Extending Atomic Chain Swaps

Jean-Yves Zié, Jean-Christophe Deneuville, Jérémy Briffaut, Benjamin Nguyen

Extending Atomic Cross-Chain Swaps

Jean-Yves Zie[1,2](✉), Jean-Christophe Deneuville[1,3], Jérémy Briffaut[1], and Benjamin Nguyen[1]

[1] Systems and Data Security (SDS) Team, INSA Centre Val de Loire, Blois, France
[2] Orange Labs, Paris, France
jeanyves.ziediali@orange.com
[3] École Nationale de l'Aviation Civile, Toulouse, France

AQ1
AQ2

Abstract. Cryptocurrencies enable users to send and receive value in a trust-less manner. Unfortunately trading the associated assets usually happens on centralized exchange which becomes a trusted third party. This defeats the purpose of a trust-less system. Atomic cross-chain swaps solve this problem for exchanges of value without intermediaries. But they require both blockchains to support Hash locked and Time locked transactions. We propose an extension to atomic swap for blockchains that only support multi-signatures (multisig) transactions. This provides greater capabilities for cross-chain communications without adding any extra trust hypothesis.

1 Introduction

Blockchains enable users to send and receive money in a trust-less manner. They offer a mean for peer-to-peer (P2P) exchanges of crypto-assets such as cryptocurrencies over the Internet, which was not possible without intermediaries before.

Blockchains are a very young ecosystem, where good practices, controls, certifications and regulations still need to be defined. For instance trading the associated assets (located on different blockchains) usually takes place on centralized market places, called exchanges, which become trusted third parties. This expose users to loss of their funds, through hacks of those platforms or straight-up scams when owners disappear with the funds. It emphasizes a common wisdom in this ecosystem: *Not your keys, Not your funds*. If we do not control the private key of the account with the assets, we do not actually own the funds.

Atomic cross-chain swaps solve this problem for exchanges of assets without intermediaries. Those swaps are atomic in the sense that either both parties receive the other crypto-assets, or they both keep their own. They are used in two ways. First, as explained earlier, they can serve as a basis for non custodial exchanges. Using these decentralized marketplaces, users keep control of their funds while being able to trade them.

Second, they enable Layer 2 scaling solutions, like [8,20], to become a network of payment channels and not just P2P channels, thus increasing the scalability of blockchains. The principle of layered scalability for blockchains is to conduct some transactions off the blockchain (*off-chain* or Layer 2) and notarize the state on the Layer 1 (*on-chain*), the blockchain, when necessary. The payment channel works like a tab between two users, with many transactions happening *off-chain* and only two transactions happening *on-chain*, one to open the channel and another one to close it. The limitation is obviously that Alice has to open a channel first with Bob before they can exchange. Atomic swaps come into play for cross-channels payments. If Alice has a payment channel with Bob and Bob has one with Charlie, Alice can atomically send a payment to Charlie with a swap between the two channels, provided there are enough funds in both channels.

The first proposal for atomic cross-chain swaps came from an online forum, *bitcointalk* [22]. It was between Bitcoin [18] and forks of Bitcoin, called altcoins for *alternative coins*. It made use of the scripting capabilities of those protocols, particularly conditional release of the coins, hashed locked and time locked transactions. This restricts the use cases to blockchains with scripting or smart contract capabilities. There are various types of cryptocurrencies with different goals and features and not all of them support Hash Time Lock Contracts (HTLC for short). In this work, we propose an extension of atomic cross-chain swaps to blockchains that do not have such capabilities with the same assumptions as HTLC atomic swaps. We construct our protocol for a blockchain with smart contracts and one that only supports multi-signatures (*multisig*) transactions. Those transactions are control by multiple private keys and require, to be valid, a certain number of signatures. Each signature is generated by a different private key and all those signatures need to be explicitly attached to the transaction. Using multisig transactions, we provide greater capabilities for cross-chain communications without adding any extra trust hypothesis.

