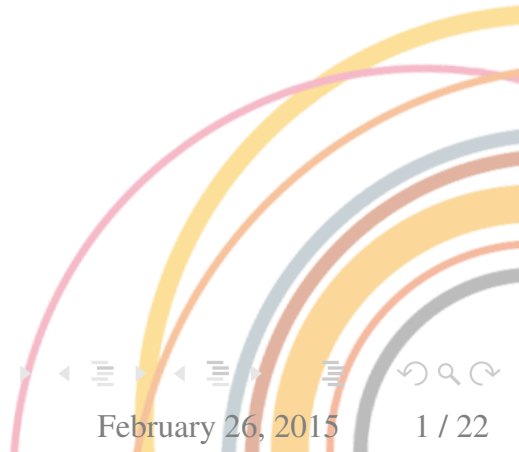
# Projet de recherche

## Procédés cryptographiques pour le vote électronique

Orestis Ioannou

Encadrant: M. Fabien Laguillaumie

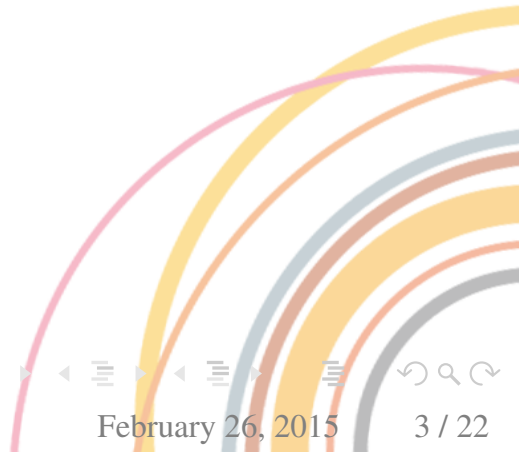


# Plan



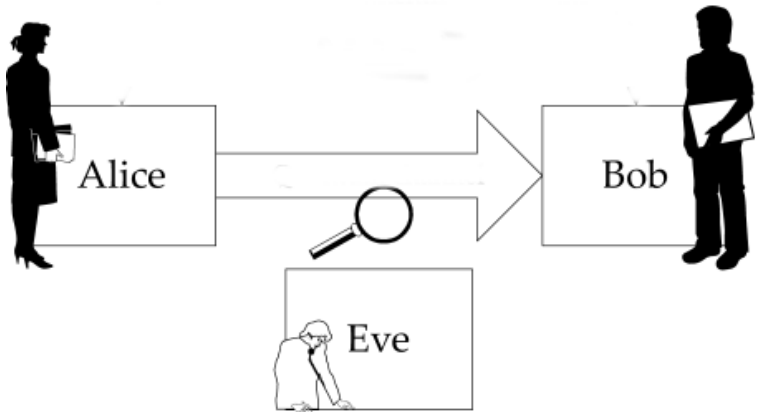
# Objectives

- Étudier les procédés cryptographiques utilisés pour le vote électronique.
- Développer un module implémentant ces procédés à l'aides des courbes elliptiques.
- But ultime: proposer une version d'Helios en utilisant les courbes elliptiques.



