import builtins; builtins.Challenge = Challenge # hack to enable challenge to be run locally, see https://cryptohack.org/faq/slistener
listener.start_server(port=13375)
```

Flag obtenu

crypto{y0ur_v0t3_i5_my_v0t3}

- Vote accepté par le serveur

Bilan

Compétences Organisationnelles

- Planification de projet et gestion du temps.
- Travail collaboratif structuré via Git et GitHub.
- Documentation et rédaction de rapport scientifique.

Compétences Scientifiques

- Application pratique des concepts cryptographiques (Encoding, XOR, RSA, Diffie-Hellman).
- Analyse de protocoles et identification de leurs vulnérabilités.
- Mise en œuvre d'attaques sur des implémentations faibles (MitM, exposants faibles, etc.).
- Compréhension du lien entre sécurité théorique et défauts d'implémentation.

Des questions ?