The rest of this paper is organized as follows: the properties of atomic cross-chain swaps and their construction using HTLCs are recalled in Sect. 2. We show how to extend them to blockchains with multisig in Sect. 3, then discuss the relative advantages and drawbacks of our proposal in Sect. 4. A comparison to related works is proposed in Sect. 5 before concluding.

2 Hash Time Locked Contracts

2.1 Properties

Atomic swaps have two properties:

1. If all parties behave, i.e follow the protocol, all swaps happen.
2. If any party misbehaves, everyone is refunded.

This gives the users the guarantee they will not lose their assets by participating in an exchange. As stated earlier, the first proposal for blockchains came from *bitcointalk* [22]. We note that most blockchains rely on digital signatures to track the ownership of the coins. Thus, the proposed protocols, so far, make use of some scripting or smart contracts capabilities.

– Hash lock: to release the funds in the contract, one needs to provide the secret preimage s that gives $H(s)$, the lock in the contract,

– Time lock: if nothing happens before some time t on a chain, refund the user on that chain.

Those swaps are currently possible between Bitcoin and altcoins such as Litecoin [17] and on smart contract platforms like Ethereum [5] or EOS [15].

2.2 HTLC Based Atomic Swaps

We now present the atomic swap based on HTLCs. Let Alice be a user of the blockchain $\mathcal{A}$ that wants to exchange X coins with Bob a user of $\mathcal{B}$ for Y coins. We suppose that there is a common cryptographic hash function $Hash$ available on both blockchains. We also suppose that Alice and Bob agree on the timelocks T for Bob and $2T$ for Alice that are function of the blockchains involved. Specifically those timelocks are function of the Block times and confirmation times. We omit the digital signatures in the description for clarity and brevity.

Setup

1. Alice generates a secret preimage s and computes $h = Hash(s)$.
2. She also generates a HTLC on $\mathcal{A}$ with the hash lock h and the time lock $2T$.
3. Alice sends h and the HTLC address to Bob.
4. Bob can verify that the HTLC on $\mathcal{A}$ is properly constructed.
5. If it is valid, then Bob can generate a HTLC on $\mathcal{B}$ with the same hash lock h but the time lock T.
6. Alice can verify that the HTLC on $\mathcal{B}$ is properly constructed.
7. If it is valid, then Alice funds the HTLC on $\mathcal{A}$ and Bob does the same on $\mathcal{B}$.

We stress that the time locks are different on $\mathcal{A}$ and B and that Bob does not know s at this point.

Swap. The swap works as follows:

1. Alice presents the secret preimage s to the HTLC in $\mathcal{B}$ and receives the Y coins. She does so sufficiently early, with respect to T and the block time and confirmation time of $\mathcal{B}$.
2. Bob, who was monitoring the HTLC on $\mathcal{B}$ learns s.
3. Bob can then present s to the HTLC on $\mathcal{A}$ and receive the X coins. He does so sufficiently early, with respect to $2T$ and the block time and confirmation time of $\mathcal{A}$.

Refunds. Each user can refund herself by waiting for her time lock to expire and call the relevant HTLC. Obviously this is only possible if the funds have not been spent, in which case the other user is able to spend hers also.

Discussions. Alice has the advantage of being able to initiate the swap. This means that she can wait as long as possible to see if the trade gets more or less interesting with prices fluctuations. She can also propose a swap just to lock Bob's funds. We do not address these issues in this paper.

3 Extending Atomic Cross-Chain Swaps

The previous protocol is designed for blockchains that support Hashed Timelock Contract. But scripting or smart contract capabilities are not supported by all blockchains, including commonly used ones such as Steem [16]. We propose a protocol to conduct an atomic swap between a blockchain that supports scripting or smart contracts and one that supports multisig instead of HTLCs. Figure 1 presents a high level overview of the actors of the protocol. The basic idea is to transfer all the time locking part on the first blockchain, since the second does not support it. For example Steem supports multisig accounts and uses ECDSA signatures [13]. On Ethereum, the smart contract [19] can be used as an implementation basis. We leave the implementation of the full protocol for future work.