# Contraintes du vote électronique

- Confidentialité
- Vérifiabilité
- Incoercibilité
- Admissibilité

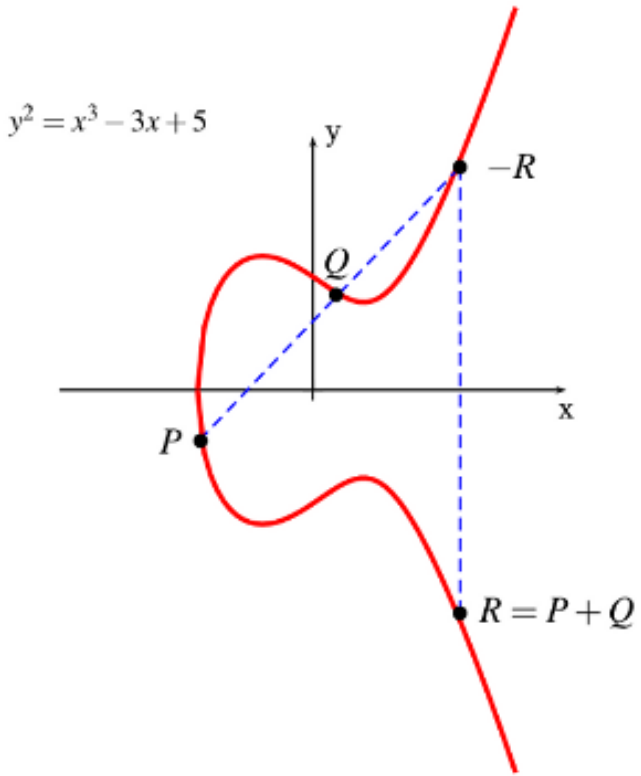



# Présentation d'Helios


- Créé par Ben Adida en 2008
- Protocole cryptographique basé sur ElGamal dans les corps finis
- Python et Django
- Avantages

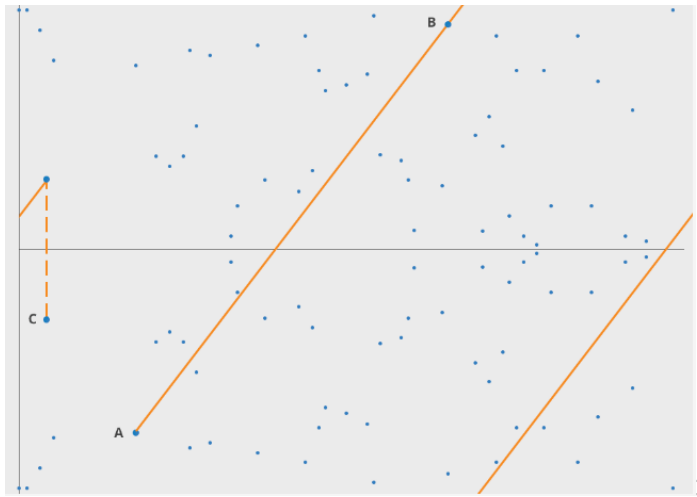
# Courbes elliptiques

- Courbe algébrique munie des **propriétés géométriques intéressantes**.
- La forme de Weierstrass: $y^2 = x^3 + ax + b$
- Soit point $P(x, y)$. Ses coordonnées vérifient l'équation de la courbe.
- Addition des points



# Problème du logarithme discret pour les courbes elliptiques

- Soit une courbe elliptique $(E)$ dans un corps fini $\mathbb{F}q$.
  $q$ un nombre premier assez grand (> 200 bits).
- $\#E \approx q$
- Problème du logarithme discret: soit $P(x,y)$ un point sur la courbe et $Q = nP$. Sachant $P$ et $Q$ il est très difficile de trouver $n$.
- Il n'existe pas d'algorithme efficace pour trouver $n$. Les meilleurs algorithmes aujourd'hui ont une **complexité de** $\sqrt{l}$ avec $l$ l'ordre du groupe (Baby Step Giant Step, Rho de Pollard).
- Pourquoi est-il intéressant pour la **cryptographie**?



# ElGamal à l'aide des courbes elliptiques

- Chiffrement **asymétrique et probabiliste**
- Soit un corps fini $\mathbb{F}q$ avec $q$ un nombre premier, E une courbe elliptique sur $\mathbb{F}q$, $P$ le point générateur et $l$ l'ordre du groupe.
- Clé Secrète: un nombre aléatoire $k$
- Clé publique $Q = kP$
- Encrypt(M) :

$$c = (R, S)$$

$$R = rP$$

$$S = M + rQ$$

ou $r$ est un nombre aléatoire.

