ZeroLender: Trustless Peer-to-Peer Bitcoin Lending Platform

Yi Xie† Computer Science Department Boise State University

Joshua Holmes† Computer Science Department Boise State University

Gaby G. Dagher* Computer Science Department Boise State University

ABSTRACT

Since its inception a decade ago, Bitcoin and its underlying blockchain technology have been garnering interest from a large spectrum of financial institutions. Although it encompasses a currency, a payment method, and a ledger, Bitcoin as it currently stands does not support bitcoins lending. In this paper, we present a platform called ZeroLender for peer-to-peer lending in Bitcoin. Our protocol utilizes zero-knowledge proofs to achieve unlinkability between lenders and borrowers while securing payments in both directions against potential malicious behaviour of the ZeroLender as well as the lenders, and prove by simulation that our protocol is privacy-preserving. Based on our experiments, we show that the runtime and transcript size of our protocol scale linearly with respect to the number of lenders and repayments.

1 INTRODUCTION

In the last decade, cryptocurrencies have emerged as the solution to low-fee decentralized banking with relatively quick settlement time. Prior to the introduction of blockchain, the main problem every payment network was concerned about is how to prevent *double-spending*, i.e., the same money cannot be spent in more than one payment. In the traditional (centralized) banking systems, double-spending is typically prevented using central databases that log transactions and balance the ledgers. Bitcoin [15] is a peer-to-peer network that achieves consensus about its distributed ledger without a central authority. Transactions in Bitcoin are verified by network nodes and recorded on a shared public ledger. Since Bitcoin was introduced, many other blockchain-based cryptocurrencies have been created [22][18][19]. Despite of the prosperity of altcoins, Bitcoin still by far has the largest network adoption, as reflected in its market cap. Currently, Over 100,000 merchants worldwide accept payments in bitcoins, including Microsoft online store, Subway, as well as the online electronics retailer Newegg[1].

*Corresponding Author (✉)

[1]Places that accept Bitcoin: http://spendbitcoins.com/places/

† These authors contributed equally.

CODASPY '20, March 16–18, 2020, New Orleans, LA, USA
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7107-0/20/03…$15.00
https://doi.org/10.1145/3374664.3375735

Peer-to-peer (P2P) economy, also referred to as *sharing economy*, is a decentralized business model whereby individuals deal with goods or services directly with each other without any intermediary. *P2P lending* is an emerging branch of P2P economy that enables direct matching of borrowers and lenders without holding the loan on an intermediary balance sheet. P2P lending platforms generate revenue from origination fees that are charged to borrowers, service fees interest that are charged to lenders, as well as from additional charges such as late fees [7]. The P2P leading model is illustrated in Figure 1.

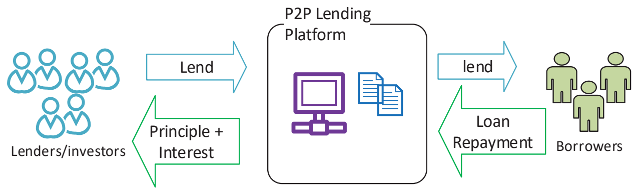


Figure 1: Illustration of the peer-to-peer lending model

Some P2P lending companies [2], such as BitBond [3] and BTCPOP [4], crowdsource loans from lenders around the world while offering low interest rates to borrowers. Although these platforms accept payments from lenders in bitcoin, the lending process is handled off-chain using fiat money, and requires the lenders to fully trust the lending platform to manage their bitcoin assets.

In this paper, we present a trustless peer-to-peer lending platform called ZeroLender. The borrower is assumed to be an established entity (e.g. business) that uses the ZeroLender platform to obtain (borrow) an amount of bitcoins, and later returns the amount with an agreed upon interest. The lenders are Bitcoin users who aim to lend (invest) their bitcoins for a certain period of time to gain interest on the amount they are lending. Our platform ensures that individual lenders can access all lending opportunities and securely transfer bitcoins to the borrowers and receive repayments back from the borrows through ZeroLender. Although this paper focuses on P2P lending, our protocol is not limited to this case, as it could be directly (or with minimal modifications) applied to other related sharing economy models, such as crowdfunding [12] and group buying business [1]. Based on the characteristics of the P2P lending process, ZeroLender consists of three phases[5]. *First*, the

[2]Bitcoin Market Journal: https://www.bitcoinmarketjournal.com/bitcoin-borrowing/
[3]BitBond: https://www.bitbond.com/
[4]BTCPOP: https://btcpop.co/home.php
[5]Since this paper focuses on the privacy of the transactions, some detailed P2P lending processes such as borrower's application acknowledge, credit, approval, and loan management are ignored.

negotiation phase, where the borrower and ZeroLender agree on the required investment amount, loan interest, repayment term, payment addresses, repayment plan, etc. *Second*, the lending phase, where ZeroLender transfers to the borrower the requested loan in return for simultaneously receiving the exact amount of bitcoins from the lenders. *Third*, the returning phase, where for each scheduled repayment, and in order for the borrower to release the repayment bitcoins to ZeroLender, the latter must prove to the former that the appropriate amount of bitcoins was sent to each lender according to the repayment plan.

The proposed ZeroLender platform in this paper satisfies the following properties:

- *No direct connection.* The Borrowers announce their investment opportunities and the lenders invest in projects and receive their repayments through the ZeroLender platform, without any need for interaction between the lenders and the borrowers.
- *Undeniability.* ZeroLender can not , nor can it dictate or change the amount the lender has invested.
- *Fair Return.* Each lender is guaranteed to receive a *proportional* amount from each net repayment made by the borrower through ZeroLender. Individual repayments are not necessarily equal, but instead fall in to an acceptable range to make them harder to track on the blockchain, and ZeroLender cannot decide who is assigned which repayment plan.
- *Unlinkability.*There is no linkage between lenders' accounts and borrower's accounts to the public (external hiding) or to each other (internal hiding).
- *Privacy.* Only ZeroLender has knowledge of the individual loan amount of each lender, and the Bitcoin addresses of the lenders and borrowers .

Note that if ZeroLender is defined as a financial institution, it can satisfy the Know Your Customer (KYC) regulation by identifying and verifying the identities of its clients (lenders and borrowers). Further regulation discussion is beyond the scope of this paper.

Limitations. While our protocol is secure against malicious lenders and ZeroLender, the borrower is assumed to be covert and must be held accountable outside the protocol for any dishonest behavior. The borrower's bitcoin addresses and transactions remain private, but its business information is public to all lenders. In addition, the borrower does not have the flexibility to make partial nor early repayments. For more details, please see Section 6.0.2.

2 BACKGROUND

2.1 Bitcoin

Bitcoin is a digital currency with no physical backing that can be sent electronically from one user to another, anywhere in the world. The minimum denomination of Bitcoin is 0.00000001 bitcoin. The system is run by a decentralized network of nodes around the world that keep track of all Bitcoin transactions. All transactions must to be submitted to the network and be verified by at least half the network to be accepted as a part of a block in the blockchain. Currently, only five standard Bitcoin transactions are accepted by the network: Pay To Public Key Hash (P2PKH), Pay To Script Hash (P2SH), Multisig, Pubkey, and Null Data.

In this paper, ZeroLender protocol only utilizes the three most common types of transactions: P2PKH, P2SH, and Multisig.

2.2 CoinSwap

Greg Maxwell presented a protocol called CoinSwap [6], where Alice can pay Bob through Carol and Carol can not steal the coins from either of them. The protocol requires four published transactions. All published transactions look like regular 2-of-2 escrow transactions (two escrow payments, two escrow releases) as long as each party follows the protocol. CoinSwap has three phases:

(1) *Escrow phase*: Alice and Carol set up an escrow through 2-of-2 multisig account with Alice's coins on the blockchain time-locked to a certain period of time window. Carol and Bob repeat the same process with carol's coins but shorter time window.
(2) *Warranty phase*: Carol sends a transaction from the escrow address between her and Bob locked by a hash value, and without posting it on blockchain. Similarly, Alice sends another transaction from the escrow address between her and Carol locked by the same hash aforementioned value. If Bob gets paid from Carol, Carol will be guaranteed to get payment from Alice.
(3) *Payment phase*: Carol signs and releases the transaction directly, then Alice does same. Both transactions will be posted to the blockchain.

A key concept in this protocol is the *warranty phase*. Transactions are protected by the knowledge of the preimage of a hash. A variation of the CoinSwap protocol integrating zero-knowledge proofs with the protocol is proposed in ZeroLender's Lending phase.

