LONG HASH HACKATHON

IN BLOCKCHAIN WE TRUST

TOKYO

Chao Liu, Anran Li, Anakorn Kyavatanakij

InfoSec Lab @ Peking University

YET ANOTHER ATOMIC SWAP

李安然 刘超 陈继业

OUR PROJECT CONSISTS OF 3 PARTS:

- Research
- Design
- Implementation

PART 1

BACKGROUND

CRYPTOCURRENCY EXCHANGE

- There are strong incentives and requirement for cryptocurrency (aka. token) exchange
- The situation is easy when using third party escrow services, examples are Alipay and PayPal
- However, difficulty rises as in decentralized world where no single trusted third party exists
- This led to some “decentralized” schemes being proposed

ATOMIC SWAP AT A GLANCE

- The process can be seen as to enforce atomic swap between two involving parties, i.e., exchange should behave as an atomic operation with no intermediate state
- Several properties can be defined to compare different atomic swap schemes, such as safety, liveness, fairness, efficiency, etc.
- The key to achieving an atomic swap is to **make the act of spending one side of the swap to also reveal information that allows the other party to spend their side**. Additionally, the protocol needs to **make sure that regardless of either party aborting the protocol, both parties are able to recover their capital.**

COMMON TECHNOLOGIES

Below are the common technologies used to implement Atomic Swap:

1. Branched Transaction Scripts

2. HashLock (Used to reveal the secret when spending)

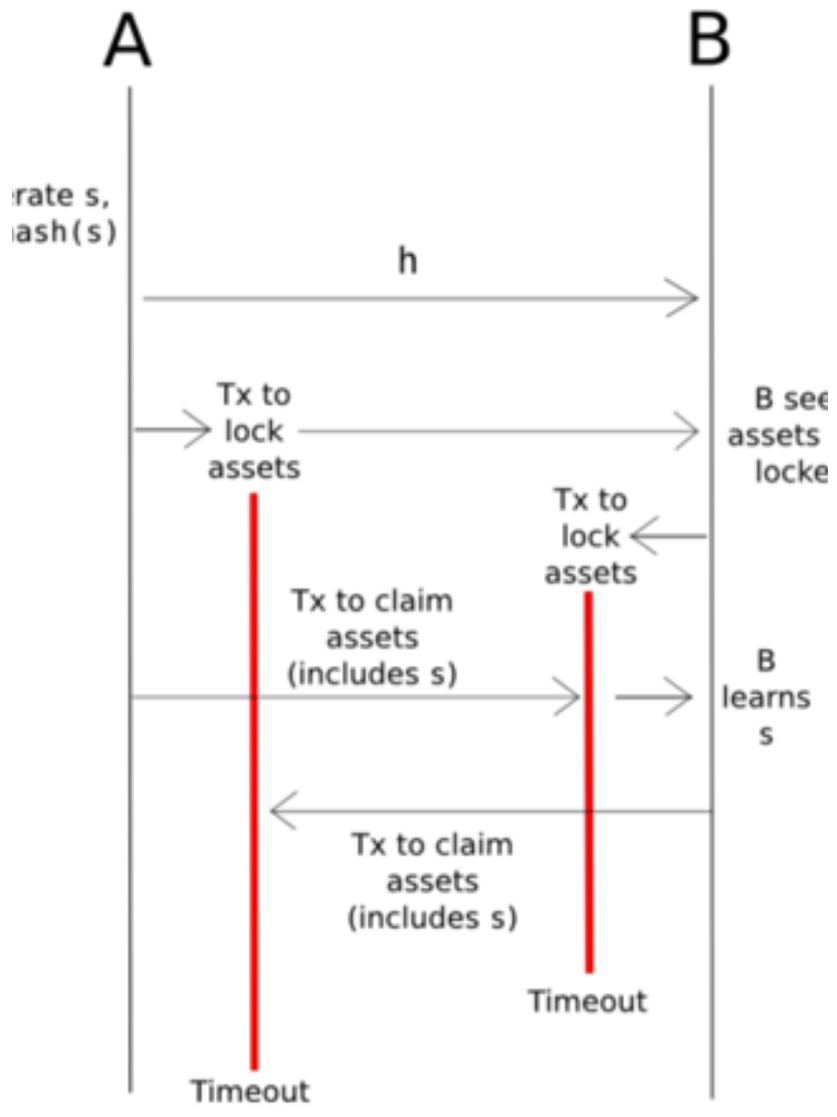
3. TimeLock (CLTV)

4. Signature Check

EXAMPLE

- Suppose Alice wants to exchange some Bitcoin to Litecoin with Bob in an “decentralized” way
- If both tokens (i.e., Bitcoin & Litecoin) support four of the above listed technologies, Atomic swap can be easily implemented by the using the CLTV-CLTV scheme
- No trusted third party is needed

CLTV-CLTV SCHEME



https://en.bitcoin.it/wiki/Atomic_cross-chain_trading

LIMITATIONS

- The main limitation is that CLTV is only supported in a limited set of cryptocurrencies, such as Bitcoin and Litecoin
- When considering a token exchange service, there are more situations where at least one token do not have some time lock mechanism such as CLTV
- Thus , only CLTV-CLTV is not enough!