- Decrypt(c):

$$S - kR = M + rkP - krP = M$$

# Propriétés intéressantes d'ElGamal

- Addition homomorphe. Soient deux chiffrés:

$$c_1 = Encrypt(M_1)$$

$$c_2 = Encrypt(M_2)$$

Alors

$$c_1\text{"+"}c_2 = Encrypt(M_1 + M_2)$$

- Re-chiffrement. Soit un chiffré:

$$c = (rP, M + rQ)$$

Reencrypt(c)

$$c_2 = (rP + r_2P, M + rQ + r_2Q) = (P(r + r_2), M + Q(r + r_2))$$

$r_2$ étant un nombre aléatoire.

- Avantages

# Recherche Exhaustive

Scénario: Soit une élection à deux candidats. Nous attribuons au premier candidat l'élément neutre $\mathbb{O}$ et au deuxième le point générateur $P$. L'addition homomorphe:

$$\ldots + \mathbb{O} + P + P + \mathbb{O} + \ldots$$

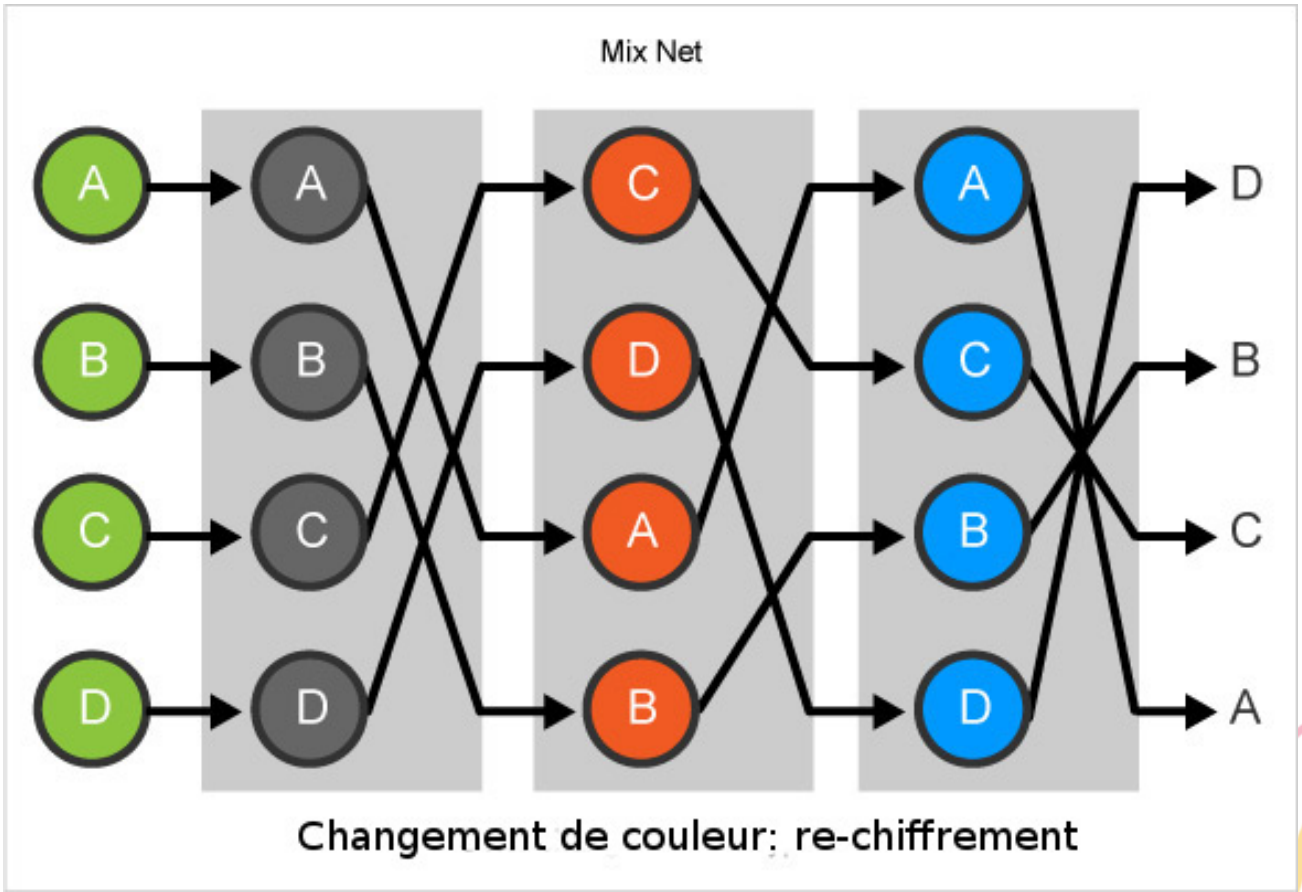
A la fin nous avons

$$nP$$

- Pourquoi est-on capable de retrouver $n$ par une recherche exhaustive?