2.3 Pedersen Commitment

In this paper, Pedersen Commitment [16] is used to commit relevant messages such as investment amount, returning amount, and repayment receiving addresses from ZeroLender. The commitments rely on two generators g and h such that no party knows the discrete log a between g and h. The commitments hide x using a random number $r \in \mathbb{Z}_q$, where q is the order of the group used. The commitments are constructed as follows[6]:

$$[\![x]\!] = g^x \cdot h^r$$

These commitments are perfectly hiding because for any pair (x, r) there exists an r' for every $x' \in \mathbb{Z}_q$ that also fulfills this relationship. However, finding another pair (x', r') without knowing a is at least as hard as finding a from g and h, i.e. the Elliptic Curve Discrete Log Problem.

2.4 Zero-Knowledge Proof

A zero-knowledge proof is a method by which one party can prove to another that a given statement is true, without conveying any additional information apart from the fact that the statement is indeed true. In the Negotiation, Lending and Returning phases, zero-knowledge proofs are used to prove each statement without revealing related parties' information. ZeroLender utilizes the Fiat-Shamir Heuristic [5] to make these protocols non-interactive.

[6]We utilize the elliptic curve used in Bitcoin, **secp256k1**, where g is the standard Bitcoin generator. Though we use elliptic curves, we use the more conventional notation $y = g^x$ instead of $Y = x \cdot G$.

2.4.1 Schnorr's Protocol. ZeroLender leverages one of the simplest and frequently used protocols, Schnorr protocol [20] (Protocol 1). It allows a party with a public value $y = g^x$, where x is secret to prove knowledge of x without revealing any information about x. ZeroLender uses Schnorr's protocol to prove that a Pedersen commitment hides a specific value and to prove that a Pedersen commitment hides the same value as another commitment.

Schnorr's Protocol [20]

Input: Public: The exponential $y = g^x$
Private : x, the discrete log of y
(Prover)

Output: Public: Verification that the Prover knows x.

(1) The Prover generates a random number r and calculates $a = g^r$
(2) The Prover sends a to the Verifier.
(3) The Verifier generates a random challenge c and sends it to the Prover.
(4) The Prover computes $z = r + c \cdot x$.
(5) The Prover sends z to the Verifier.
(6) The Verifier accepts if $g^z = y^c \cdot a$.

Schnorr's Protocol's Simulator

(1) The Prover* uses the Verifier's random number generator to generate the Verifier's challenge c.
(2) The Prover* randomly generates the response z.
(3) The Prover* calculates $a = g^z \cdot y^{-c}$.
(4) Run the protocol from protocol step 2, skipping step 4, instead using the generated z.

Protocol 1: Schnorr's Protocol

2.4.2 Logical Composition of Zero Knowledge Proofs. Logical composition allows us to answer more complicated questions. We can logically compose any Zero Knowledge Proof using AND or OR [8] [4].

Zero Knowledge AND. It is a construction where the same challenge is shared between multiple proofs. To accomplish this, the Prover creates the initial communications of each proof, then receives the challenge, then computes the response for each proof. The Verifier accepts if all the proof transcripts are valid.

Zero Knowledge OR. Zero Knowledge OR is a more complicated construction. The Prover generates internal challenges for all but one of the proofs and simulates the initial communications for these proofs. The remaining proof must be the one that the Prover can actually prove, so the Prover calculates the appropriate initial communications. The Prover now receives the challenge and computes the remaining internal challenge for the true proof such that all of the internal challenges XOR [8] [4] to the challenge. They then complete the simulated proofs and the real proof and the Verifier accepts if all the proof transcripts are valid and the internal challenges XOR to the challenge.

Since the Prover can not know all of the internal challenges in the initial communication phase, at least one of the proofs must have been executed without prior knowledge of that challenge. Therefore, at least one of the proofs must be genuine. However, due to the nature of simulators, there is no method with which to differentiate the true proof from the simulated proofs.

3 PROTOCOL OVERVIEW

In this section, we provide an overview of the three main phases of the ZeroLender protocol.

3.1 Negotiation Phase

This is the investment project's initial setup phase. In this phase, the borrower discusses the loan with ZeroLender, and if accepted, a loan information is posted to the bulletin board BB by ZeroLender. In addition, a unit-based raw repayment plan is agreed upon between ZeroLender and the borrower to decide each repayment amount in the returning phase.

3.2 Lending Phase

In this phase, our protocol integrates ZKP with the CoinSwap protocol to detect if ZeroLender over-collect funds, given that there is no linkage between the lenders and the borrower. The lending phase can be split into three main steps:

(1) **Fundraising**. Each intended lender and ZeroLender creates a 2-of-2 multisig escrow account with lender's coin on the blockchain and is timelocked to a certain period of time in case a party has to abort the protocol. When the total fund matches the requested loan amount, ZeroLender escrows requested loan amount to a timelocked 2-of-2 multisig account shared by ZeroLender and the borrower. ZeroLender commits the ID of each potential lenders, project ID, lending amount, repayment acceptance addresses, among other information, to BB.
(2) **Securing the fund**. Lenders and ZeroLender setup a mutually signed redeeming hash-locked transaction from the escrow account. Similarly, ZeroLender and the borrower setup a mutually signed escrow redeeming transactions that share the same hash-lock (not posted on blockchain). With all hash-locked transactions in hand, ZeroLender works with the lenders to generate the final repayment plan.
(3) **Deposition**. ZeroLender directly signs the escrow account shared with the borrower, and then lenders also sign their escrow accounts between them and ZeroLender.

3.3 Returning (Repayment) Phase

In this phase, the borrower starts to repay the loan to the lenders. The protocol consists of three main steps:

(1) **Creating repayment transaction**. ZeroLender created a transaction to each eligible lender to pay her according to her lending amount and the repayment amount.
(2) **Proof of transaction**. ZKP is used in this step for ZeroLender to prove to the borrower that it paid all lenders the proper amount, and the total amount matches the repayment amount minus its fees.
(3) **Finishing repayment transaction**. If the borrower is able to verify that ZeroLender prepaid each lender according to the

protocol, the borrower sends a transaction to ZeroLender to complete one repayment cycle.

4 SECURITY MODEL

Our security model is as follows:

- Lenders and borrowers are protected against malicious ZeroLender.
- The lenders are malicious adversaries. Privacy of ZeroLender and the borrower from the lenders or external parties is preserved.
- The protocol is secure against collusion between ZeroLender and the lenders.
- The borrower is a covert adversary. Privacy of ZeroLender and lenders from the borrower is preserved.

5 SOLUTION: ZeroLender PROTOCOL

Our goal is to allow the lenders to transfer investment coins to the borrower, and the borrower distribute repayments to all lenders, while achieving unlinkability. Without a loss of generality, we assume there are n_A lenders and one borrower. We separate the protocol into *negotiation phase*, *lending phase*, and *returning phase*. We follow the notations summarized in Table 1.

Symbol	Description
$\mathcal{Z}$	ZeroLender
$\mathcal{A}$	Lenders
$\mathcal{B}$	Borrower
BB	Bulletin board
L_{id}	Loan ID
ZID	User's ID
$\mathcal{L}$	Intentional lenders list
$\mathcal{L}'$	lenders waiting list
$\mathcal{T}$	Repayment list
$\mathcal{R}$	Raw repayment table
$\mathcal{M}$	Mapping table
$\hat{\mathcal{R}}$	Consolidated repayment table $\mathcal{R}$
$addr_{(\mathcal{X},\mathcal{Y})}$	Multisig account between $\mathcal{X}$ and $\mathcal{Y}$
$TX_{i(\mathcal{X},\mathcal{Y})}$	Transaction i from $\mathcal{X}$ to $\mathcal{Y}$
$addr$	Bitcoin address
σ	transaction amount
$u_{\mathcal{A}}$	Number of lenders
M	Total amount of loan
n_t	Number of repayments (loan terms)
I	Interest rate
b_j	Borrower's repayment for time period j
u	Unit of loan
m	Number of unit
m_j	Number of unit of lender $\mathcal{A}_j$
n_u	Total row numbers of $\mathcal{R}$, where $n_u = M/u$
$v_{i,j}$	Repayment for time period j at row i
$\hat{v}_{k,j}$	Final repayment for time period j for lender k
$\mathcal{M}_{i,j}$	Plain element value of $\mathcal{M}$
r	Total repayment in whole time period for investment u
g, h	generator
q	The order of the group
$[\![x]\!]$	Committed value x
Δ	Repayment range
N	Anonymity set of transactions

