Silk Pay: Secure Send/Receive Architecture

Sutera Duniya
shadeprotocol.io

Abstract. One of the key barriers of adoption for cryptocurrency transactions is the transfer of large amounts of capital between entities. Test transactions are a popular mechanism by which users check to make sure they are sending to the correct counterparty by sending a non-consequential amount of cryptocurrency to a target address. However, this does not create a true sender/receiver confirmation system as every "fresh" transaction can create a risk of destination error user input during the new send. Additionally, transactions on blockchains are traditionally completely transparent - exposing users' privacy and transaction history.

Enter Silk Pay - a privacy-preserving payment application built on Secret Network that introduces a new sender and receiver confirmation architecture that empowers senders to escrow capital in a contract while awaiting intended counterparties to confirm the inbound transaction by interacting with the escrow contract. Using the send request and receive request architecture, Silk Pay ensures outbound capital both safely and consistently sent to the correct location, enabling peace of mind and usability for everyday users.

Silk Pay

Silk Pay is a simple payment application focused on the SNIP-20 token sending and receiving user experience on Secret Network. The primary goal of Silk Pay is to bring Silk, the privacy-preserving stablecoin of Shade Protocol, to everyday end users. The initial implementation of Silk Pay is focused on a web experience, with future iterations to be built directly on mobile. Silk Pay is uniquely differentiated because it solves the jarring and risky "wire-transfer" paradigm of the majority of Web3 user-to-user transactions. Silk Pay solves this by giving users a varying degrees of sending and receiving options:

- Send Request
- Receive Request
- Direct Transfer

Direct Transfer (DT) is the traditional method of sending funds using the traditional wire-transfer architecture. With *DT*, users specify X amount of funds to be sent to Y address. Upon execution of the transaction, if the destination address is incorrect or improperly input, there is no recourse. Note that there is no transaction confirmation from the counter-party prior to funds being sent with the *DT* architecture. All double-checking has to be done in advance of the transaction - introducing risks as each send transaction user interaction carries its own set of risks for potential mis-step. To date, *DT* is the standard means by which users send funds.

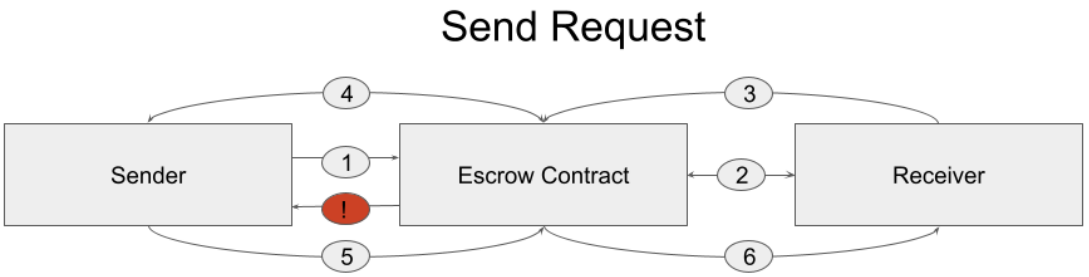
This is what the **send request** and **receive request** architecture is created to solve. Using escrow contract architecture, users can privately send X amount of funds to an encrypted destination address using a multistep process that involves confirmation interactions between both the sender, receiver, and escrow contract. As such, this architecture ensures that senders only ever send funds to correct addresses that are confirmed but the intended counterparty. Additionally, request infrastructure allows users the flexibility to privately request funds from other users. Eventually, Silk Pay will also include a system of monikers and registered addresses to make the initiation of a send or receive request even easier. While vetted addresses can remove the need for *send request* methodology (as send request is essentially a destination

verification system), there will always be the need for *send request* architecture for first time transactions. With the incorporation of monikers and registered addresses, *receive requests* become an even more seamless UI/UX experience.

Send Request

Send Request (SR) architecture is a six part process (2 parts for sender, 1 part for receiver, 3 parts contract) that ensures the safe arrival of funds to intended destinations via a system of confirmations between the sender, receiver, and escrow contract. With the *SR* model, senders await confirmation from an escrow contract that interacts with the intended destination address.

Traditional Web2 payment request models involve users requesting a certain amount of funds from a counterparty, which the counterparty has the option to confirm. *SR* flips the traditional request model by instead having the sender initiate a "send ask" before funds are sent. A *send ask* is viewable by the potential counterparty, who can then create an inbound confirmation that is viewable by the sender. Upon seeing the inbound confirmation, the sender is guaranteed that the target destination address has been properly targeted, allowing the sender to safely perform an *execute send*.



