nomadic labs

Making confidential transactions on Tezos

Table of contents

Introduction

Introduction

The majority of public blockchains allow anyone with access to the Internet to have access to all operations on the chain.
Moreover, by knowing a user's address (from a payment, for example), anyone can:

- read the balance of their account
- see the transactions that they have issued and received since the account was created, as well as the amounts, recipients and issuers of the transactions.

These aspects may be incompatible[1] with the protection of users' privacy[2], as well as the right to be forgotten[3].

To remedy the situation, Tezos offers contracts based on the **Sapling** protocol. This protocol provides both **the protection of users' privacy and transparency with regard to the regulators**.

[1] https://www.eublockchainforum.eu/sites/default/files/reports/20181016_report_gdpr.pdf

[2] On this subject, in the European Union, the GDPR regulates the protection of natural persons with regard to the processing of personal data and the free circulation of such data.

[3] The right to be forgotten allows an individual to request the removal of information about them and that may harm them.

Definition of the main concepts

Definition of the main concepts

- **Shielded transaction:** A transaction is said to be shielded when an external observer cannot find out any data from it (including the amount, the recipient and the sender[1]).
- **Transparent transaction:** A transaction is said to be transparent when it is not shielded.
- **Shielded pool:** A shielded pool is a set of tokens whose number we can find out but whose internal transactions are shielded.
- **Smart-contract:** In a cryptocurrency, a smart contract is a self-managed account whose interactions are configured by a code and data. On each transaction to the address of a smart contract, its code determines changes to its data and any transactions issued by it.

[1] In practice, an external observer can see the sender if they call the contract.

Key management

Key management

Sapling is a cryptographic protocol that allows shielded transactions to be conducted, while ensuring that trusted third parties can view the transactions. Three elements need to be identified for this to happen:

- The **expense** key, which should be kept secret and which is used to sign transfers.
- The **viewing** key, which is derived from the expense key and allows transactions issued or received to be viewed (it is possible to create viewing keys that allow only transactions issued or transactions received to be viewed).
- Addresses that are used to receive transactions are all derived from the **expense** key. We can derive as many addresses as we wish, so two different users will not be able to know whether or not they are making a transfer to the same person.

Sapling is a cryptographic protocol that uses elliptic curves (Bls 12-381 and jubjub).

Demonstration

Example without monitoring of the transaction by a third party

In this example, Alice wants to send 8ℨ to Bob, without anyone else knowing.

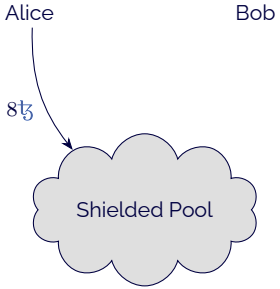
Alice

Bob

Example without monitoring of the transaction by a third party

In this example, Alice wants to send 8ꜩ to Bob, without anyone else knowing.

1. Alice begins by creating tokens in the shielded pool, sending 8ꜩ to the contract.



Example without monitoring of the transaction by a third party

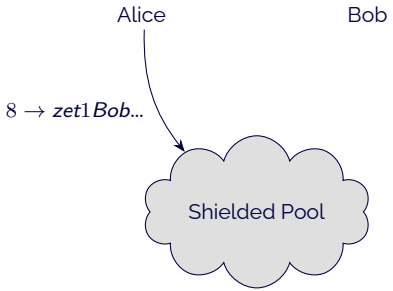
In this example, Alice wants to send 8ꜩ to Bob, without anyone else knowing.

1. Alice begins by creating tokens in the shielded pool, sending 8ꜩ to the contract.
2. Bob derives an address from his view key and sends it to Alice.

Example without monitoring of the transaction by a third party

In this example, Alice wants to send 8ꜩ to Bob, without anyone else knowing.

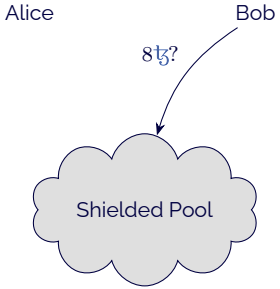
1. Alice begins by creating tokens in the shielded pool, sending 8ꜩ to the contract.
2. Bob derives an address from his view key and sends it to Alice.
3. Alice sends Bob the tokens through the shielded pool.



Example without monitoring of the transaction by a third party

In this example, Alice wants to send 8ꜩ to Bob, without anyone else knowing.