Table 1: Notations

5.1 Negotiation Phase

First, the borrower $\mathcal{B}$ submits its loan application to ZeroLender $\mathcal{Z}$. Both of them agree on the following loan information: total amount of loan M, numbers of repayment (loan term) n_t, and interest rate I. All these loan information along with $\mathcal{B}$'s public entity information are posted to BB for lender's review. ***Second***, $\mathcal{Z}$ creates 2-of-2 multisig escrow account $addr_{(\mathcal{Z},\mathcal{B})}$ with $\mathcal{B}$. ***Third***, based on the interest rate and loan terms, a repayment plan must be set upfront. A typical P2P lending platform has fixed monthly repayment calculated based on the interest rate and loan terms. After lending platform gets a repayment from borrower, the platform distributes the repayment to the lenders based on their lending amount. Using this model, the fixed amount transaction impacts the privacy of all parties since these transactions have very obvious patterns (fixed amount, fixed transaction timeline, etc.) on the blockchain. To overcome this shortcoming, we introduce *variable repayment list* $\mathcal{T}$ of n_t: $[\![b_1, b_2, b_3, \ldots, b_{n_t}]\!]$ agreed upon by both $\mathcal{B}$ and $\mathcal{Z}$, then committed and posted to BB. In $\mathcal{T}$, each repayment b_j varies in a certain range Δ, but the total repayment amount is still equal to the desired repayment. Since $\mathcal{Z}$ can not predict the total number of lenders $u_{\mathcal{A}}$, we can not construct the repayment plan base on $u_{\mathcal{A}}$. Therefore, we choose to create the repayment plan based on a certain fixed denomination, *unit of loan* u, which could be different in different loans (investment projects) based on the M value. Each individual lending amount must be dividable by u. We denote by m the number of units. According to the above information, $\mathcal{Z}$ creates a raw payment table $\mathcal{R}$ (see Table 2), where row number $n_u = M/u$ and column number equals n_t. Each cell $v_{i,j}$ is a committed repayment amount based on u. Each $v_{i,j}$ also varies in a certain range with following condition:

$$b_j = \sum_{i=1}^{n_u} v_{i,j} \qquad (1)$$

The total repayment r of loan u:

$$r = \sum_{j=1}^{n_t} v_{i,j} \qquad (2)$$

Repayment Schedule					Subtotal
$1st$	$2nd$	$3rd$	$\ldots$	$n_t\ th$	
$[\![v_{1,1}]\!]$	$[\![v_{1,2}]\!]$	$[\![v_{1,3}]\!]$	$\ldots$	$[\![v_{1,n_t}]\!]$	r
$[\![v_{2,1}]\!]$	$[\![v_{2,2}]\!]$	$[\![v_{2,3}]\!]$	$\ldots$	$[\![v_{2,n_t}]\!]$	r
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
$[\![v_{n_u,1}]\!]$	$[\![v_{n_u,2}]\!]$	$[\![v_{n_u,3}]\!]$	$\ldots$	$[\![v_{n_u,n_t}]\!]$	r
$[\![b_1]\!]$	$[\![b_2]\!]$	$[\![b_3]\!]$	$\ldots$	$[\![b_{n_t}]\!]$	

Table 2: Raw Repayment Table $\mathcal{R}$

Since each row is one unit of investment in n_t periods, if the loan unit is m_j, the lender will be assigned to m_j rows. Further details about the assignment of rows in $\mathcal{R}$ will be explained in the lending phase.

$\mathcal{T}$ and $\mathcal{R}$ are committed to BB, along with the proof transcripts. $\mathcal{Z}$ proves that $\mathcal{R}$ follows the above requirements in Protocol 2.

Construction of Repayment Plan

(1) ZeroLender $\mathcal{Z}$ creates the Raw Repayment Table $\mathcal{R}$ of size $n_u \times n_t$ such that it follows the conditions defined in the Negotiation Phase. All the values in the table are committed.
- Every row adds to $u \cdot (I + 1)$
- Each column j adds to b_j
- Each number is within the specified range Δ.
- Any other condition agreed upon.

(2) For each row i of $\mathcal{R}$, use Schnorr's Protocol to prove knowledge of the discrete log of the following with respect to h:

$$\left(\prod_{j=1}^{n_t} [\![v_{i,j}]\!]\right) \cdot g^{-u\cdot(I+1)}$$

(3) For each column j of the $\mathcal{R}$, use Schnorr's Protocol to prove knowledge of the discrete log of the following with respect to h:

$$\left(\prod_{i=1}^{n_u} [\![v_{i,j}]\!]\right) \cdot [\![b_j]\!]^{-1}$$

(4) For each cell in table $\mathcal{R}$, use a range proof (adapted from [14]) to prove that each commitment $[\![v_{i,j}]\!]$ is in the range agreed in the Negotiation phase, unconditionally above 0.

Protocol 2: Proof of Proper Construction of Repayment Plan

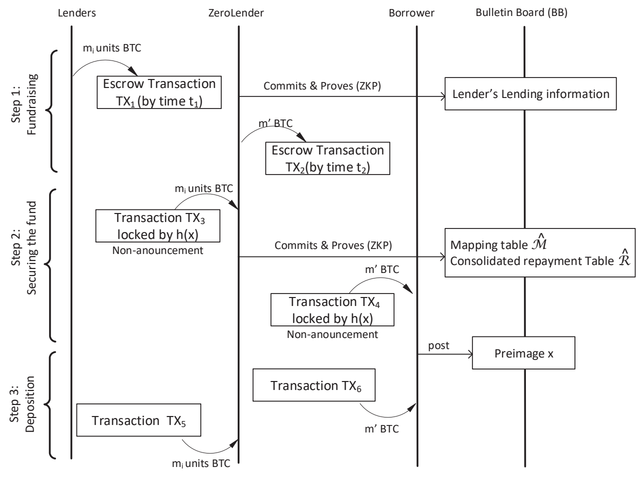


Figure 2: Lending phase

5.2 Lending Phase

In this phase, we integrate the CoinSwap model with zero-knowledge proofs at different steps, as shown in Figure 5.

5.2.1 Fundraising. After lenders compare borrower's project information and interest rate on BB, we assume lender $\mathcal{A}_j$ wants to lend m_j units of coins to $\mathcal{B}$ through $\mathcal{Z}$. $\mathcal{A}_j$ asks $\mathcal{Z}$ set up a payment channel. $\mathcal{A}_j$ escrows m_j on the blockchain to the escrow address $addr_{(\mathcal{A}_j,\mathcal{Z})}$ via a 2-of-2 escrow transaction denoted as $TX_{1(\mathcal{A}_j,\mathcal{Z})}$ that m_j can be claimed by any transaction signed by both $\mathcal{A}_j$ and $\mathcal{Z}$. $TX_{1(\mathcal{A}_j,\mathcal{Z})}$ is locked by by time window tw_1, which means after tw_1, $\mathcal{A}_j$ can reclaim her m_j units of coins. Again, as mentioned before, m_j must be multipliable of u. Similarly, other interested lenders individually escrow their desired investing amount: m_1, m_2, m_3, ..., m_n, m_{n+1}, ...on the blockchain via 2-of-2 escrow transaction with $\mathcal{Z}$, upon the funded coins exceed the proposed m in a certain of ratio, where $m = \sum_{n=1}^{n_u} m_i$. We let $\mathcal{Z}$ have a capability of over-fundraising in case some lenders change their mind before the deadline of fundraising and drop off this loan opportunity.

If $\mathcal{B}$'s loan request is successfully funded, similarly $\mathcal{Z}$ escrows m' units of coins on the blockchain to the escrow address $addr_{(\mathcal{Z},\mathcal{B})}$ via a 2-of-2 escrow transaction denoted as $TX_{2(\mathcal{Z},\mathcal{B})}$ and $TX_{2(\mathcal{Z},\mathcal{B})}$ is locked by time window tw_2, where $tw_1 > tw_2$ and m' is amount after $\mathcal{Z}$'s onetime origination fee, e.g. $m' = 0.09 \cdot m$, where origination fee is publicly known to all users. Transactions $TX_{1(\mathcal{A}_j,\mathcal{Z})}$ and $TX_{2(\mathcal{Z},\mathcal{B})}$ are confirmed on the blockchain.