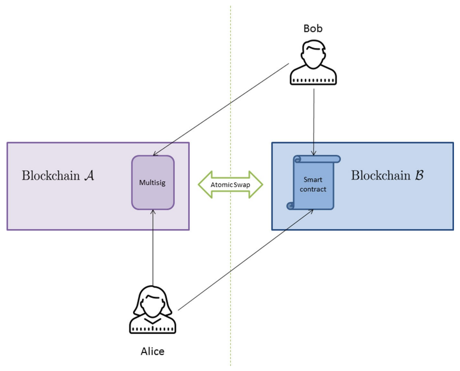

Fig. 1. Atomic Cross-Chain Swaps using multisig and smart contract

3.1 Assumptions

Our protocol enables two users, Alice and Bob, of different blockchains to securely swap their coins. As such we rely on the security model of those blockchains. We assume that each of them has a majority (>2/3) of consensus participants, miners for Proof-of-Work and validators for Proof-of-Stake, that are honest [4,6,9,10,14,18].

We also assume that Alice and Bob have a secure communication channel to do price discovery and exchange the swap details. We suppose that they agree on the timelocks T for Bob and $2T$ for Alice that are function of the blockchains involved (see Discussion Sect. 4).

We further assume that one blockchain supports multi-signature(*multisig*) transactions and that the second blockchain supports smart contracts and can verify the signature of a transaction issued in the first blockchain.

3.2 Notations

Let $\mathcal{A}$ be the blockchain with only multisig and $\mathcal{B}$ be the smart chain. Let skA be the private key for the signature scheme on a blockchain $\mathcal{A}$ and pkA the corresponding public key. We note $\mathsf{sk}\mathcal{A}_A$ and $\mathsf{pk}\mathcal{A}_A$ if these are Alice's private and public keys and $\mathsf{sk}\mathcal{A}_B$ and $\mathsf{pk}\mathcal{A}_B$ for Bob's. We call SC the smart contract for the atomic swap on $\mathcal{B}$. We call M the multisig account on $\mathcal{A}$ that helps make the atomic swap. This can be understood as a shared account between Alice and Bob and we note $\mathrm{M} = (\mathsf{pk}\mathcal{A}_A, \mathsf{pk}\mathcal{A}_B)$ to express that this account requires both signatures, meaning Alice and Bob need to cooperate to transact on the *behalf* of M.

Let the refund operations be $\mathrm{R}(\mathcal{A}, \mathsf{pk}\mathcal{A}_A)$ and $\mathrm{R}(\mathcal{B}, \mathsf{pk}\mathcal{B}_B)$, which means that Alice (resp. Bob) makes a transaction so that the public key $\mathsf{pk}\mathcal{A}_A$ (resp. $\mathsf{pk}\mathcal{B}_B$) has sole control of the funds on the chain $\mathcal{A}$ (resp. $\mathcal{B}$) and Alice (resp. Bob) is thus refunded. Let the swap operations be $\mathrm{S}(\mathcal{B}, \mathsf{pk}\mathcal{B}_A)$ and $\mathrm{S}(\mathcal{A}, \mathsf{pk}\mathcal{A}_B)$, which means that Alice (resp. Bob) makes a transaction so that the public key $\mathsf{pk}\mathcal{B}_A$ (resp. $\mathsf{pk}\mathcal{A}_B$) has sole control of the funds on the chain $\mathcal{B}$ (resp. $\mathcal{A}$) and Alice (resp. Bob) has completed the swap.