1. User sends funds and intended destination address to escrow contract - creating a **send ask**
2. Receiver queries escrow contract for **send ask**, confirming there is an inbound transaction
3. Receiver creates an **inbound confirmation** for the escrow contract in response to the **send ask**
4. Sender queries escrow contract checking for a corresponding **inbound confirmation**
5. Sender creates **execute send** command for the escrow contract after receiving **inbound confirmation**
6. Escrow contract sends X funds to the receiver

! Sender can retract funds at will

SR User Story

- Bob creates a *send ask*, which is first staged in the *escrow contract*
    - 100 SCRT
    - secret1zfk9yoptw7nlnwc4r66d84rgc3cqpd7hynczgq as the address he believes to be Alice
- Alice queries the escrow contract, expecting a *send ask* for 100 SCRT
- Alice does not see a *send ask*, and let's Bob know asynchronously
- Bob **retracts funds from the escrow contract** and destroys the original *send ask*
- Bob sends another *send ask* to the escrow contract, realising the address was wrong
    - 100 SCRT
    - secret1zfkzyzptw7nlnwc4r66d84rgc3cqpd7hynczgq as the new address he believes to be Alice's
- Alice queries the escrow contract, and sees the incoming *send ask*
- Alice creates an "inbound confirmation" that gets sent to the escrow contract
    - Alice can send an asynchronous message to Bob that *send ask* was received
- Bob queries the escrow contract, and sees the *inbound confirmation* for his potential transaction
- Bob performs the *execute send*, funds are sent to Alice

SR Risks

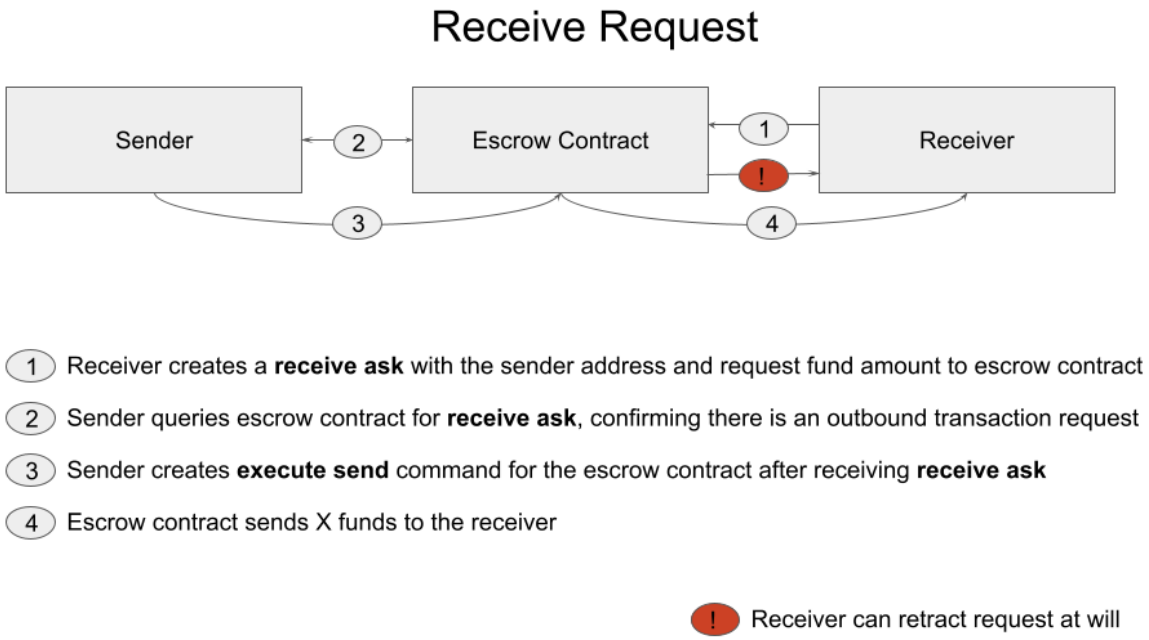
There is a potential risk that Bob potentially uses an address of another user who is not Alice, and that the malicious user would create an *inbound confirmation* for the escrow contract in hopes that Bob would perform an *execute send* after seeing the *inbound confirmation*. There are two safeguards against this:

- Possible addresses on Secret Network are 2^160 = 1461501637330902918203684832716283019655932542976 = 1.4615e+48. The odds that you accidentally put an address in that has an existing counterparty that is prepared to send an inbound confirmation message has less likelihood than a user accidentally sending to the wrong person on Venmo or PayPal by an order of magnitude.
- Due to Silk Pay being focused on A to B intermediary, it is highly unlikely that a Send Request is sent to a counterparty that does not have some form of asynchronous communication coordinating on the sending and receiving. That is to say, it will be highly obvious when a send-ask has not been received, with responsibility ultimately resting squarely on the sender for the final execution of the *execute send*.