PART 2

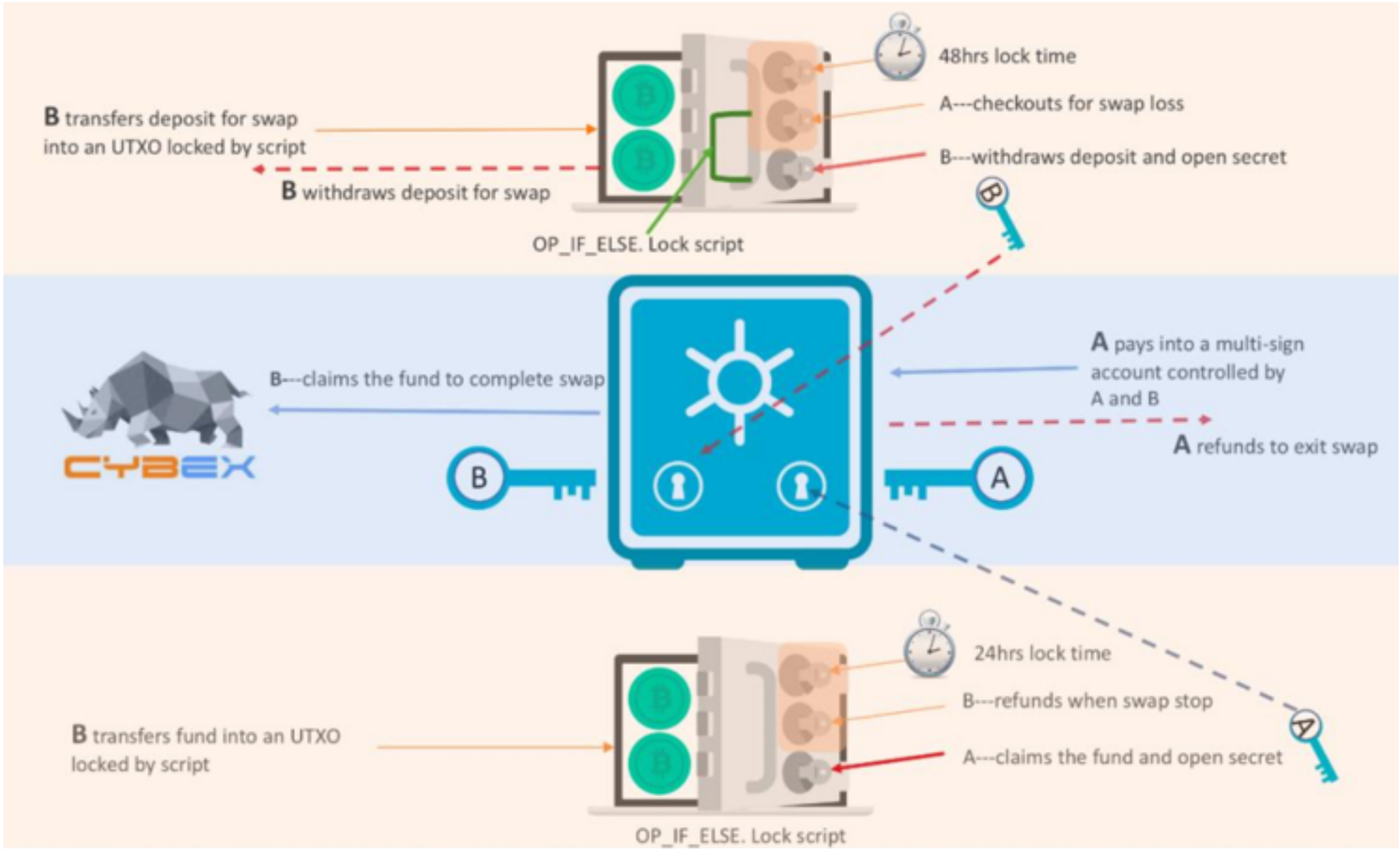
CLTV-MULTISIG

CLTV-MULTISIG

- For cases where only one token has time lock like CLTV, and the other is powerful enough to support multiple signature (abbr. multisig), there exists another protocol for atomic swap
- We call it the CLTV-Multisig protocol
- In fact, CLTV-Multisig can even be seen as a scheme that leads to several different protocols (yet with similar properties)