3.3 Setup, Transactions and Operations

Setup

1. Alice generates $\mathsf{sk}\mathcal{A}_A$, $\mathsf{pk}\mathcal{A}_A$, $\mathsf{sk}\mathcal{B}_A$, $\mathsf{pk}\mathcal{B}_A$ and sends $\mathsf{pk}\mathcal{A}_A$ and $\mathsf{pk}\mathcal{B}_A$ to Bob.
2. Bob generates $\mathsf{sk}\mathcal{A}_B$, $\mathsf{pk}\mathcal{A}_B$, $\mathsf{sk}\mathcal{B}_B$, $\mathsf{pk}\mathcal{B}_B$ and sends $\mathsf{pk}\mathcal{A}_B$ and $\mathsf{pk}\mathcal{B}_B$ to Alice.
3. Alice creates $\mathrm{M} = (\mathsf{pk}\mathcal{A}_A, \mathsf{pk}\mathcal{A}_B)$ on $\mathcal{A}$ and sends its address to Bob for verification. She also creates and sends $tx_1 = \mathrm{R}(\mathcal{A}, \mathsf{pk}\mathcal{A}_A)$ and $tx_2 = \mathrm{S}(\mathcal{A}, \mathsf{pk}\mathcal{A}_B)$.
4. After verification, Bob computes (σ_1, tx_1), where $\sigma_1 = Sign_{\mathsf{sk}\mathcal{A}_B}(tx_1)$. He can then create SC, fund this contract and send its address and (σ_1, tx_1).
5. Alice verifies that everything is in order and fund the account M.

The Smart Contract SC. It consists of four procedures SC1, SC2, SC3 and SC4 respectively presented in Algorithm 1.

Algorithm 1. The smart contract SC.

```
1:  $unlock \leftarrow \mathsf{false}$
2:  procedure SC1
3:      if $t \geq T$ and $unlock = \mathsf{false}$ then
4:          $\mathrm{R}(\mathcal{B}, \mathsf{pk}\mathcal{B}_B)$
5:  procedure SC2($\sigma_1', tx_1$)
6:      if $\mathsf{Verify}_{\mathsf{pk}\mathcal{A}_A}(\sigma_1', tx_1)$ then
7:          $\mathrm{R}(\mathcal{B}, \mathsf{pk}\mathcal{B}_B)$
8:  procedure SC3($\sigma_2', tx_2$)
9:      if $\mathsf{Verify}_{\mathsf{pk}\mathcal{A}_A}(\sigma_2', tx_2)$ then
10:         $unlock \leftarrow \mathsf{true}$
11: procedure SC4
12:     if $t \geq 2T$ and $unlock = \mathsf{true}$ then
13:         $\mathrm{S}(\mathcal{B}, \mathsf{pk}\mathcal{B}_A)$
```

Operations. The list of possible operations is summarized in Table 1. We use OPX if the operation X succeeds and will mention explicit failure if it does not.

Table 1. List of operations.

OP1	Alice creates $\sigma_1' = Sign_{\mathsf{sk}\mathcal{A}_A}(tx_1)$ and broadcasts $(\sigma_1', \sigma_1, tx_1)$
OP2	Bob waits $t \geq T$ and calls SC1
OP3	Bob creates $\sigma_2 = Sign_{\mathsf{sk}\mathcal{A}_B}(tx_2)$ and sends (σ_2, tx_2) to Alice
OP4	Alice creates $\sigma_2' = Sign_{\mathsf{sk}\mathcal{A}_A}(tx_2)$ and calls SC3
OP5	Bob learns σ_2' monitoring SC and broadcasts $(\sigma_2', \sigma_2, tx_2)$
OP6	Alice waits for $t \geq 2T$ and calls SC4
OP7	Bob learns σ_1' from $\mathcal{A}$ and calls SC2

3.4 Protocol Flow and Guarantees

The protocol flow is described in Fig. 2. It starts with $OP3$, an off-chain communication (black arrow). Alice then calls SC3 (blue arrow) while Bob learns σ_2' (dashed blue arrow). The following action (*) can happen before or after $t = T$ and Bob then controls the X coins (dashed purple arrow). The final steps is $OP6$, where Alice waits for $t \geq 2T$ and calls SC4 (blue arrow) to receive the Y coins (dashed blue arrow).