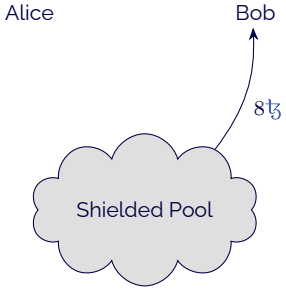
1. Alice begins by creating tokens in the shielded pool, sending 8ꜩ to the contract.
2. Bob derives an address from his view key and sends it to Alice.
3. Alice sends Bob the tokens through the shielded pool.
4. Bob can then claim the tokens that he has received.



Example without monitoring of the transaction by a third party

In this example, Alice wants to send 8ꜩ to Bob, without anyone else knowing.

1. Alice begins by creating tokens in the shielded pool, sending 8ꜩ to the contract.
2. Bob derives an address from his view key and sends it to Alice.
3. Alice sends Bob the tokens through the shielded pool.
4. Bob can then claim the tokens that he has received.
5. The contract itself then sends 8ꜩ and withdraws the corresponding tokens from the shielded pool.



Example without monitoring of the transaction by a third party

In this example, Alice wants to send 8ꜩ to Bob, without anyone else knowing.

1. Alice begins by creating tokens in the shielded pool, sending 8ꜩ to the contract.
2. Bob derives an address from his view key and sends it to Alice.
3. Alice sends Bob the tokens through the shielded pool.
4. Bob can then claim the tokens that he has received.
5. The contract itself then sends 8ꜩ and withdraws the corresponding tokens from the shielded pool.

In this scenario, someone could notice that Alice is sending 8ꜩ and that Bob is also withdrawing 8ꜩ. This might make it possible to guess that there has been a transaction between the two, especially if the deposit and the withdrawal are temporally close.

Ideally, Bob should not withdraw his tokens and should use them to pay transactions.

To increase his anonymity, Bob can ensure that he never interacts with the contract to deposit or withdraw tez.

Example without monitoring of the transaction by a third party

In this example, Alice wants to send 8ꜩ to Bob, without anyone else knowing.

1. Alice begins by creating tokens in the shielded pool, sending 8ꜩ to the contract.
2. Bob derives an address from his view key and sends it to Alice.
3. Alice sends Bob the tokens through the shielded pool.
4. Bob can then claim the tokens that he has received.
5. The contract itself then sends 8ꜩ and withdraws the corresponding tokens from the shielded pool.

In this scenario, someone could notice that Alice is sending 8ꜩ and that Bob is also withdrawing 8ꜩ. This might make it possible to guess that there has been a transaction between the two, especially if the deposit and the withdrawal are temporally close.

Ideally, Bob should not withdraw his tokens and should use them to pay transactions.

To increase his anonymity, Bob can ensure that he never interacts with the contract to deposit or withdraw tez.

This example is a simple case of the use of Sapling.

It is also possible to allow a third party, a regulator for example, to control all transactions of the Sapling contract.

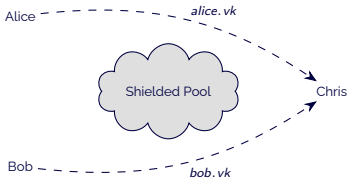
Example with monitoring of the transaction by a third party (1/2)

In this scenario, only Chris's address can send transactions (apart from deposits and withdrawals) to the shielded pool:

Example with monitoring of the transaction by a third party (1/2)

In this scenario, only Chris's address can send transactions (apart from deposits and withdrawals) to the shielded pool:

1. Alice and Bob identify themselves to Chris and send him their respective view keys.[1]



[1] This may have been done well before the current transaction.

Example with monitoring of the transaction by a third party (1/2)

In this scenario, only Chris's address can send transactions (apart from deposits and withdrawals) to the shielded pool:

1. Alice and Bob identify themselves to Chris and send him their respective view keys.[1]
2. Alice creates tokens in the shielded pool, sending 8ꜩ to the contract.

[1] This may have been done well before the current transaction.

Example with monitoring of the transaction by a third party (1/2)

In this scenario, only Chris's address can send transactions (apart from deposits and withdrawals) to the shielded pool:

1. Alice and Bob identify themselves to Chris and send him their respective view keys.[1]
2. Alice creates tokens in the shielded pool, sending 8 ꜩ to the contract.
3. Bob derives an address from his view key and sends it to Alice.[2]

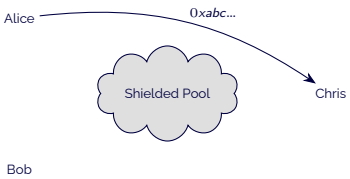
[1] This may have been done well before the current transaction.