A MORE POWERFUL SCHEME

CLTV-MULTISIG PROTOCOL

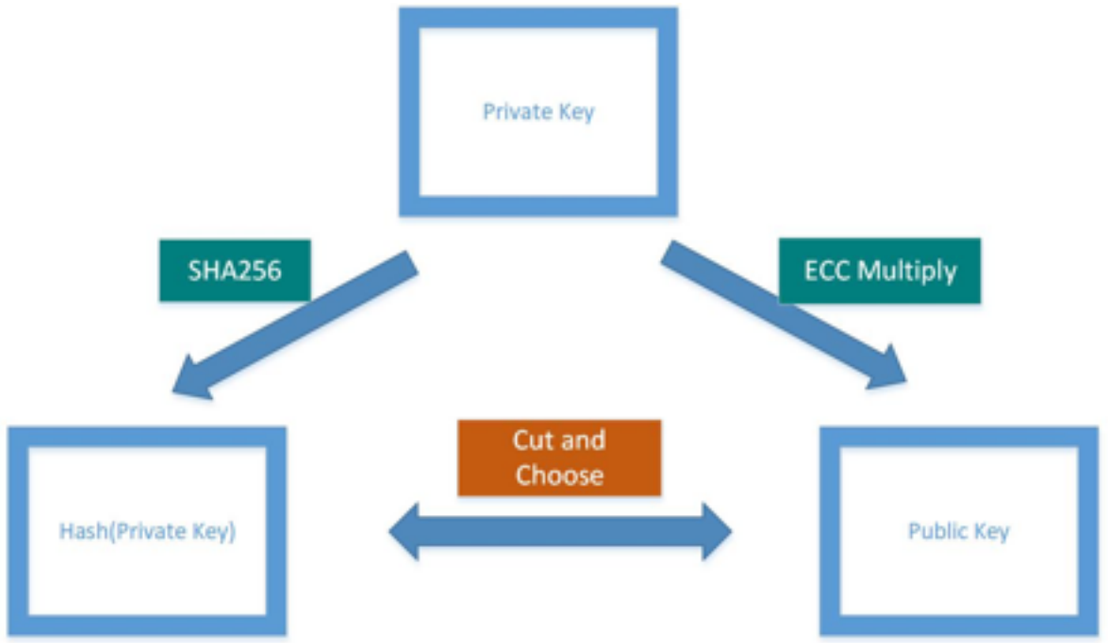


Cybex current implementation uses this one.

PURPOSE OF CUT-AND-CHOOSE

- In the CLTV-MultiSig Paradigm, the purpose of cut-and-choose is to statistically **prove that hash(private key) matches the public key** or in other words **prove that the hash of the private key and the public key are both derived from the same private key**

- The security of the cut-and-choose based atomic swap relies on the above condition



CUT-AND-CHOOSE AS A PROOF

- Suppose there are two parties: A and B
- Both A and B generate 1000 public/private key pairs and corresponding hash value (of public key)
- The set of 1000 <public key, private key hash> pairs are sent to the other, followed by each opposite party randomly choosing one pair, denote pair_m (for A) and pair_n (for B)
- Private keys for pairs except pair_m (for A) and pair_n (for B) are therefore sent to the opposite, proving the validity of the 1000 public/private key pairs each
- In the end, both A and B will have a 99.9% (1-1/1000) belief that the opposite party owns a private key corresponding to the pair they chose and that the private key hash matches the public key.

LIMITATIONS

- One limitation of this cut-and-choose process is higher proof probability will mean **larger data transmission** (say 1000 <public key, private key hash> pair, plus 1 private key for probability 99.9%=1-1/1000)
- Another problem is both parties may also have to **pay an extra transaction fee as a disincentive to them so they are not willing to cheat** (e.g., the chosen public/private key pair has 1/1000=0.1% to pass the random test) in the process
- In some cases, these are not acceptable or might be even overwhelming, especially for very tiny/micro amount exchange

PART 3

BEYOND CUT-AND-CHOOSE

ALTERNATIVES TO CUT-AND-CHOOSE

- Zero-Knowledge Proof
- ECC Multiplication Lock
- ~~Multi-Hash~~
- Combined-Hash-Sig
- and ...

ZERO-KNOWLEDGE PROOF

- By generating a proof, anyone can run a program to check whether the hash of the private key and the public key are both derived from the same private key.
- Hence, by proving that they both derived from the same private key, cut-and-choose is no longer needed.
- https://bitcointalk.org/index.php?topic=1340621.msg13828271#msg13828271

```
boolean check(byte[] bob_priv_key) {
    if (computePublicKey(bob_priv_key) != BOB_PUB_KEY)
        return false;

    if (computeHash160(bob_priv_key) != BOB_PRIV_KEY_HASH)
        return false;

    return true;
}
```

ECC MULTIPLICATION LOCK

- Basic Idea:
  Remove hash(private key) , use ‘public key’ only.
- Replaced:
  OP_HASH160 <hash(bob_priv_n)> OP_EQUALVERIFY
  With:
  <Bob's timelocked public key> CPRKV 2DROP
- The purpose of a hash lock is to let a user expose a secret to another user. This can also be achieved with ECC multiplication lock. Instead of checking whether the hash of the input matches the specified hash, ECC Multiply is used instead to check whether the input private key and the specified public key is an ECC Key Pair.
- https://github.com/TierNolan/bips/blob/cpkv/bip-cprkv.mediawiki

MULTI-HASH

- Basic Idea: Remove 'public key' , use hash(private key) only.
  Replaced:
  OP_2 <bob_pub_n> <alice_pub_m> OP_2 OP_CHECKMULTISIG
  With:
  OP_2 <hash(bob_pub_n)> <hash(alice_pub_m)> OP_2
  OP_CHECKMULTIHASH
- OP_CHECKMULTIHASH: check if the hashes of inputs matches the specified hashes.
- Known **problem** with Multi-Hash: Race Condition
  - Anyone can create a transaction using bob_priv_n and alice_priv_m when both are released.