# Mix network

- Intérêt?
- Permutation et re-chiffrement



# Preuves à connaissance nulle

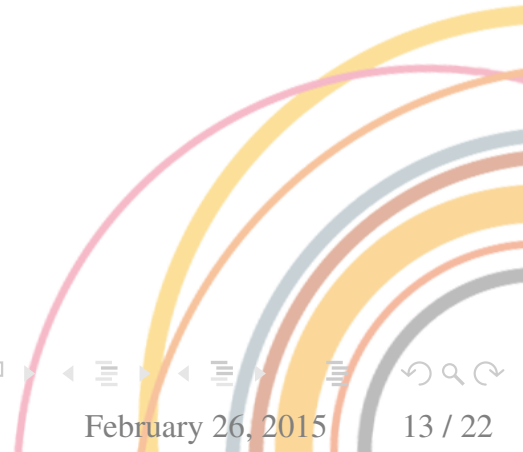
- Hypothèse d'un tuple Decisional Diffie-Hellman (DDH) $(P, tP, wP, wtP)$
- Preuve de Chaum Pedersen
- Problème du bourrage: que se passe-t-il en cas d'un utilisateur malhonnête qui décide d'envoyer un vote qui correspond à Encrypt(100) au lieu de Encrypt(0) ou Encrypt(1)? Addition Homomorphe:

$$\ldots + 0 + 1 + 0 + 1 + 0 + 100 + 1 + 0 + 1 + \ldots$$

- Preuve de disjonction: Encrypt(0) **OU** Encrypt(1)
- Preuve de re-chiffrement

# Module courbes elliptiques

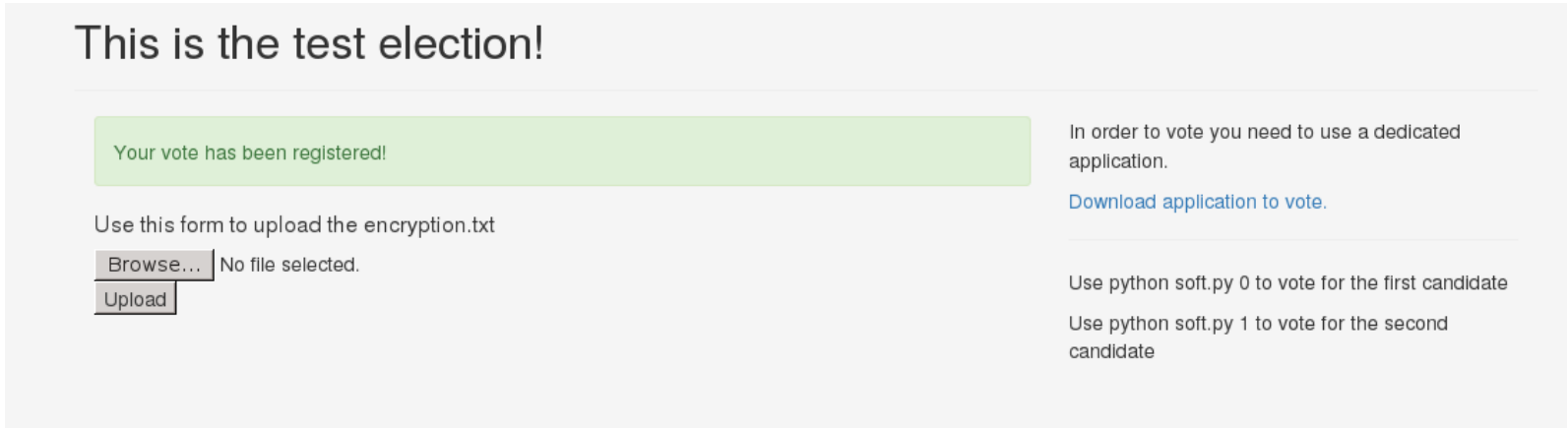
- Environ 1000 lignes de code Python
- Implémentation de plusieurs types des courbes
- ElGamal
- Preuves "zero-knowledge"
- Addition homomorphe
- Re-chiffrement