On first-come, first-serve basis, the first $n_{\mathcal{A}}$ lenders whose investment coins equals M list on the lending list $\mathcal{L}$, extra lenders are list in the waiting list $\mathcal{L}'$ as backup. The commitment looks like:

$$[\![L_{id}, ZID_i, m_i, u, addr_{\mathcal{A}_j,1}, addr_{\mathcal{A}_j,2}, \ldots, addr_{\mathcal{A}_j,n_t}]\!]$$

Where $addr_{\mathcal{A}_j,1}, \ldots, addr_{\mathcal{A}_j,n_t}$ are $\mathcal{A}_j$'s repayments receiving addresses. We encourage lenders provide the total number of receiving addresses at least equal to the total numbers of repayments, which maximize lender's privacy. Therefore, different repayment receiving address will be used at each returning circle. $\mathcal{Z}$ opens commitment to verifiers to prove posted information. If any of proofs fails, it shows $\mathcal{Z}$ plays malicious. $\mathcal{A}$ aborts protocol.

5.2.2 Securing the fund. To secure the loan transaction, $\mathcal{B}$ generates a random number x with 256 bits of entropy and hashes it with the SHA-256 function, creating a hash value $H = h(x)$. $\mathcal{B}$ signs H and posts to BB. $\mathcal{B}$ keeps preimage of H as a secret. $\mathcal{Z}$ informs all intended lenders to secure the loan. We assume Lender $\mathcal{A}_j$ indeed want to invest on this project. Then $\mathcal{A}_j$ sends $\mathcal{Z}$ a Hashed Timelock Contract (HTLC) transaction with m_j units of coins from escrow address $addr_{(\mathcal{A}_j,\mathcal{Z})}$, which stipulates $\mathcal{Z}$ may claim the funds by presenting a valid preimage x of the hash digest H before given timeout tw_3. However, this transaction $TX_{3(\mathcal{A}_j,\mathcal{Z})}$ is not broadcasted on the blockchain. Similarly, all rest of assured lenders create similar HTLC transaction to $\mathcal{Z}$ locked by the same hash value H. If TX_3 has not been presented before the deadline for creating TX_3. In case 1, if lender $\mathcal{A}_j$ fail to send $TX_{3(\mathcal{A}_j,\mathcal{Z})}$, whose committed information should be omitted from $\mathcal{L}$, then lenders from $\mathcal{L}'$ moves to $\mathcal{L}$; in case 2, lender $\mathcal{A}_j$ sends the $TX_{3(\mathcal{A}_j,\mathcal{Z})}$, but trustless $\mathcal{Z}$ omits $\mathcal{A}_j$ information from $\mathcal{L}$. To solve this security issue, ZeroLender updates $\mathcal{L}$ by deleting failed lenders and moving lenders from $\mathcal{L}'$ to $\mathcal{L}$, $\mathcal{B}$ generates a new hash value H' and restarts this step. This stops $\mathcal{A}_j$ from blocking the protocol while also stopping $\mathcal{Z}$ from stealing

If the total amount of $TX_{3(\mathcal{A}_j,\mathcal{Z})}$ reaches the m, $\mathcal{Z}$ sends $\mathcal{B}$ a HTCL transaction with m' unit of coins from escrow address $addr_{(\mathcal{Z},\mathcal{B})}$, where m' is the amount after origination fee. Then $\mathcal{B}$ may claim the fund by present created preimage x before timeout tw_4, where $tw_3 > tw_4$. This transaction $TX_{4(\mathcal{Z},\mathcal{B})}$ is not broadcasted on the blockchain either.

At the end of this step, $\mathcal{Z}$ should have all $\mathcal{A}$'s lending information based on $\mathcal{L}$. $\mathcal{Z}$ now creates a mapping table $\mathcal{M}$, see Table 3. In $\mathcal{M}$,

Lender				
$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$	$\dots$	$\mathcal{A}_{n_A}$
$[\![0]\!]$	$[\![0]\!]$	$[\![1]\!]$	$\dots$	$[\![0]\!]$
$[\![1]\!]$	$[\![0]\!]$	$[\![0]\!]$	$\dots$	$[\![0]\!]$
$[\![0]\!]$	$[\![1]\!]$	$[\![0]\!]$	$\dots$	$[\![0]\!]$
$[\![1]\!]$	$[\![0]\!]$	$[\![0]\!]$	$\dots$	$[\![0]\!]$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
$[\![0]\!]$	$[\![0]\!]$	$[\![0]\!]$	$\dots$	$[\![1]\!]$
$[\![m_1]\!]$	$[\![m_2]\!]$	$[\![m_3]\!]$	$\dots$	$[\![m_{n_A}]\!]$

Table 3: Mapping Table: $\mathcal{M}$ is created by $\mathcal{Z}$,Where rows are mapping to rows in Raw Payment Table $\mathcal{R}$. Each column mapping to individual lender $\mathcal{A}_j$. Each cell is committed as 0 or 1 with conditions: Each row only have one 1 and each column adds to m_j, the number of lending unit of $\mathcal{A}_j$.

row i maps to same row position in $\mathcal{R}$ and each row have and only have one committed 1, which means mapped row in $\mathcal{R}$ is assigned to the lender $\mathcal{A}_j$ and the sum of column j equals $\mathcal{A}_j$'s lending units m_j. Therefore, in $\mathcal{M}$:

$$\sum_{j=1}^{n_A} \mathcal{M}_{i,j} = 1 \tag{3}$$

$$\sum_{i=1}^{n_u} \mathcal{M}_{i,j} = m_j \tag{4}$$

where plain cell value $\mathcal{M}_{i,j}$ is either 1 or 0. $\mathcal{Z}$ commits $\mathcal{M}$ to BB along with proof transcript. $\mathcal{Z}$ proves knowledge of Equation 3 and 4 according to Protocol 3.

To prevent $\mathcal{Z}$ dictate which lender is assigned which unit in the repayment plan with any bias, each $\mathcal{A}_j$ permutes row of $\mathcal{M}$ then commits permutation to BB. After all $\mathcal{A}$ open permutations, the final shuffled mapping table $\hat{\mathcal{M}}$ is released. If any $\mathcal{A}_j$ fails to open their permutation commitment, the lenders each create and commit a new permutation except the parties who failed to open their commitments, as they are now excluded from this process. The reasons and implications for this are discussed in Section 7. The Protocol 3 shows the process of $\hat{\mathcal{M}}$ construction.

Now we need to combine the tables $\mathcal{R}$ and $\mathcal{M}$. We first transform $\mathcal{R}$ to a 3-dimensional table by adding the third dimension k to the table $\mathcal{R}$, where k is column of the table $\mathcal{M}$. If the value in $\mathcal{M}$ hides 0, then value in 3-dimensional table equals 0, otherwise, equals to the value hidden in $\mathcal{R}$. Second, The newly generated values $v_{i,j,k}$ are committed to BB. $\mathcal{Z}$ proves this intermediary 3-dimensional table by following Protocol 4.

Finally, to increase the ZKP efficiency, all commitments belong to same lender are homomorphically added together, which transforms previous 3-dimensional table back to 2-dimensional table $\hat{\mathcal{R}}$. The finial consolidated repayment assignment table $\hat{\mathcal{R}}$ is shown in Table 4. The consolidation of repayment plan is illustrated in Protocol 4.

5.3 Returning phase

Before repayment due date, $\mathcal{B}$ prepares to pay back to $\mathcal{Z}$ according to repayment list $\mathcal{T}$ that is set up in the Negotiation Phase.

Assignment of Repayment Plan

(1) $\mathcal{Z}$ creates a committed Mapping table $\mathcal{M}$ of size $n_u \times n_A$. The table is created with the following conditions:
- Each row i adds to 1.
- Each column j adds to m_j.
- Each number in the table is either a 0 or a 1.

(2) Each $\mathcal{A}_j$ creates a row-wise permutation of $\mathcal{M}$ and posts a commitment to their permutation.

(3) Each $\mathcal{A}_j$ opens their commitments of the permutation on BB.
- If any parties refuse to open their commitments, go to step 2, except any lender that failed to open their commitment is barred from contributing a new permutation.

(4) $\mathcal{M}$ is shuffled following the lenders' permutations, resulting in $\hat{\mathcal{M}}$.

(5) $\mathcal{Z}$ proves the following on $\hat{\mathcal{M}}$:
- Each row i adds to 1. This is proved by using Schnorr's protocol to prove knowledge of the discrete log of the following with respect to h:
$$\left(\prod_{j=1}^{n_A} [\![\hat{\mathcal{M}}_{i,j}]\!]\right) \cdot (g^{-1})$$
- Each column j adds to m_j. This is proved by using Schnorr's protocol to prove knowledge of the discrete log of the following with respect to h:
$$\left(\prod_{i=1}^{n_u} [\![\hat{\mathcal{M}}_{i,j}]\!]\right) \cdot (g^{-m_j})$$

(6) For each investor A_j, $\mathcal{Z}$ opens column j of $\hat{\mathcal{M}}$ to A_j. On that j, for each i such that $\hat{\mathcal{M}}_{i,j}$ hides 1, $\mathcal{Z}$ opens row i of $\mathcal{R}$ to A_j.