[2] Alice could also address Chris because he has access to Bob's key and so may also be able to derive an address from it.

Example with monitoring of the transaction by a third party (1/2)

In this scenario, only Chris's address can send transactions (apart from deposits and withdrawals) to the shielded pool:

1. Alice and Bob identify themselves to Chris and send him their respective view keys.[1]
2. Alice creates tokens in the shielded pool, sending 8ꜩ to the contract.
3. Bob derives an address from his view key and sends it to Alice.[2]
4. Alice forges a transaction of 8ꜩ to Bob and sends it to Chris.



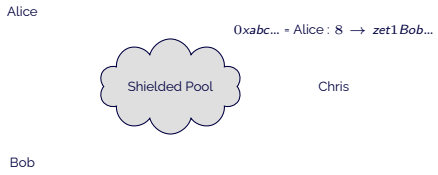
[1] This may have been done well before the current transaction.

[2] Alice could also address Chris because he has access to Bob's key and so may also be able to derive an address from it.

Example with monitoring of the transaction by a third party (1/2)

In this scenario, only Chris's address can send transactions (apart from deposits and withdrawals) to the shielded pool:

1. Alice and Bob identify themselves to Chris and send him their respective view keys.[1]
2. Alice creates tokens in the shielded pool, sending 8ꜩ to the contract.
3. Bob derives an address from his view key and sends it to Alice.[2]
4. Alice forges a transaction of 8ꜩ to Bob and sends it to Chris.
5. Chris notes, from Alice's keys, that it relates to a transaction coming from Alice to an address.



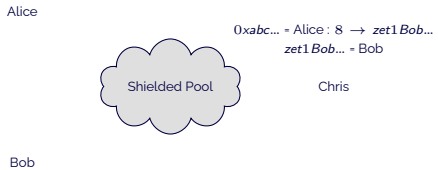
[1] This may have been done well before the current transaction.

[2] Alice could also address Chris because he has access to Bob's key and so may also be able to derive an address from it.

Example with monitoring of the transaction by a third party (1/2)

In this scenario, only Chris's address can send transactions (apart from deposits and withdrawals) to the shielded pool:

1. Alice and Bob identify themselves to Chris and send him their respective view keys.[1]
2. Alice creates tokens in the shielded pool, sending 8ꜩ to the contract.
3. Bob derives an address from his view key and sends it to Alice.[2]
4. Alice forges a transaction of 8ꜩ to Bob and sends it to Chris.
5. Chris notes, from Alice's keys, that it relates to a transaction coming from Alice to an address.
6. He can then verify, using Bob's key, that this address is derived from Bob's key.



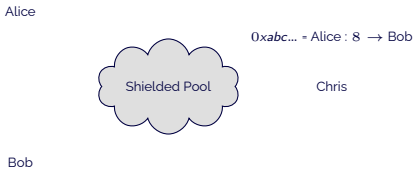
[1] This may have been done well before the current transaction.

[2] Alice could also address Chris because he has access to Bob's key and so may also be able to derive an address from it.

Example with monitoring of the transaction by a third party (1/2)

In this scenario, only Chris's address can send transactions (apart from deposits and withdrawals) to the shielded pool:

1. Alice and Bob identify themselves to Chris and send him their respective view keys.[1]
2. Alice creates tokens in the shielded pool, sending 8 ꜩ to the contract.
3. Bob derives an address from his view key and sends it to Alice.[2]
4. Alice forges a transaction of 8 ꜩ to Bob and sends it to Chris.
5. Chris notes, from Alice's keys, that it relates to a transaction coming from Alice to an address.
6. He can then verify, using Bob's key, that this address is derived from Bob's key.
7. Both keys show him the amount. He then knows that it relates to a transaction of 8 tokens from Alice to Bob.



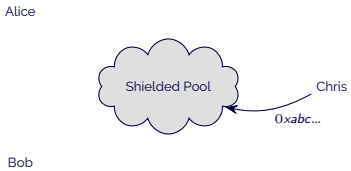
[1] This may have been done well before the current transaction.

[2] Alice could also address Chris because he has access to Bob's key and so may also be able to derive an address from it.

