Onderzoek Signed Messages

# Introductie

Voor ons platform willen we een reputatie systeem ontwikkelen waarin wij kunnen laten zien hoe productief en effectief de delegates op ons platform zijn.

Om dit te meten op ons platform gaan we de productiviteit van delegates meten. Dit doen we om de delegates een accurate reputatiescore te geven. Dit is handig voor klanten omdat die dan precies kunnen zien hoe betrouwbaar of goed een delegate is. ook en om de klant alleen te laten betalen voor wat de delegate heeft gedaan op de sidechain van de klant. We kwamen tegen het probleem aan dat de identiteit (of delegate) per unieke sidechain uniek is, dit betekent dat we de reputatie niet makkelijk kunnen binden aan een account op de marketplace.

We hebben hierover gespard en in dit document staat hoe wij de aanpak om delegates te identificeren.

# Onderzoek

Tijdens het bezoek aan het LCU zijn we tot een oplossing gekomen. Het is mogelijk om in een transactie een stukje data toe te voegen. In dit stukje data kan een delegate zich identificeren.

In de Lisk documentatie staat dat iemand een signature op een transactie kan zetten deze signature bestaat uit een passphrase. Deze passphrase is de private key. Waar andere cryptocurrencies soms een grote hash hebben als private key heeft Lisk ervoor gekozen om met gebruik van [BIB39 mnemonic code](https://link.springer.com/referenceworkentry/10.1007%2F1-4020-0613-6_11624) een passphrase te genereren. Deze kan dus gebruikt worden om meerdere wallets te linken aan elkaar.

Dit gaat als volgt:

* Een delegate is eigenaar van wallet A en wallet B.
* Wallet A is een wallet waarmee hij een delegate is op een sidechain.
* Wallet B is een wallet die hij heeft bij de Lisk Marketplace.
* Hij wilt vervolgens bewijzen dat hij eigenaar van beide wallets is. Hij vraagt op de Lisk Marketplace website een unieke code waarmee hij zich kan bewijzen.
* De delegate stuurt vanaf wallet B een transactie naar de Lisk marketplace wallet voor identificatie. Hij betaalt een kleine bedrag om zich te registreren en de transactie fee.
* In deze transactie gebruikt Wallet B vervolgens de private key van wallet A om de data, die bestaat uit de unieke code, te versleutelen.
* Lisk Marketplace decodeert de versleutelde data met de public key van wallet B en controleert of de data is wat die moet zijn.
* Wanneer de unieke code juist is update het systeem de reputatie van de delegate met de productivity die hij heeft op de delegate node die hij op een andere sidechain heeft draaien en waarvan hij net heeft bewezen de eigenaar van te zijn.

### 

# Code

### <https://lisk.io/documentation/lisk-sdk/lisk-elements/packages/api-client/signatures.html>

### **encryptMessageWithPassphrase**

This encrypts a message under a recipient’s public key, using a passphrase to create a signature.

#### **Syntax**

encryptMessageWithPassphrase(message, passphrase, recipientPublicKey)

#### **Parameters**

message: The plaintext message to encrypt.

passphrase: The passphrase used to sign the encryption and ensure message integrity.

recipientPublicKey: The public key to be used in encryption.

#### **Return value**

object: The result of encryption. This contains the nonce and encryptedMessage, both in hex string format.

#### **Examples**

const encryptedMessage = cryptography.encryptMessageWithPassphrase(

'Ik ben lisk Delegate nummer 101!',

'robust swift grocery peasant forget share enable convince deputy road keep cheap',

'9d3058175acab969f41ad9b86f7a2926c74258670fe56b37c429c01fca9f2f0f'

);

/\* {

encryptedMessage: '7bef28e1ddb34902d2e006a36062805e597924c9885c142444bafb',

nonce: '5c29c9df3f041529a5f9ba07c444a86cbafbfd21413ec3a7',

} \*/

**decryptMessageWithPassphrase**

This decrypts a message that has been encrypted for a given public key using the corresponding passphrase:

#### **Syntax**

decryptMessageWithPassphrase(encryptedMessage, nonce, passphrase, senderPublicKey)

#### **Parameters**

encryptedMessage: The hex string representation of the encrypted message.

nonce: The hex string representation of the nonce used during encryption.

passphrase: The passphrase to be used in decryption.

senderPublicKey: The public key of the message sender, (this is used to ensure the message was signed by the correct person).

#### **Return value**

string: The decrypted message.

#### **Examples**

const decryptedMessage = cryptography.decryptMessageWithPassphrase(

'7bef28e1ddb34902d2e006a36062805e597924c9885c142444bafb',

'5c29c9df3f041529a5f9ba07c444a86cbafbfd21413ec3a7',

'robust swift grocery peasant forget share enable convince deputy road keep cheap',

'9d3058175acab969f41ad9b86f7a2926c74258670fe56b37c429c01fca9f2f0f'

); // 'Ik ben lisk Delegate nummer 101'