The protocol needs to ensure that for every outcome, swap or refund, for one user, there exists a sequence of operations the second user can do or could have done to have the same outcome, complete the swap or get a refund. Additionally, each user should get refunded if the other user aborts the swap or does nothing.

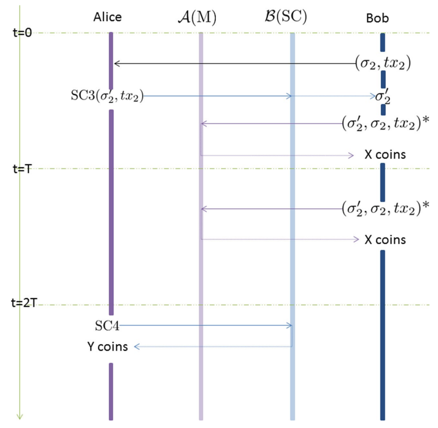

Fig. 2. Protocol flow resulting in a swap (Color figure online)

Swaps. Assuming $\mathrm{S}(\mathcal{A}, \mathsf{pk}\mathcal{A}_B)$, i.e Bob received the X coins, we have:

$$\mathrm{S}(\mathcal{A}, \mathsf{pk}\mathcal{A}_B) \implies OP5 \tag{1}$$
$$OP5 \implies OP4 \tag{2}$$
$$OP4 \implies unlock == True \tag{3}$$
$$\text{if } t \geq 2T \text{ and } OP6 \implies \mathrm{S}(\mathcal{B}, \mathsf{pk}\mathcal{B}_A) \tag{4}$$

The step (3) relies on the fact that Bob can only learn σ_2' if Alice creates it, assuming $\mathcal{A}$ uses a secure signature scheme. The only way for Bob to access the Y coins is to use SC2 since $unlock == True$. He thus needs σ_1', which Alice has no reason to provide.

Assuming $\mathrm{S}(\mathcal{B}, \mathsf{pk}\mathcal{B}_A)$, we have:

$$\mathrm{S}(\mathcal{B}, \mathsf{pk}\mathcal{B}_A) \implies unlock == True \text{ from SC4} \tag{5}$$
$$unlock == True \implies OP4 \tag{6}$$
$$OP4 \implies OP5 \tag{7}$$
$$OP5 \implies \mathrm{S}(\mathcal{A}, \mathsf{pk}\mathcal{A}_B) \tag{8}$$

Thus Bob can complete the swap. There is however one setting in which his part of the swap can fail: Alice has already taken the funds out of M, which is described in Fig. 3. This is covered by the refund case.

Refunds. If Alice does no operation, Bob has to wait for $t \geq T$ to be refunded using $OP2$.

Alice, on the other end, can be refunded any time by using $OP1$. The problem is that Alice can refund herself and then try to get the funds on $\mathcal{B}$ using $OP2$ after $OP1$ while $t \leq T$, as described in Fig. 3. This means that:

$$\text{Since } unlock == True \implies OP2 \text{ will fail when calling SC1} \tag{9}$$

But Alice does not have yet the funds from $\mathcal{B}$ and $T < t < 2T$.

$$\text{if } T < t < 2T \text{ and } OP7 \implies \mathrm{R}(\mathcal{B}, \mathsf{pk}\mathcal{B}_B) \tag{10}$$

4 Discussion

Cross-chain exchanges inherit the adversarial environment of each blockchain. This introduces multiple points of failure that we need to take into account to make the swaps atomic.

For example, Alice or Bob may have the means to launch an Eclipse attack [11]. One user would then have control over the other's network connections and decide which transactions reach the rest of the network. One user can also just have a better connectivity than the other and be sure his or her transactions would go through first, in the case of race. We can not solve such problems in the protocol without further assumptions thus we rely on the time locks the users set. They represent the intervals of time necessary for the users, in comparison to the block times, the confirmation times, to create, broadcast the relevant transactions and have them confirmed. In practice, they should also consider what is the current states of both blockchains as to not use them when they are more vulnerable. This could be the case if the mining hashing rate in Proof-of-Work, or the price of the coin in Proof-of-Stake crashes dramatically.