Example with monitoring of the transaction by a third party (1/2)

In this scenario, only Chris's address can send transactions (apart from deposits and withdrawals) to the shielded pool:

1. Alice and Bob identify themselves to Chris and send him their respective view keys.[1]
2. Alice creates tokens in the shielded pool, sending 8 ꜩ to the contract.
3. Bob derives an address from his view key and sends it to Alice.[2]
4. Alice forges a transaction of 8 ꜩ to Bob and sends it to Chris.
5. Chris notes, from Alice's keys, that it relates to a transaction coming from Alice to an address.
6. He can then verify, using Bob's key, that this address is derived from Bob's key.
7. Both keys show him the amount. He then knows that it relates to a transaction of 8 tokens from Alice to Bob.
8. He decides that this transaction is lawful and sends the transaction to the contract.



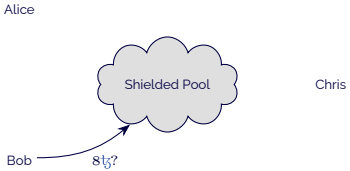
[1] This may have been done well before the current transaction.

[2] Alice could also address Chris because he has access to Bob's key and so may also be able to derive an address from it.

Example with monitoring of the transaction by a third party (1/2)

In this scenario, only Chris's address can send transactions (apart from deposits and withdrawals) to the shielded pool:

1. Alice and Bob identify themselves to Chris and send him their respective view keys.[1]
2. Alice creates tokens in the shielded pool, sending 8ꜩ to the contract.
3. Bob derives an address from his view key and sends it to Alice.[2]
4. Alice forges a transaction of 8ꜩ to Bob and sends it to Chris.
5. Chris notes, from Alice's keys, that it relates to a transaction coming from Alice to an address.
6. He can then verify, using Bob's key, that this address is derived from Bob's key.
7. Both keys show him the amount. He then knows that it relates to a transaction of 8 tokens from Alice to Bob.
8. He decides that this transaction is lawful and sends the transaction to the contract.
9. Bob can then claim the tokens that he has received and receive his 8ꜩ.



[1] This may have been done well before the current transaction.

[2] Alice could also address Chris because he has access to Bob's key and so may also be able to derive an address from it.

Example with monitoring of the transaction by a third party (1/2)

In this scenario, only Chris's address can send transactions (apart from deposits and withdrawals) to the shielded pool:

1. Alice and Bob identify themselves to Chris and send him their respective view keys.[1]
2. Alice creates tokens in the shielded pool, sending 8ꜩ to the contract.
3. Bob derives an address from his view key and sends it to Alice.[2]
4. Alice forges a transaction of 8ꜩ to Bob and sends it to Chris.
5. Chris notes, from Alice's keys, that it relates to a transaction coming from Alice to an address.
6. He can then verify, using Bob's key, that this address is derived from Bob's key.
7. Both keys show him the amount. He then knows that it relates to a transaction of 8 tokens from Alice to Bob.
8. He decides that this transaction is lawful and sends the transaction to the contract.
9. Bob can then claim the tokens that he has received and receive his 8ꜩ.

[1] This may have been done well before the current transaction.

[2] Alice could also address Chris because he has access to Bob's key and so may also be able to derive an address from it.

Example with monitoring of the transaction by a third party (2/2)

This example shows the case where Chris deals with validating the compliance of operations, potentially because he is answerable for what happens in the contract, or even because he is responsible for protecting Alice (and the other members of the contract) against potential fraudulent transactions. Furthermore, if he decides that an operation is not valid, he may decide to contact Alice to tell her about it and ask her for more information concerning this transaction, or quite simply to verify that she has not had her wallet stolen and that she is indeed the originator of this transaction.

Chris is not necessarily a human; he may also be a simple secure oracle[1] that has view keys and automatically conducts transactions where the parties are known.

In both examples, we have assumed that the tokens represented tez. This is not mandatory and we might suppose that the tokens represented any other fungible asset (financial assets, raw materials, etc.). In this case, the regulator could also manage the deposits and withdrawals. There would no longer be any direct interaction between the actual asset and its representation on the contract (as we could do with the tez). The contract **might then have an evidential value** in the event of a dispute.

[1] An oracle is a program external to the blockchain, generally standalone, the role of which is to make calls to a smart contract to give it orders according to parameters that the smart contract cannot or does not need to control.

nomadic labs

Let's keep the conversation going

https://tezos.com
https://developers.tezos.com
https://tezos.gitlab.io/alpha/sapling.html