Protocol 3: Assignment of Repayment Plan

Lender	Repayment schedule				
	$1st$	$2nd$	$3rd$	$\dots$	n_t th
$\mathcal{A}_1$	$[\![\hat{v}_{1,1}]\!]$	$[\![\hat{v}_{1,2}]\!]$	$[\![\hat{v}_{1,3}]\!]$	$\dots$	$[\![\hat{v}_{1,n_t}]\!]$
$\mathcal{A}_2$	$[\![\hat{v}_{2,1}]\!]$	$[\![\hat{v}_{2,2}]\!]$	$[\![\hat{v}_{2,3}]\!]$	$\dots$	$[\![\hat{v}_{2,n_t}]\!]$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
$\mathcal{A}_{n_A}$	$[\![\hat{v}_{n_{\mathcal{A}},1}]\!]$	$[\![\hat{v}_{n_{\mathcal{A}},2}]\!]$	$[\![\hat{v}_{n_{\mathcal{A}},3}]\!]$	$\dots$	$[\![\hat{v}_{n_{\mathcal{A}},n_t}]\!]$

Consolidated Repayment Table $\hat{\mathcal{R}}$

$[\![b_1]\!]$	$[\![b_2]\!]$	$[\![b_3]\!]$	$\dots$	$[\![b_{n_t}]\!]$

Total Monthly Repayment Table $\mathcal{T}$

Table 4: Consolidated repayment table $\hat{\mathcal{R}}$: each row match related lender $\mathcal{A}_j$. $\hat{v}_{i,j}$ is consolidated repayment value, where $b_j = \sum_{i=1}^{n_u} \hat{v}_{i,j}$.

5.3.1 *Creating repayment transactions.*

$\mathcal{Z}$ create each individual repayment transaction to related A_j according to $\hat{\mathcal{R}}$. The receiving address shall match one of the committed receiving address in $\mathcal{L}$.

5.3.2 *Proof of transaction.*

Without loss of generality, we assume $\mathcal{Z}$ generates n_{TX} repayment transactions and send total $\hat{v}_{i,j}$ coins to lender $\mathcal{A}_j$ for repayment time period j. $\mathcal{Z}$ commits $\hat{v}_{i,j}$ and $\mathcal{A}_j$'s receiving address *addr* to BB.

For each transaction TX_i in the anonymity set N, $\mathcal{Z}$ commits transaction amount σ_i if TX_i is from $\mathcal{Z}$'s address to $addr_{\mathcal{A}_j}$, otherwise, it commits 0. $\mathcal{Z}$ submits the proof transcription to BB.

Consolidation of Repayment Plan

(1) For each unit $i \in \mathbb{Z}_{n_u}$ and each lender $k \in \mathbb{Z}_{n_A}$:
- For each $j \in \mathbb{Z}_{n_t}$, $v_{i,j,k}$ be defined as follows:
  (a) If $[\![\hat{\mathcal{M}}_{i,k}]\!]$ hides 0, then $v_{i,j,k} = 0$.
  (b) If $[\![\hat{\mathcal{M}}_{i,k}]\!]$ hides 1, then $v_{i,j,k}$ is equal to the value hidden in the commitment $[\![v_{i,j}]\!]$ from $\mathcal{R}$
  (c) $v_{i,j,k}$ is committed as $[\![v_{i,j,k}]\!] = g^{v_{i,j,k}} h^{\mu_{i,j,k}}$ where random number $h^{\mu_{i,j,k}} \in \mathbb{Z}_q$
- Prove the following: Either ($[\![\hat{\mathcal{M}}_{i,k}]\!]$ hides 1 AND for each $j \in \mathbb{Z}_{n_t}$, $[\![v_{i,j,k}]\!]$ hides the same value $v_{i,j}$ in $\mathcal{R}$) OR ($[\![\hat{\mathcal{M}}_{i,k}]\!]$ hides 0 AND for each $j \in \mathbb{Z}_{n_t}$, $[\![v_{i,j,k}]\!]$ hides 0).

(2) All the commitments associated with lender k for time period j are homomorphically added. The resulting table $\hat{\mathcal{R}}$ is posted to BB as the Consolidated Repayment Table.

(3) Create a $n_A \times n_t$ table called $\hat{R}$:
- For each $k \in \mathbb{Z}_{n_A}$ and $j \in \mathbb{Z}_{n_t}$:

$$[\![\hat{v}_{k,j}]\!] = \prod_{i=0}^{n_u} [\![v_{i,j,k}]\!] = g^{\sum_{i=0}^{n_u} \hat{v}_{i,j,k}} h^{\sum_{i=0}^{n_u} \mu_{i,j,k}}$$

$$[\![\hat{v}_{k,j}]\!] = g^{\hat{v}_{k,j}} h^{\mu_{k,j}}$$

Protocol 4: Consolidation of Repayment Plan

Finally, a homomorphic addition of commitments all σ_i

$$[\![\beta']\!] = \prod_{i=0}^{n} [\![\hat{\sigma}_i]\!]$$

$\mathcal{Z}$ uses Schnorr's protocol to prove $\mathcal{Z}$ follows the above requirements in Protocol 5.

Proof of Transaction

(1) Begin with a commitment of the total value $[\![\beta]\!] = g^{\beta} h^{\alpha}$ and a commitment to the output address, $[\![addr]\!] = g^{addr} h^{\gamma}$

(2) For each transaction TX_i in the anonymity set of size n:
  (a) Let σ_i be defined as follows:
  - If TX_i is from $\mathcal{Z}$ to the committed lender $\mathcal{A}_i$'s account, then σ_i is set to the amount in the transaction going to the output.
  - Else, $\sigma_i = 0$

  (b) Pick a random number $r_i \in \mathbb{Z}_q$
  (c) Create a commitment $[\![\sigma_i]\!] = g^{\sigma_i} h^{r_i}$
  (d) $\mathcal{Z}$ proves the following: Either ($\mathcal{Z}$ knows the private key(s) associated with the input address AND the value committed in $[\![\sigma_i]\!]$ is equal to the output amount AND the commitment $[\![addr]\!]$ is equal to the destination address) OR $[\![\sigma_i]\!]$ hides 0.

(3) Homomorphically combine all the $[\![\sigma_i]\!]$:

$$[\![\beta']\!] = \prod_{i=0}^{n} [\![\sigma_i]\!] = g^{\sum_{i=0}^{n} \sigma_i} h^{\sum_{i=0}^{n} r_i} = g^{\beta'} h^{r}$$

(4) Use Schnorr's protocol to prove knowledge of the discrete log of the following with respect to h:

$$[\![\beta']\!] \cdot [\![\beta]\!]^{-1}$$

Protocol 5: Proof of Transaction

5.3.3 Finishing repayment transaction. $\mathcal{B}$ verifies each individual repayment transaction which shall match the value committed in $\hat{\mathcal{R}}$. If all verifications are true, $\mathcal{B}$ sends $\mathcal{Z}$ b_j coin to close this repayment circle.

6 SYSTEM DISCUSSION

In this section, we discuss more about role of borrower and different payment situations.

6.0.1 Covert Borrower. We assume borrowers are covert, meaning that they are willing to deviate from the protocol, but they are unwilling to get caught. In fact, borrower can play malicious, collude with ZeroLender. For example, Verifying the repayments from ZeroLender to the lenders is irrelevant to borrower's repayment. To find methods to protect against potential malicious behavior of the borrower would be an interesting future direction to take this work. Due to the nature of the protocol, it is impossible to protect against a fully malicious adversary, as the borrower can simply refuse to repay ZeroLender. However, it may be possible to make the protocol secure against covert borrowers, where the borrowers are willing to cheat but unwilling to get caught.

On the other hand, P2P lending platforms in the market put all risks on lenders. Lenders have to choose to trust platform's risk evaluation, credit report, etc. If borrower is in financial insolvency, the only lost for platform is service fee. Our model bind platform with lenders. If borrower is insolvency, ZeroLender loses one repayment since it is pre-paid to lenders, which may lead ZeroLender do more rigorous screening on borrowers.