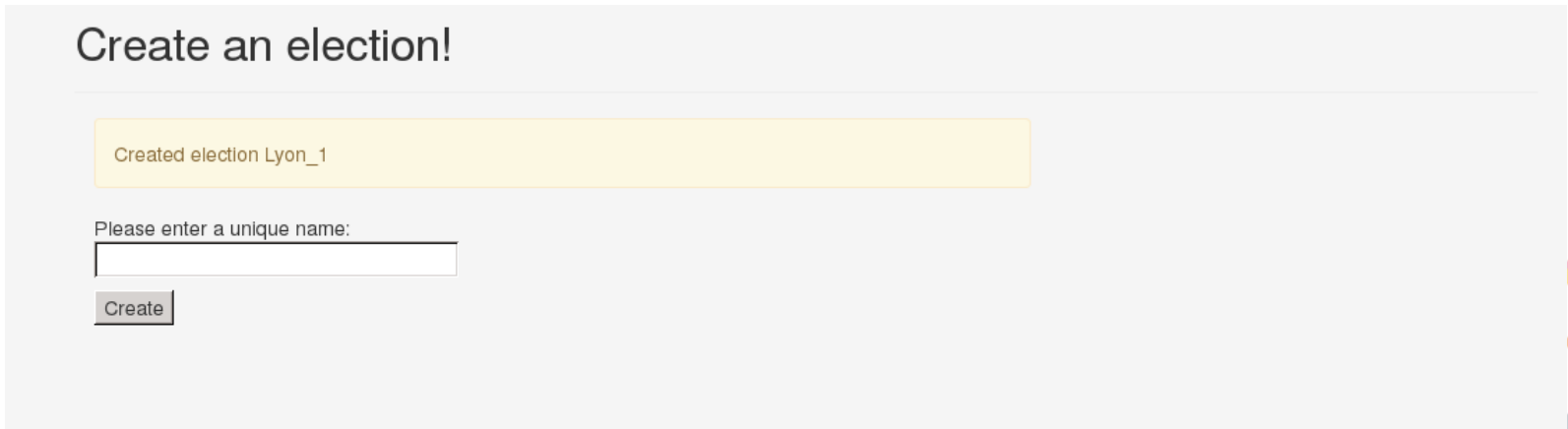


# Application web

- Tornado et Flask
- WebSockets
- Élections simultanées
- Chiffrement côté client
- Mixnets


# Démo


This is the test election!
Your vote has been registered!
Use this form to upload the encryption.txt
Browse...
No file selected.
Upload
In order to vote you need to use a dedicated application.
Download application to vote.
Use python soft.py 0 to vote for the first candidate
Use python soft.py 1 to vote for the second candidate


