process de l'encryption de la viewingKey

1-De secret network la clé est envoyée par le biais du wallet Kepler

2-le Front appel l'api Serenity Shield par le biais du protocole Https et doublement securisée par le SSL

3-L'API Serenity reçois la clé et la split en trois parties par le biais de Shamir Secret Sharing

les clés sont ainsi unique et crypté par ce processus

source(https://www.geeksforgeeks.org/shamirs-secret-sharing-algorithm-cryptography/#:~:text=Approach%3A%20The%20main%20idea%20behind,polynomial%20ax%20%2B%20by%20%3D%20c)

4-Nous Rajoutons une couche de cryptage par le biais d'une clé symetrique, donc nos trois parties sont cryptées et unique par le Shamir et en plus nous les cryptons

pour qu'elles ne soient pas lisibles par autrui. ce système est un processus à la volée, rien n'est stocké sur nos serveurs et bases de données.

5-Par le biais de AES (https://www.geeksforgeeks.org/advanced-encryption-standard-aes/)

et du protocole Rfc2898DeriveBytes

qui est utilisé pour produire une clé dérivée à partir d'une clé de base et d'autres paramètres.

Dans une fonction de dérivation de clé basée sur un mot de passe, la clé de base est un mot de passe

et les autres paramètres sont une valeur Salt et un nombre d'itérations.

L'api renvoi après encryption au Front-end

Les parties sont doublements encryptés.

Nos serveurs servent d'outils d'encryption ,

extrait de code:

public string Encrypt(string clearText)

{

string EncryptionKey = "XXXXXXX";

byte[] clearBytes = Encoding.Unicode.GetBytes(clearText);

using (Aes encryptor = Aes.Create())

{

Rfc2898DeriveBytes pdb = new Rfc2898DeriveBytes(EncryptionKey, new byte[] { 0x49, 0x76, 0x61, 0x6e, 0x20, 0x4d, 0x65, 0x64, 0x76, 0x65, 0x64, 0x65, 0x76 });

encryptor.Key = pdb.GetBytes(32);

encryptor.IV = pdb.GetBytes(16);

using (MemoryStream ms = new MemoryStream())

{

using (CryptoStream cs = new CryptoStream(ms, encryptor.CreateEncryptor(), CryptoStreamMode.Write))

{

cs.Write(clearBytes, 0, clearBytes.Length);

cs.Close();

}

clearText = Convert.ToBase64String(ms.ToArray());

}

}

return clearText;

}

public string Decrypt(string cipherText)

{

string EncryptionKey = "XXXXXX";

cipherText = cipherText.Replace(" ", "+");

byte[] cipherBytes = Convert.FromBase64String(cipherText);

using (Aes encryptor = Aes.Create())

{

Rfc2898DeriveBytes pdb = new Rfc2898DeriveBytes(EncryptionKey, new byte[] { 0x49, 0x76, 0x61, 0x6e, 0x20, 0x4d, 0x65, 0x64, 0x76, 0x65, 0x64, 0x65, 0x76 });

encryptor.Key = pdb.GetBytes(32);

encryptor.IV = pdb.GetBytes(16);

using (MemoryStream ms = new MemoryStream())

{

using (CryptoStream cs = new CryptoStream(ms, encryptor.CreateDecryptor(), CryptoStreamMode.Write))

{

cs.Write(cipherBytes, 0, cipherBytes.Length);

cs.Close();

}

cipherText = Encoding.Unicode.GetString(ms.ToArray());

}

}

return cipherText;

}

6-Une fois que le Front-end récupère ces trois clés cryptés , il génère les Nft sur la blockchain

7-les Nft sont ensuite envoyés dans les wallet leurs corespondants.