COMBINED-HASH-SIG

- Basic Idea: Break the relationship between hash(private key) and public key.
- Replaced
  OP_2 <bob_pub_n> <alice_pub_m> OP_2 OP_CHECKMULTISIG
  With:

```
OP_IF
  OP_HASH160 <hash(bob_secret)> OP_EQUALVERIFY <alice_pub_1001> OP_CHECKSIG
OP_ELSE
  OP_HASH160 <hash(alice_secret)> OP_EQUALVERIFY <bob_pub_1001> OP_CHECKSIG
OP_ENDIF
```

- https://github.com/lianran/LonghashHackathon/blob/master/combined-hash-sig.md

COMPARISON OF DIFFERENT ATOMIC SWAP PROTOCOLS

SCHEME	CASE	Blockchain 1				Blockchain 2				Advantages/Disadvantages
		Branch	HashLock	TimeLock	Additional Features	Branch	Hash Lock	TimeLock	Additional Features	
CLTV-CLTV SCHEME	CASE 1: CLTV <-> CLTV e.g. BTC <->LTC	Y	Y	Y	-	Y	Y	Y	-	Best Case;Not all support CLTV
CLTV-MULTISIG SCHEME	CASE 2: CUT-AND-CHOOSE	Y	Y	Y	-	-	-	-	MultiSig	Deposit needed; Not efficient; Fees must be paid to disincentive cheating; Blockchain 2 only need to support Multi Sig;
	CASE 3: ECC Multiply Lock (e.g. CPRKV)	Y	-	Y	CHECKPRIVATEKEYVERIFY (CPRKV)	-	-	-	MultiSig	Deposit needed; ECC Multiply is not widely supported; Blockchain 2 only need to support Multi Sig;
	CASE 4: Mix-Hash-Sig	Y	Y	Y	-	Y	Y	-	-	Deposit needed; Most bitcoin forks support this; only work for coins which allow non-standard scripts (branch and hashlock);

PART 4

Other Proposal

ECONOMIC-INSPIRED PROTOCOL WITHOUT CLTV

- Basic Idea:
  1. If Bob wants to cheat, he put a large sum of his money at stake.
  2. Bob has an incentive to claim his deposit because otherwise his large fund of coins will be locked (especially on exchange sites, you wouldn't want a lot of your money to be locked).

- https://github.com/lianran/LonghashHackathon/blob/master/econ-rm-cltv.md

FROM MULTI-SIG TO 1-1 SIGNATURE

- Some atomic swap models require one of the blockchain to support multi-sig. However, not all blockchain has multi-sig support, such as Nano/RaiBlocks. Here we propose a way of removing the need for multi-sig support by using Elliptic Curve Cryptography's (ECC) Composite Property.
- ECC Composite Property:
  Assume G is the generator point of an ECC group with order n.
  For any private key a,b and its public key aG,bG, the addition of a and b modular n times G equals to the ECC addition of its public key.
- Replaced:
  OP_2 <bob_pub_n> <alice_pub_m> OP_2 OP_CHECKMULTISIG
  With:
  <bob_pub_n + alice_pub_m> OP_CHECKSIG
- https://github.com/lianran/LonghashHackathon/blob/master/multi-sig-optimization-scheme.md

PART 5

IMPLEMENTATION

UNDERSTANDING CYBEX

- Strength:
  1. Powerful MultiSig Feature
  2. Unlike bitshares, only vested balance can be claimed. (vp.vesting_cliff_seconds = ext1.vesting_period)

- Weaknesses:
  1. Lack of scripting language
  2. No TimeLock

OUR GOAL

- Enable CYB to be able to do atomic swap with any other cryptocurrency
- When we trade CYB with cryptocurrency that support CLTV/ Timelocks, we wanted to use the CLTV-CLTV protocol (the one BTC and Litecoin use) , since this protocol does not need its users to deposit money.
- When we trade CYB with cryptocurrency that does not support CLTV/Timelocks, we do not want to use cut-and-choose since users have to pay extra fee and there is a possibility of cheating.

IMPLEMENTATION OPTIONS

- To do an atomic swap with a blockchain that supports timelock(CLTV), CYBEX has the following options:
  1. Keep on using Cut-and-choose
  2. Implement a protocol to support Zero Knowledge Proof.
  3.Implement Combined-hash-sig
  4. Use our economic-inspired protocol
  5. Implement Timelock
- To do an atomic swap with a blockchain that does not support timelock(CLTV), CYBEX has the following options:
  1. Implement Timelock with cut-and-choose / combined-hash-sig / ECC Multiply
  2. Implement ‘Blockchain 1’ part of Economic-Inspired Atomic Swap (branch, hash+cut-n-choose / ECC Private Public Key Pair Check)
- Our ‘Multi-Sig to 1-1 Single Signature’ Scheme will ALLOW Cybex to do atomic swap with ECDSA-based blockchains that could not do atomic swap due to its lack of the multi-sig feature (e.g. Nano/ Raiblocks).

OUR COMBINATION CHOICE

- We decided to implement TimeLock on Cybex, since it will allow us to do atomic swap securely with cryptocurrencies that do no support TimeLock/CLTV.