Create an election!
Created election Lyon_1
Please enter a unique name:
Create

# Performances

Courbe utilisée: "Curve2213"

$$y^2 = x^3 + 117050x^2 + x$$

$$q = 2^{221} - 3$$

Chiffrement et vérification du vote par l'urne: 1.12s

Déchiffrer un vote: 0.132s

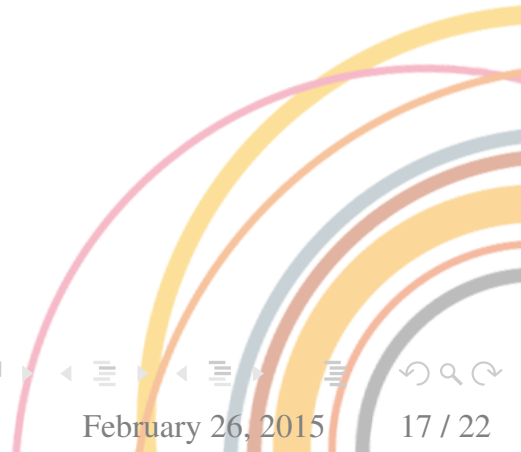
| Nb Votes | R.Exhaustive (s) | Mixnet (s) | WebSocket (s) |
|---|---|---|---|
| 10 | 0.106 | 8.918 | 0.041 |
| 1000 | 2.443 | 1183.747 | 0.137 |