Another question is the pre-generation of the transactions that are in the smart contract M. This is relevant for security. It is better to require the signature of a *particular* message under a private key than *any* message under that key. This is also relevant for the fees of the transactions. Some blockchains do not take fees (Steem) or have very low fees (Litecoin) while other have a volatile fee market (Bitcoin, Ethereum). Alice can take care of this on $\mathcal{A}$ while Bob pays on $\mathcal{B}$. Still, if the fees were too low at creation, there is the risk that the transactions would take too long to appear in the blockchain(s).

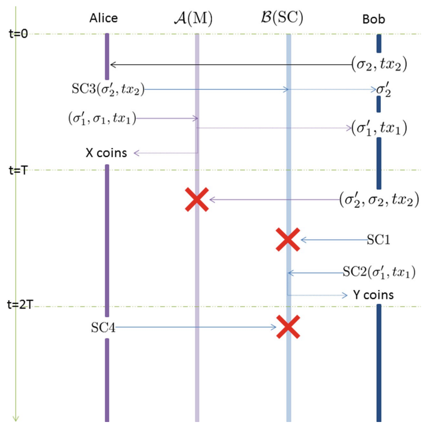

Fig. 3. Alice attack scenario

5 Related Works

The first atomic swap protocol emerged on a Bitcoin online forum and is based on HTLCs [22]. It has since been standardized on Bitcoin development resources [1,2]. This protocol is briefly presented in Sect. 2.

To our knowledge, Herlihy [12] presented the first formal study of the underlying theory of atomic cross-chain swaps. It focuses on a swap of on-chain and off-chain assets between three parties. Using graph theory and the HTLCs, they explore the time locks constraints for the atomicity in that particular setting.

Many protocols focus on cross-chain communications and exchanges. Interledger [21] uses Hashed-Timelock Agreements (HTLAs) [7], which generalized the idea of HTLCs for payments systems with or without a private or public ledger. Those agreements are used to create secure multi-hop payments using conditional transfers. The parties involved decide where their trust lies, for example a HTLC between blockchains or a legal contract.

Cosmos [3,4], Polkadot [23] and earlier versions of Interledger [21] rely on a set of validators to ensure cross-chain communication. Each round, a subset of those validators decides which cross-chain information to notarize on their chain, provided only a small protocol-defined portion of those validators is Byzantine. The problem is to guarantee that those systems are decentralized enough to be consider censorship resistant and secure while remaining scalable.

XClaim [24] is a framework for achieving trustless cross-chain exchanges using cryptocurrency-backed assets based on a smart contract blockchain and an existing link between the other blockchains like a relay. It requires vault providers to guarantee the liquidity and ownership on the original chain and to have collateral on the smart contract blockchain. Bad behaviour is discouraged using slashing or proof-of-punishment on the smart contract chain. With these assumptions, XClaim builds a trustless and non interactive protocol for issuance, redemption and swapping of tokenized assets.

Our construction does not require collateral (because there is no vault to provide) or an existing link between the blockchains. We argue that our protocol uses ~~the~~ simpler assumptions to enable trustless exchange between users, in the same manner as HTLC based atomic cross-chain swaps.

6 Conclusion

In this work, we propose a new protocol for atomic cross-chain swaps. We extend the support of atomic swaps to blockchains without hash lock and time lock capabilities without additional trust requirements. Users can now trade in a P2P way between blockchains with smart contracts and blockchain with only mutli-signatures transactions. We leave the full implementation and live experimentation of this protocol for future work.