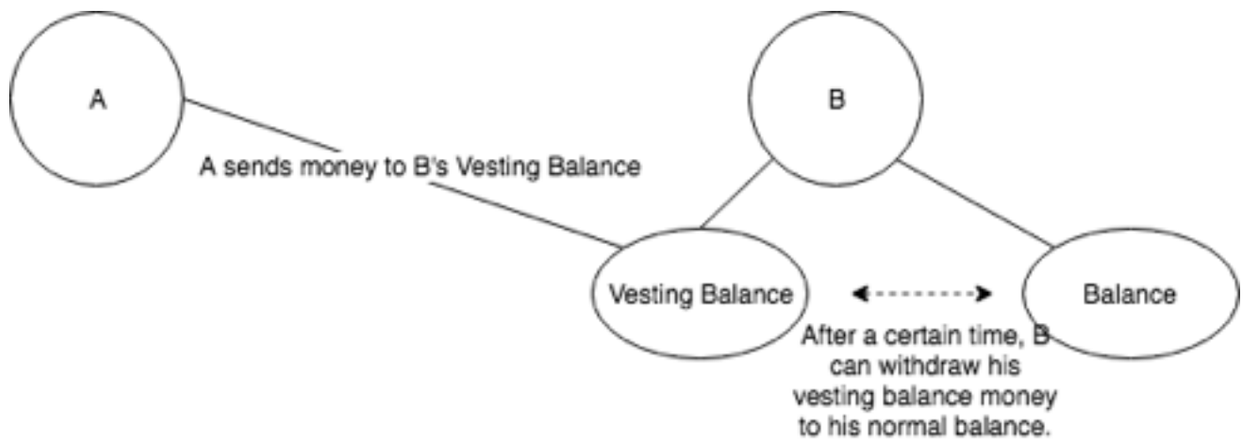
- For CLTV(CYB) <-> CLTV case
  We need HashLock

- For CLTLV(CYB) <-> Non CLTLV case
  We need ECC Multiply (Best Case because the other cryptocurrency only need to support multi-sig unlike combined-hash-sig)

- Our proposed ‘From Multi-Sig To 1-1 Signature' Scheme will allow CYB to do atomic swap with any ECDSA-based cryptocurrency.

WHAT CYBEX HAVE

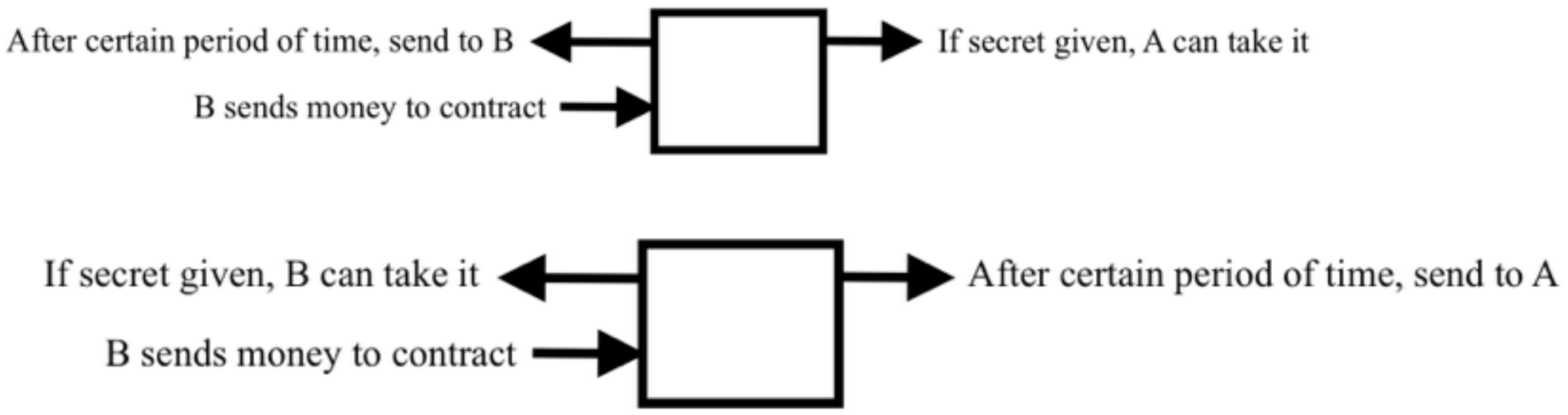
- Vesting balance
  Has a TimeLock-like mechanism