In order to give senders maximum amount of flexibility, Senders will have the ability to perform an *execute send* prior to receiving an inbound confirmation from the receiver. Note that this will be highly discouraged within a UI/UX experience, but ultimately will be up to the user.

Receive Request

In addition to SR, there is also the **Receive Request** (RR) architecture which many users are already familiar with in Web2 that mirrors the traditional payment request operations.



RR User Story

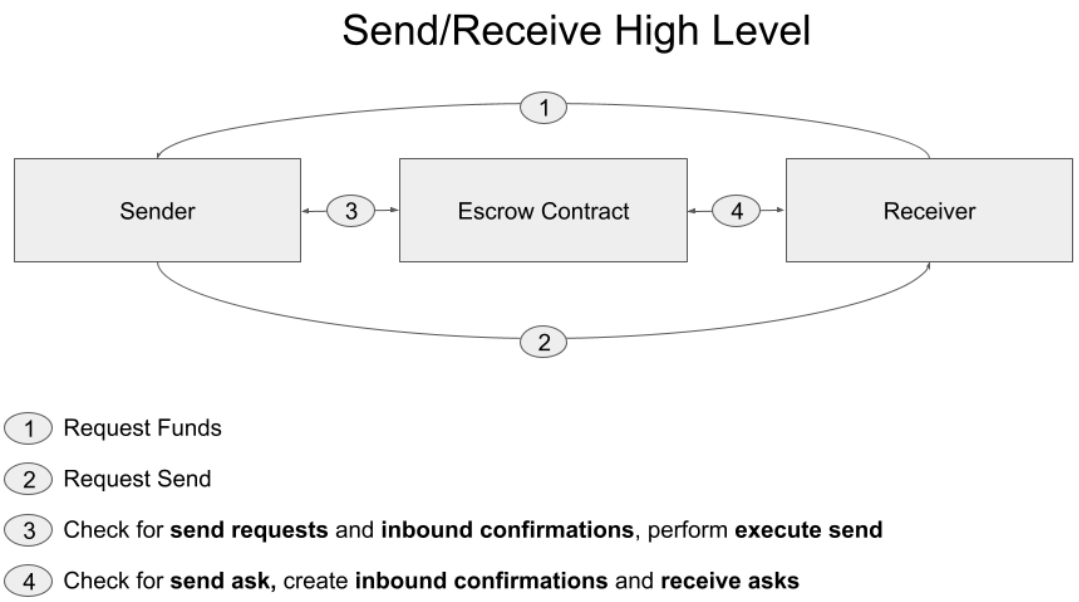
- Bob creates a *receive ask*, which is first staged in the escrow contract
    - Origin address
    - 100 SCRT
    - secret1zfk9yoptw7nlnwc4r66d84rgc3cqpd7hynczgq as the address he believes to be Alice
- Alice queries the escrow contract and sees the *receive ask* for 100 SCRT as well as which address created the *receive ask*
    - Alice can asynchronously check with Bob that he is the owner of the address and that he made the request
- Alice performs an *execute send* on the *receive request*
- Bob receives his requested funds from Alice

If Alice let's Bob know that she did not get the *receive ask*, Bob can retract the *receive ask* that was originally created by him. One of the risks of the Receive Request architecture is users could spam popular addresses requesting funds. Recourse for this behaviour is spam filtering on the front-end where users will only see *receive asks* from addresses they have already registered on the escrow contract as "contacts".

S/R High Level

The end result of Send/Receive architecture is a simplified experience that empowers users to safely send funds to new counterparties in a controllable manner that is managed by an escrow privacy-preserving smart contract. Users will still have the flexibility to send funds "wire-transfer" style with Silk Pay, but this will generally be discouraged (unless an "address book" or "moniker" contact is used). Cryptocurrency transactions require both privacy,

security, and usability. Silk Pay using this brand new Sender Request architecture will help bring Web3 comfortably to Web2 users.



Silk Pay Fees

Silk Pay is a key primitive of Shade Protocol - empowering an entire suite of privacy-preserving DeFi applications. Direct transfer payments will not be charged any additional fees, as this is simply a “normal” transaction by blockchain definition. Whenever a SR or RR is generated or executed upon, a small additional flat rate gas fee will be generated and sent to the Shade Protocol treasury address - creating an additional revenue stream for Shade Protocol SHD stakers.

Conclusion

The next generation of payment applications must ensure users have a way to safely send to new addresses with large amounts of capital. Current architecture in blockchain is essentially a blind wire transfer with no recourse for mistakes. By leveraging Secret Network’s privacy via secret contracts, Silk Pay provides a best in class experience that is currently unmatched by any other payment verification platform on Web3.