6.0.2 Early payoff or partial payment. Some types of loan can be paid off early and there are a few different ways to go about it. Borrowers may choose make larger monthly payment, multiple payments each billing cycle, or-if available they may even choose to pay off the loan in one lump sum. Similarly, if borrower only has capability to make part of repayment, it will cause payment penalty. All these activities involve in interest and repayment recalculations. More details can be found in [11]. Though it is a limitation that we assume that the borrower is capable of paying, the protocol is capable of handling variable repayment plans where the interest is payed monthly and they can choose multiple repayments. To accomplish this, the rows of the $\mathcal{R}$ add to $u \cdot I$ instead of $u(I+1)$ and the columns would contain the repayment amount times the interest as well. The plans are then mapped to lenders and the table is consolidated as prescribed in the Lending phase. Interest is calculated by taking every non-payed column from the consolidated and homomorphically adding them, resulting in total interest. If the Borrower wishes to repay multiple repayments, they can say so a month in advance. They then take values in the selected repayment columns and homomorphically divide it by the interest ($[\![\hat{v}_{i,j}]\!]^{(I^{-1} \bmod q)}$), which results in the amount of the repayment. This is added to the interest, and $\mathcal{Z}$ is required to repay the interest as prescribed in the table, the homomorphic addition of the row plus the repayment of part of the principle. Using a similar method, we can also support partial repayments, though such repayments would harm the privacy of the transactions.

7 SECURITY ANALYSIS

7.1 Privacy by Simulation

Now, we will prove that privacy is maintained against a covert $\mathcal{B}$ and a malicious $\mathcal{A}_j$ through the protocols. Recall that, though this model allows for a malicious $\mathcal{Z}$, they are required to have the details (accounts information) of both the $\mathcal{A}$ and the $\mathcal{B}$ so that direct secure communication is not required. If a party wishes to have privacy from ZeroLender, they can use a mixing service, e.g. TumbleBit [9], to achieve privacy.

Theorem 7.1. *Privacy is preserved against a covert $\mathcal{B}$ in the Negotiation phase.*

To prove Theorem 7.1, we provide a simulator $\mathcal{S}$ that can simulate a valid transcript of Protocol 2.

(1) $\mathcal{S}$ generates the Raw Repayment Table $\mathcal{R}'$ populated with random group elements.
(2) $\mathcal{S}$ uses the simulator for Schnorr's protocol (in Protocol 1) to simulate the proof that the product of the rows hide $u \cdot (I+1)$.
(3) $\mathcal{S}$ uses the simulator for Schnorr's protocol that the product of the columns hide the same values as b_j in $\mathcal{T}$.
(4) $\mathcal{S}$ runs the simulator for the range proof (Adapted from [14]) on each group element in $\mathcal{R}'$.

□

Theorem 7.2. *Privacy is preserved against a covert $\mathcal{B}$ in the Lending Phase.*

To prove Theorem 7.2 we need to prove that both Protocols 3 and 4 of the Lending phase are privacy preserving.

Lemma 7.1. *Privacy is preserved against a covert $\mathcal{B}$ in the Assignment of the Repayment Plan (Protocol 3)*

To prove Lemma 7.1, we provide a simulator $\mathcal{S}$ to simulate the view of $\mathcal{B}$.

(1) $\mathcal{S}$ creates random group elements C_j for each lender $\mathcal{A}_j$ representing their $[\![m_j]\!]$
(2) $\mathcal{S}$ has all the lenders approve the commitments on BB.
(3) $\mathcal{S}$ has $\mathcal{Z}$ generate a fake mapping table $\mathcal{M}'$ with random group elements.
(4) $\mathcal{S}$ has lenders commit permutations of $\mathcal{M}'$.
(5) $\mathcal{S}$ has lenders open their permutation commitments of $\mathcal{M}'$.
(6) $\mathcal{M}'$ is shuffled according to the permutations, resulting in $\hat{\mathcal{M}}'$.
(7) $\mathcal{S}$ uses the simulator for Schnorr's protocol to prove that the product of the rows each hide 1.
(8) $\mathcal{S}$ uses the simulator for Schnorr's protocol to prove that the product of each of the columns hides the same value as its respective C_j.

□

Lemma 7.2. *Privacy is preserved against a covert $\mathcal{B}$ in the Consolidation of Repayment Plan (Protocol 4).*

To prove Lemma 7.2, we provide a simulator $\mathcal{S}$ to simulate the view of $\mathcal{B}$.

(1) $\mathcal{S}$ chooses a random group element $w_{i,j,k}$ for each supposed commitment $[\![v_{i,j,k}]\!]$.
(2) For each unit i and each repayment k, $\mathcal{S}$ uses the simulator for the zero knowledge proof used to prove the following statement Either ($\hat{\mathcal{M}}'_{i,k}$ hides 1 AND for each $j \in \mathbb{Z}_{n_t}$, $w_{i,j,k}$ hides the same value $v_{i,j}$ in $\mathcal{R}'$) OR ($\hat{\mathcal{M}}_{i,k}$ hides 0 AND for each $j \in \mathbb{Z}_{n_t}$, $w_{i,j,k}$ hides 0).
(3) The $w_{i,j,k}$ values are homomorphically combined as prescribed in Protocol 4 to produce $\hat{\mathcal{R}}$.

□

Theorem 7.3. *Privacy is preserved against a malicious $\mathcal{A}_j$ in the Lending phase.*

To prove Theorem 7.3, we need to prove that both Protocol 3 and 4 of the Lending Phase can be simulated.

Lemma 7.3. *Privacy is preserved against a malicious $\mathcal{A}_j$ in the Assignment of the Repayment Plan (Protocol 3).*

To prove Lemma 7.3, we provide a malicious simulator that is able to extract $\mathcal{A}_j$'s inputs and use them to simulate a view indistinguishable from the ideal model.

Define the ideal model as follows: The $\mathcal{A}_j$'s input is his permutation. Based on that permutation, the Trusted Third Party (TTP) returns the values of the rows they were assigned, as well as the result of the total permutation. A lack of a permutation is interpreted as the primitive permutation p ($1 \rightarrow 1$, $2 \rightarrow 2$, etc).

(1) $\mathcal{A}_j$ generates $\mathcal{R}'$ using random group elements, as detailed in Theorem 7.1's proof.
(2) $\mathcal{A}_j$ has $\mathcal{Z}$ generate a fake $\mathcal{M}'$ with random group elements.
(3) $\mathcal{S}$ has the all lenders commit permutations of $\mathcal{M}'$.
(4) If $\mathcal{A}_j$ fails to post a commitment, $\mathcal{S}$ extracts the primitive permutation p.
(5) If $\mathcal{A}_j$ posted a commitment to a permutation:
   - $\mathcal{S}$ has the lenders open their commitments.
   - If the $\mathcal{A}_j$ does not open the commitment:
     (a) $\mathcal{S}$ extracts the primitive permutation p.
     (b) $\mathcal{S}$ uses p as $\mathcal{A}_j$'s input to the TTP and obtains $\mathcal{A}_j$'s outputs.
     (c) $\mathcal{S}$ rewinds to the construction of the Raw Repayment Table $\mathcal{R}'$ and constructs it to match the TTP outputs. Specifically, the rows that will be assigned to $\mathcal{A}_j$ contain committed values to be given to $\mathcal{A}_j$.
     (d) $\mathcal{S}$ proceeds exactly as before to the point where $\mathcal{A}_j$ refused to open their permutation commitment.
     (e) $\mathcal{S}$ has the simulated lenders create permutations that, when combined, it is equivalent to the TTP's permutation.
     (f) $\mathcal{S}$ had the simulated lenders open their commitments.
     (g) All parties shuffle $\mathcal{M}'$ using the permutation $\hat{\mathcal{M}}'$
     (h) $\mathcal{S}$ opens the lender's column of $\hat{\mathcal{M}}'$, as well as their assigned rows from $\mathcal{R}'$ and the protocol concludes.
   - If the permutation is invalid, $\mathcal{S}$ extracts the primitive permutation p.
   - Else, $\mathcal{S}$ extracts the $\mathcal{A}_j$'s permutation p.
(6) $\mathcal{S}$ uses the permutation p as $\mathcal{A}_j$'s input to the TTP.
(7) $\mathcal{S}$ rewinds to the construction of the Raw Repayment Table $\mathcal{R}'$ and constructs it to match the TTP outputs. Specifically,

the rows that will be assigned to $\mathcal{A}_j$ by the total permutation will contain commitments to the values to be given to $\mathcal{A}_j$.