- Authority a flat hierarchy similar to multi-signature, a kind of branch

CLTV/TIMELOCKS + BRANCH

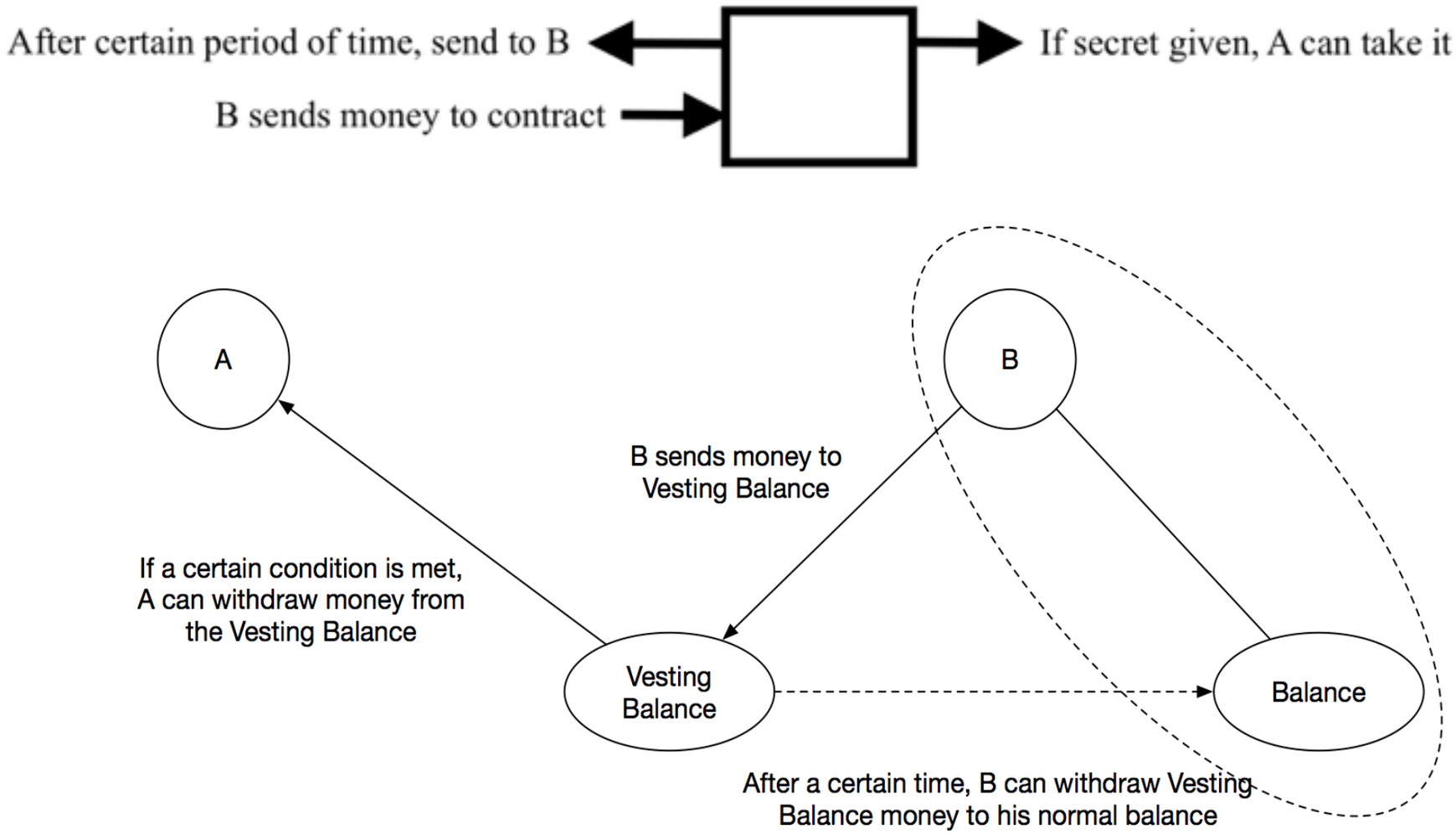
- In Atomic Swap, CLTV is used in two different ways.



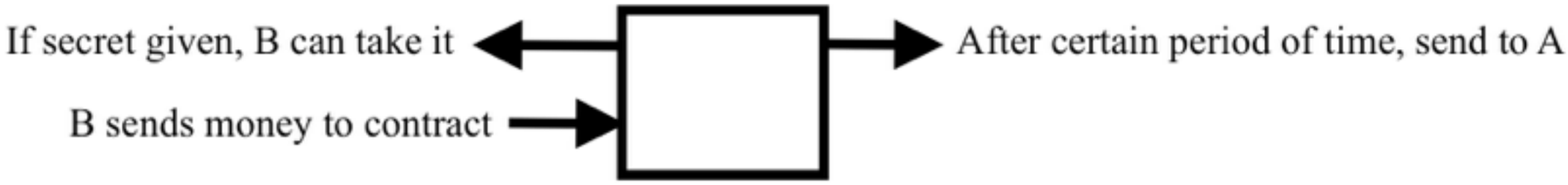
WHAT WE DID

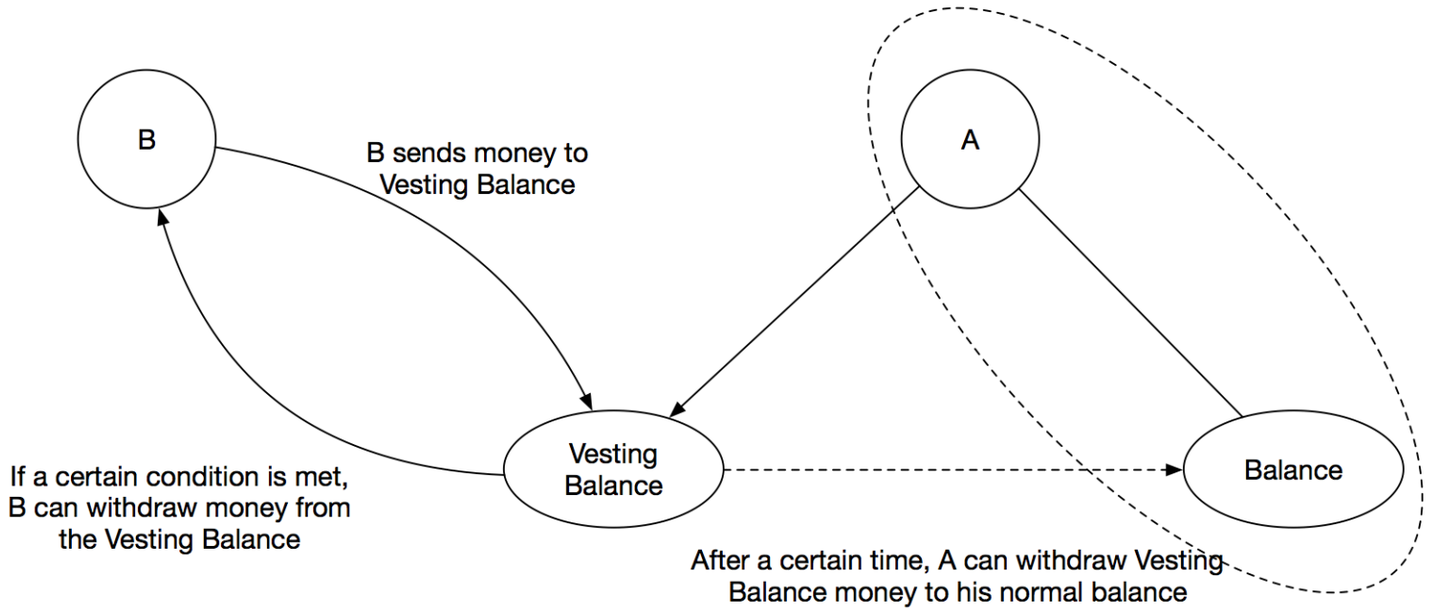
- Implemented TimeLock using Vesting Balance
- Added a conditional cashback feature to Vesting Balance (branching)