References

1. bitcoinwiki: Atomic swap (2019). https://en.bitcoin.it/wiki/Atomic_swap. Accessed 03 July 2019

2. Bowe, S., Hopwood, D.: Hashed time-locked contract transactions, March 2017. https://github.com/bitcoin/bips/blob/master/bip-0199.mediawiki. Accessed 03 July 2019
3. Buchman, E., Kwon, J.: Cosmos: a network of distributed ledgers. https://github.com/cosmos/cosmos/blob/master/WHITEPAPER.md. Accessed 03 July 2019
4. Buchman, E., Kwon, J., Milosevic, Z.: The latest gossip on BFT consensus. CoRR abs/1807.04938 (2018)
5. Buterin, V.: Ethereum: a next-generation smart contract and decentralized application platform. Accessed 03 July 2019 AQ3
6. Daian, P., Pass, R., Shi, E.: Snow white: robustly reconfigurable consensus and applications to provably secure proofs of stake. In: IACR, pp. 1–64 (2017)
7. de Jong, M., Schwartz, E.: Hashed-timelock agreements, Novemebr 2017. https://github.com/interledger/rfcs/blob/master/0022-hashed-timelock-agreements/0022-hashed-timelock-agreements.md. Accessed 03 July 2019
8. Decker, C., Wattenhofer, R.: A fast and scalable payment network with Bitcoin duplex micropayment channels. In: Pelc, A., Schwarzmann, A.A. (eds.) SSS 2015. LNCS, vol. 9212, pp. 3–18. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21741-3_1
9. Eyal, I., Sirer, E.G.: Majority is not enough: Bitcoin mining is vulnerable. Commun. ACM **61**(7), 95–102 (2018)
10. Gervais, A., Karame, G.O., Wüst, K., Glykantzis, V., Ritzdorf, H., Capkun, S.: On the security and performance of proof of work blockchains. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS 2016, pp. 3–16. ACM, New York (2016)
11. Heilman, E., Kendler, A., Zohar, A., Goldberg, S.: Eclipse attacks on Bitcoin's peer-to-peer network. In: 24th USENIX Security Symposium (USENIX Security 2015), pp. 129–144 (2015)
12. Herlihy, M.: Atomic cross-chain swaps. In: Newport, C., Keidar, I., (eds.) Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC 2018, Egham, UK, 23–27 July 2018, pp. 245–254. ACM (2018)
13. Johnson, D., Menezes, A., Vanstone, S.: The elliptic curve digital signature algorithm (ECDSA). Int. J. Inf. Secur. **1**(1), 36–63 (2001)
14. Kiayias, A., Russell, A., David, B., Oliynykov, R.: Ouroboros: a provably secure proof-of-stake blockchain protocol. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 357–388. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_12
15. Larimer, D.: EOS.IO technical white paper V2. Accessed 03 July 2019
16. Larimer, D., Scott, N., Zavgorodnev, V., Johnson, B., Calfee, J., Vandeberg, M.: Steem: an incentivized, blockchain-based social media platform. Self-published, March 2016
17. Lee, C.: Litecoin (2011)
18. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008). http://bitcoin.org/bitcoin.pdf
19. OpenZeppelin: ECDSA solidity library. https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/cryptography/ECDSA.sol. Accessed 03 July 2019
20. Poon, J., Dryja, T.: The Bitcoin lightning network (2016). https://lightning.network/lightning-network-paper.pdf. Accessed 03 July 2019
21. Thomas, S., Schwartz, E.: A protocol for interledger payments (2015). https://interledger.org/interledger.pdf

22. TierNolan: bitcointalk: Alt chains and atomic transfers, May 2013. https://bitcointalk.org/index.php?topic=193281.msg2224949#msg2224949. Accessed 03 July 2019
23. Wood, G.: Polkadot: vision for a heterogeneous multi-chain framework draft 1. https://polkadot.network/PolkaDotPaper.pdf. Accessed 03 July 2019
24. Zamyatin, A., Harz, D., Lind, J., Panayiotou, P., Gervais, A., J. Knottenbelt, W.: XCLAIM: trustless, interoperable, cryptocurrency-backed assets, March 2019