(8) $\mathcal{S}$ continues to the construction of $\mathcal{M}'$. $\mathcal{S}$ constructs the table so that the $\mathcal{A}_j$'s column will, after the total permutation, result in the appropriate rows being assigned to $\mathcal{A}_j$.
(9) $\mathcal{S}$ has the simulated lenders construct permutations such that the result combined with the permutation p.
(10) $\mathcal{S}$ has the $\mathcal{A}_j$'s commit their permutation.
(11) $\mathcal{S}$ has the $\mathcal{A}_j$'s open their commitments.
(12) All parties shuffle $\mathcal{M}'$ using the permutation $\hat{\mathcal{M}}'$
(13) $\mathcal{S}$ opens the lender's column of $\hat{\mathcal{M}}'$, as well as their assigned rows from $\mathcal{R}'$.

□

Lemma 7.4. *Privacy is preserved against a malicious $\mathcal{A}_j$ in the Consolidation of the Repayment Plan (Protocol 4).*

$\mathcal{A}_j$ has no inputs nor a role in for the Consolidation of the Repayment Plan, therefore they have no inputs or interactions in which they can be malicious in this step. Therefore, we use the same simulator as the borrower for Protocol 4. □

Theorem 7.4. *Privacy is preserved against a covert $\mathcal{B}$ in the Returning Phase.*

To prove Theorem 7.4, we provide a simulator $\mathcal{S}$ to simulate the view of $\mathcal{B}$ in Protocol 5 for repayment i.

- In the Lending Phase, each of the the simulated lenders approved a random group element α' representing a commitment their address for this repayment.
- For each lender, there exists a random group element $\hat{v}'_{i,j}$ in the Repayment Table $\hat{\mathcal{R}}'$ representing the repayment amount.
- For each lender $\mathcal{A}_j$
  - For each transaction in the anonymity set N
    (1) $\mathcal{S}$ chooses a random group element σ'_i
    (2) $\mathcal{S}$ uses the simulator for the Zero Knowledge Proof used to prove the following statement: Either ($\mathcal{S}$ knows the private key(s) associated with the input address AND the value committed in σ'_i is equal to the output amount AND the commitment $addr'$ is equal to the destination address) OR σ'_i hides 0.
  - $\mathcal{S}$ uses Schnorr's protocol's simulator to prove that $\prod \sigma'_i$ hides the same value as $\hat{v}_{i,j}$

□

Theorem 7.5. *Privacy is preserved against a malicious $\mathcal{A}_j$ in the Returning phase.*

To prove Theorem 7.5, we provide a malicious simulator $\mathcal{S}$ to simulate the view of $\mathcal{A}_j$ in Protocol 5 for repayment i.

We assume that the block chain contains the transactions to the lender's account and that these transactions are in the anonymity set. However, $\mathcal{S}$ does not know the private keys associated with the sending accounts.

- In the Lending phase, each of the the simulated lenders approved a random group element α' representing a commitment his address for this repayment. At the same time, $\mathcal{A}_j$ confirmed his receiving address commitment $[\![addr]\!]$
- $\mathcal{A}_j$ confirms that the bitcoins have been transferred to his account.
- For each lender, there exists a random group element $\hat{v}'_{i,j}$ in the Repayment Table $\hat{\mathcal{R}}'$ representing the repayment amount.
- For each lender $\mathcal{A}_j$
  - For each transaction in the anonymity set N
    (1) $\mathcal{S}$ chooses a random group element σ'_i
    (2) $\mathcal{S}$ uses the simulator for the Zero Knowledge Proof used to prove the following statement: Either ($\mathcal{S}$ knows the private key(s) associated with the input address AND the value committed in σ'_i is equal to the output amount AND the commitment $addr'$ is equal to the destination address) OR σ'_i hides 0.
  - $\mathcal{S}$ uses Schnorr's protocol's simulator to prove that $\prod \sigma'_i$ hides the same value as $\hat{v}'_{i,j}$.

□

7.2 Fund Security

As lenders, one of the most concerns is that ZeroLender steals or occupies funds/repayment. We analyze fund security in several cases.

7.2.1 ZeroLender $\mathcal{Z}$ is corrupt. We want to show that desired bitcoins from all lenders $\mathcal{A}$ can be passed to borrower $\mathcal{B}$. In the Lending phase, if fund is successfully raised, which means securing the fund step is done, CoinSwap paradigm guarantees that $\mathcal{B}$ get correct loan. Otherwise, $\mathcal{B}$ will not post preimage of the hash value after the TX_3 and TX_4. If fund is partially raised, then $\mathcal{Z}$ only gets some TX_3s, there is no reason for Z to continue to create TX_4 since $\mathcal{Z}$ will lose coins if this fundraising fails.

Moreover, ZKPs for table $\mathcal{R}$, $\hat{\mathcal{R}}$, $\mathcal{M}$ and $\hat{\mathcal{M}}$ guarantees all repayment are fair to $\mathcal{A}$. Even though $\mathcal{Z}$ joins lender's party as a real lender, $\mathcal{Z}$ can not operate hidden value in these tables since tables are shuffled by $\mathcal{A}$. In the Returning phase, Protocol 5 requires $\mathcal{Z}$ prepay all repayments before $\mathcal{Z}$ get repayment from $\mathcal{B}$. Therefore, $\mathcal{Z}$ can not deny planed repayment in the Returning phase.

7.2.2 Lender $\mathcal{A}_j$ is corrupt. We want to show that malicious $\mathcal{A}_j$ can not claim he does not receive repayment or repayment amount is not correct. First, in the Negotiation phase, $\mathcal{A}_j$ committed his receiving addresses on the lending list $\mathcal{L}$; second, in the Lending phase, final repayment plan $\hat{\mathcal{R}}$ has decided repayment amount, last, in the Returning phase, any verifier can verify if the repayment matches information in the $\mathcal{L}$ and $\hat{\mathcal{R}}$.

In the Lending phase, securing the fund step, there are two places, $\mathcal{A}_j$ may play malicious: (1) $\mathcal{A}_j$ refuses to send TX_3 to $\mathcal{Z}$ and (2) if $\mathcal{A}_j$ is the last one to open the permutation of $\mathcal{M}$, he may refuse to open the permutation. Our solution is to reboot the securing the fund step. For the case one, $\mathcal{A}_j$ will be replaced by the lender in the waiting list. $\mathcal{A}_j$ wastes his transaction fee for the TX_1; for the case two, $\mathcal{A}_j$ losses his shuffling right in the new securing the fund step. Our Protocols guarantee that the permutation is valid if one of the lenders are honest.

7.2.3 Lenders $\mathcal{A}$ and $\mathcal{Z}$ collude. Now suppose $\mathcal{A}_j$ and $\mathcal{Z}$ collude try to harm other honest $\mathcal{A}$. Basically, this can be treated as two cases above. Since our protocol is designed to allow $\mathcal{Z}$ as a valid lender, moreover, $\mathcal{Z}$ as a platform knows all related information, collusion can not make it get extra information.

8 PERFORMANCE ANALYSIS

8.1 Theoretical Analysis

THEOREM 8.1. *The Negotiation Phase has a runtime complexity of* $O(n_u \cdot n_t \cdot |\Delta|)$.

Proof. This protocol consists of three distinct sections. The commitment of the table, the proofs that the rows and columns add to their appropriate values, and the proofs that each repayment is within the range proof.

To populate the plaintext Raw Repayment Table $\mathcal{R}$, it requires an operation of order $O(n_u \cdot n_t)$. Then, each value needs to be committed. Since each commitment is $O(1)$, this total operation is $O(n_u \cdot n_t)$.

After this, ZeroLender must prove each row adds up to $u \cdot (I+1)$. To do this, first all the commitments from each row must be homomorphically added. Homomorphically adding one row is $O(n_t)$. Since this operation happens on each row, it results in a $O(n_u \cdot n_t)$. Similarly, the column proofs also has a $O(n_u \cdot n_t)$.

The range proofs now have to be executed. Since the range proof is executed on each value in the table, $n_u \cdot n_t$ range proofs must be executed. Each individual range proof has a $O(|\Delta|)$. Therefore, the complexity of the range proofs is $O(n_u n_t \cdot |\Delta|)$. It should be noted that it would be perfectly reasonable for a service to have a constant unit size and a constant range. If that were the case, then the range proofs would be constant and the result would be $O(n_u \cdot n_t)$.

This results in $O(n_u \cdot n_t \cdot |\Delta|)$ for the entire construction phase. □

THEOREM 8.2. *The Lending Phase has a runtime complexity of* $O(n_u \cdot n_{\mathcal{A}} \cdot n_t)$.

Proof. The Lending phase consists of two major sections: The Assignment of the Repayment Plan (Protocol 3) and the Consolidation of the Repayment Plan (Protocol 4).