VESTING BALANCE BASED TIMELOCK (1)



VESTING BALANCE BASED TIMELOCK (2)





VESTING BALANCE BASED TIMELOCK

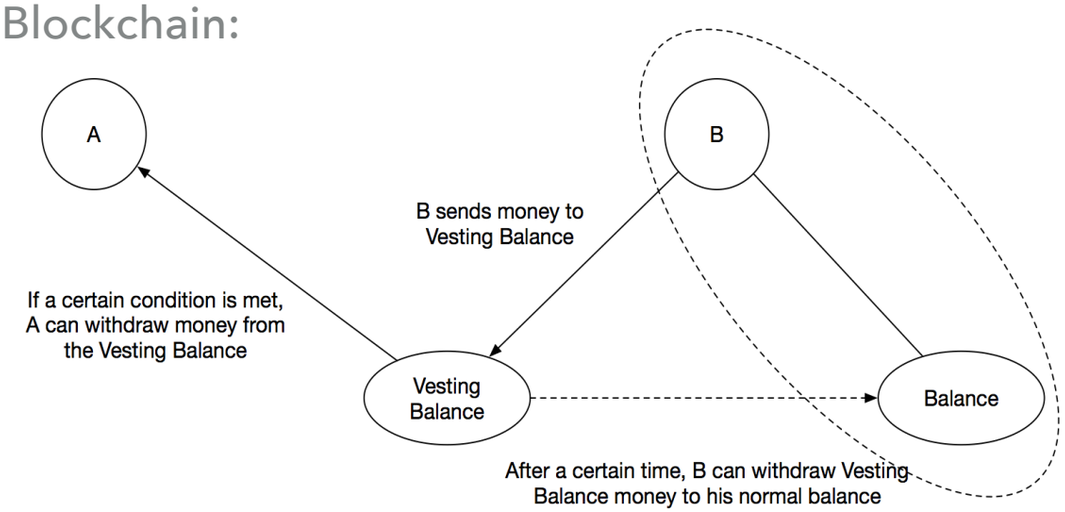
- Our Vesting Balance has Two Owners:
  Owner , CashbackOwner
- Owner will be able to claim after the vesting period.
- During the vesting period, CashbackOwner can claim if provided the correct secret/private key.
- By setting Owner and CashbackOwner, we can implement the above two situations.

IMPLEMENTATION

CYB (CYBEX) <-> CRYPTOCURRENCY X WITH CTLV

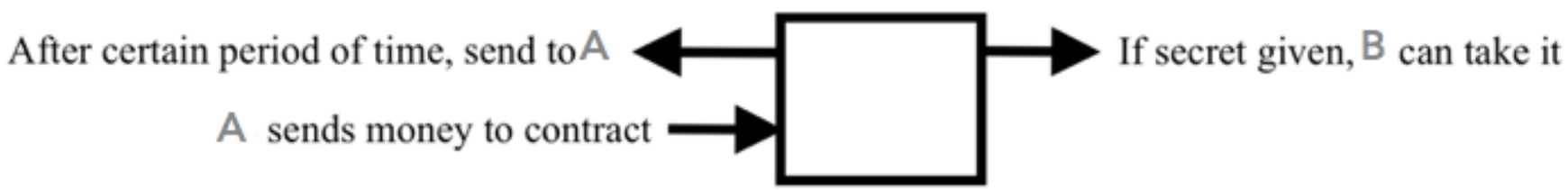
B wants to exchange CYB to X. A wants to exchange X to CYB.