# Conclusion

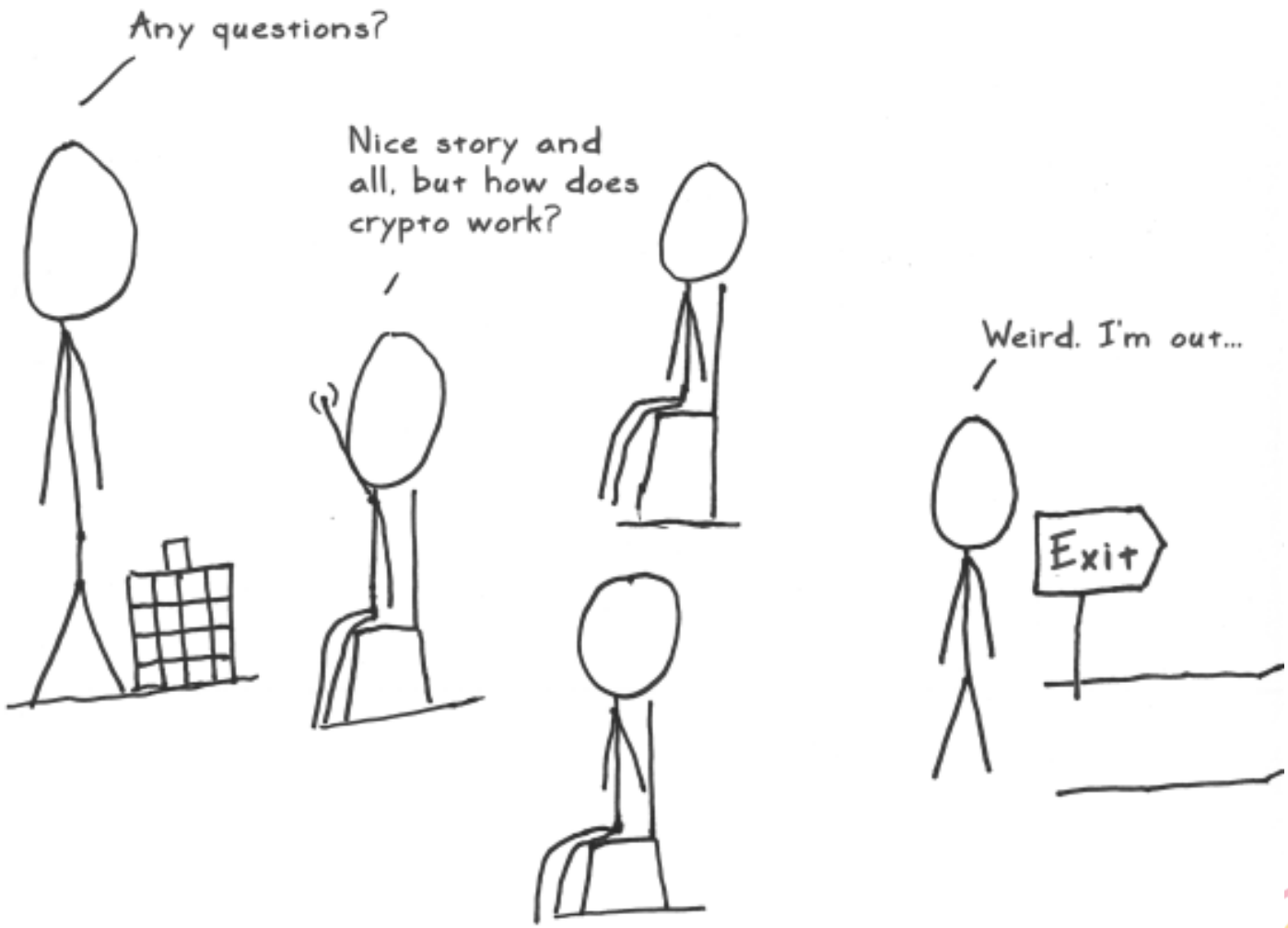
**Questions ouvertes**

- Preuve de permutation
- Authentification
- Élections avec plus de deux candidats
- Encodage de points sur la courbe

**Ressenti personnel**



Merci de votre attention!!!!!


Any questions?
Nice story and all, but how does crypto work?
Weird. I'm out...
Exit

# Preuve de Chaum-Pedersen

Soit $c$ un chiffré de $O$. Nous prouvons le tuple $(P, Q, W, U)$
avec $W = R$ et $U = S$.

1. Le "prouver" envoie le "commitment" $(a = sP, b = sQ)$, s étant un aléa.
2. Le "verifier" envoie un "challenge" $v$
3. Le "prouver" envoie $t = s + vr$

Le verifier vérifie alors deux égalités:

$$tP == a + qW$$

Ceci est vraie car $tP = (s + vr)P$ et $a + qW = sP + vrP = (s + vr)P$

$$tQ == b + qU$$

Aussi vraie car $tQ = (s + vr)kP$ et $b + qU = skP + vrkP = (s + vr)kP$

# Preuve de re-chiffrement

Soit un chiffré

$$c_1 = (R_1, S_1)$$

et

$$c_2 = (R_2, S_2)$$

son re-chiffrement.
On construit le tuple DDH suivant:

$$(P, Q, W, U)$$

avec $W = R_2 - R_1$ et $U = S_2 - S_1$.

# Preuve de disjonction

**Prouver** $c_1 \vee c_2$

1. Le prouver calcule $a_1 = sP$. Le "commitment" pour Encrypt(1) est $a_2 = Pt_2 - U_2 v_2$ avec $s_2, v_2 \xrightarrow{r} \mathbb{Z}_q$ et $U_2$ le chiffré de 1. Alors cette fois le "commitment" est le tuple $(a_1, a_2)$.
2. Le vérifier envoie un "challenge" $v$
3. Le "prouver" envoie le tuple $(v_1, v_2, t_1, t_2)$ avec $v_1 = v - v_2$.

Le vérifier a besoin de vérifier 3 preuves:

$$v == v_1 + v_2$$

$$t_1 P == a_1 + v_1 U_1$$

Cette preuve est identique à la première preuve de Chaum Pedersen

$$t_2 P == a_2 + v_2 U_2$$

Ceci est vrai car $a_2 + v_2 U_2 = t_2 P - U_2 v_2 + v_2 U_2 = t_2 P$.