Tx on Cybex's Blockchain:



B is responsible for creating the secret and sending its hash to the A

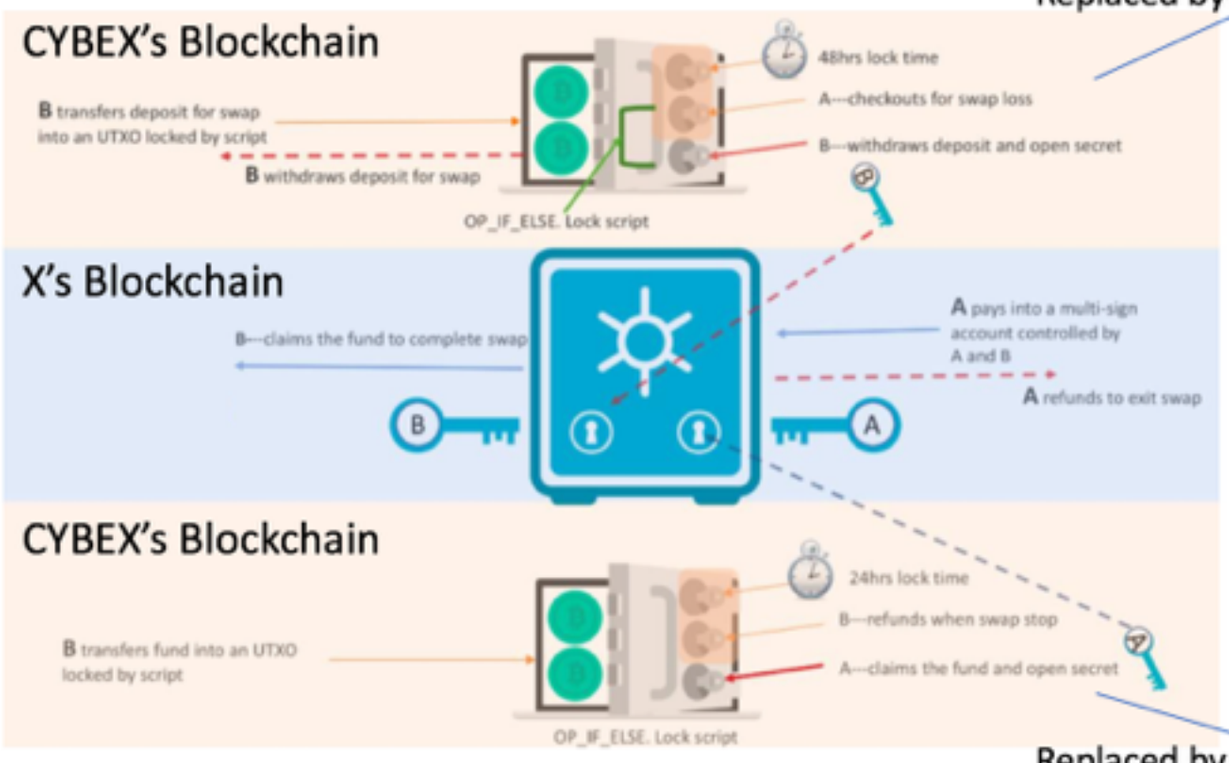
Tx on X's blockchain:



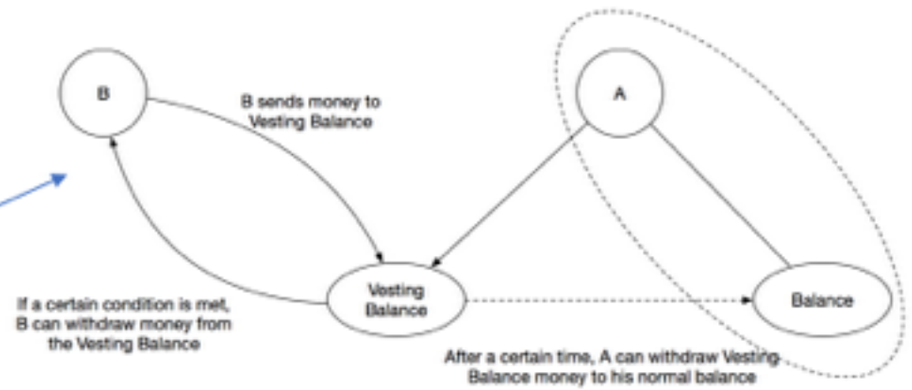
IMPLEMENTATION

CYB (CYBEX) <-> CRYPTOCURRENCY X WITHOUT CTLV

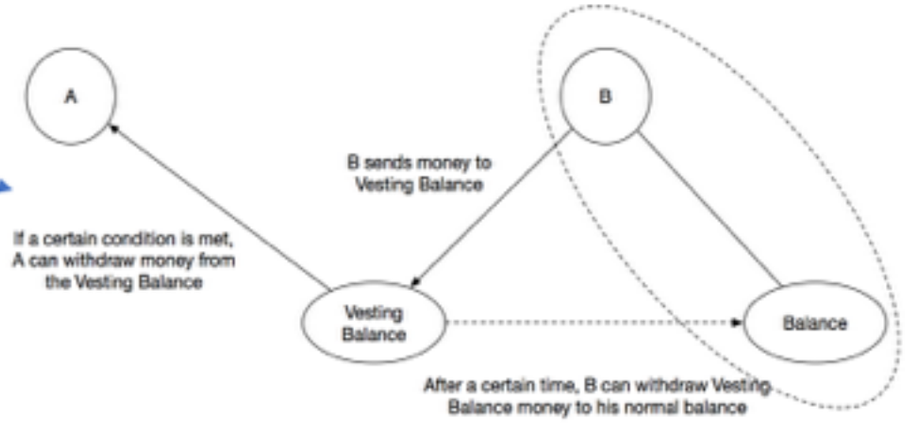
B wants to exchange CYB to X. A wants to exchange X to CYB.



Replaced by



Replaced by



CODE

- https://github.com/lianran/cybex-core

THE END