The Assignment of the Repayment Plan. This section consists of three major parts: The creation and committing of the mapping table $\mathcal{M}$, the shuffling of the $\mathcal{M}$, and proving the rows and columns add to the appropriate values.

The creation and committing of the table requires the choosing and commitment of $n_u \cdot n_{\mathcal{A}}$ bits, therefore, this step has $O(n_u \cdot n_{\mathcal{A}})$.

The shuffling of the table consists of two minor steps. Each lender has to create a permutation, which takes $O(n_u)$. The method we employed to commit these permutations takes $O(n_u)$ time. After this, all the parties have to open all the other lender's commitments, which means that each party has to run $O(n_{\mathcal{A}} \cdot n_u)$. Then, the permutations are applied, which requires n_u row repositions for each lender, resulting in $O(n_u \cdot n_{\mathcal{A}})$, though this part has a very small coefficient associated with it.

The proofs that the rows and columns add to the appropriate values both have $O(n_u \cdot n_{\mathcal{A}})$, much like the similar proofs in the $\mathcal{R}$ analysis.

Therefore, the creation and shuffling of $\mathcal{M}$ is $O(n_u \cdot n_{\mathcal{A}})$

Consolidation of the Table. The consolidation of the table $\hat{\mathcal{R}}$ consists of a proof on every cell of the $mat\hat{h}calR$, a n_t by n_u table, with each investor, resulting in a complexity of $O(n_u \cdot n_{\mathcal{A}} \cdot n_t)$. Then, each investor's commitments from the proofs are homomorphically added together, resulting in $O(n_u \cdot n_{\mathcal{A}} \cdot n_t)$ additions. Thus, the runtime complexity of the consolidation of the table is $O(n_u \cdot n_{\mathcal{A}} \cdot n_t)$.

Therefore, the runtime complexity of the Lending Phase is $O(n_u \cdot n_{\mathcal{A}} \cdot n_t)$. □

THEOREM 8.3. *One return period of the Returning Phase protocol has a runtime complexity of* $O(n_u \cdot n_{\mathcal{A}})$.

Proof. This phase consists of a constant series of proofs for each elements in an anonymity set N for each investor. If the size of the anonymity list is $|N|$, then the complexity of this section is $O(|N| \cdot n_A)$ for each repayment. □

8.2 Experimental Evaluation

We have developed a proof-of-concept implementation of ZeroLender to test the performance of our protocol in Java 1.8. Bouncy Castle [13], a standard cryptographic library for Java is used to handle the elliptic curve operations. We performed our tests on a PC with i7-6700 CPU @ 3.4GHZ and 32GB RAM.

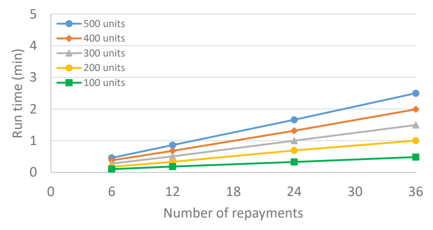
All results are measured in computational time rather than real time to show computational work. Many of these protocols contain significant portions of *embarrassingly parallel* computations, so this can be multithreaded for performance. To reflect the amount of computational work required, we used a single threaded implementation.

In difference phase of ZeroLender, we choose different variables to show the ZKP construction time, verification time, and proof size of the ZKP transcript.

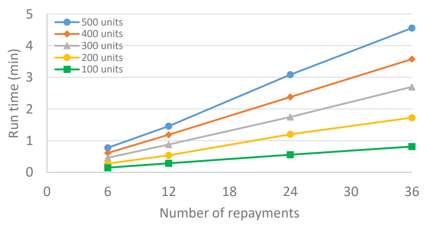
8.2.1 Negotiation phase. In this phase, we have tested the Protocol 2 by varying numbers of repayments u_t (6, 12, 24, 36 month) and number of unit m (100, 200, 300, 400, 500). Figure 3.a and 3.b show protocol computation time of construction and verification respectively. Figure 3.c shows proof transcript size. As expected, the computation time and transcript size increase linearly with the number of repayments. Moreover, computation time and transcript size also increase linearly with the number of units. By comparing construction computation in Figure 3.a and verification computation in Figure 3.b, we notice verification is more expensive than construction computation.

8.2.2 Lending phase. In this phase, using same setting in the Negotiation phase plus constant lenders ($n_A = 60$), we have tested the Protocol 3 and 4. Figure 4 also shows the computation time and transcript size increase linearly with the number of repayments. Again, computation time and transcript size also increase linearly with the number of units. Also, verification is more expensive than construction computation. Comparing to the Negotiation phase, the proof of Lending phase is a more expensive operation. One thing to note about these experiments is that the slopes on the graphs in Figure 4 do not converge at 0 due to the performance of the permutations not being effected by the number of repayments.

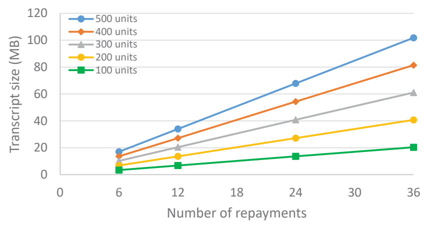
In Figure 5, we hold the number of repayments (n_u = 24 months), number of units (m = 300) constant, vary the number of lenders. This figure demonstrates the linear scaling of computation time



(a) Construction Time

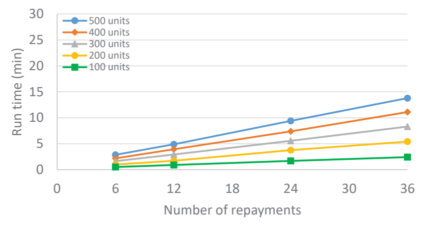


(b) Verification Time

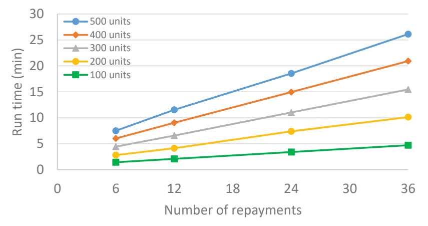


(c) Proof Size

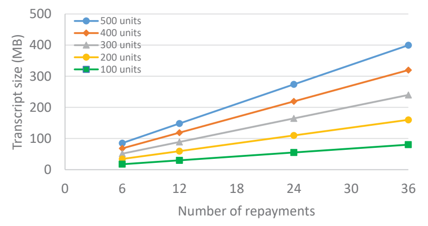
Figure 3: Performance for Protocol 2 (Construction of repayment plan) in Negotiation phase



(a) Construction Time



(b) Verification Time



(c) Proof Size

Figure 4: Performance for Protocols 3 & 4 in the Lending phase (the number of lenders is held constant at 60).

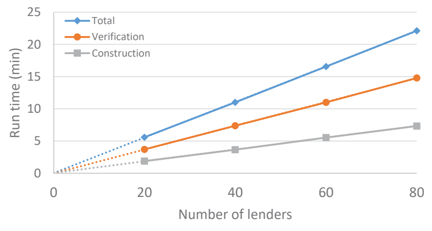


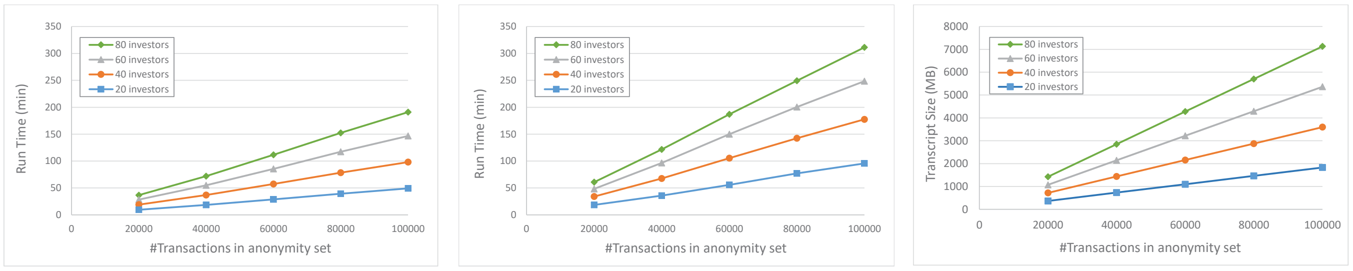
Figure 5: Performance with different joined lenders

and size of transcript with respect to number of lenders. It should be noted that this is the most time sensitive section of the protocol. The Negotiation phase can have the proofs pending while ZeroLender gathers lenders and the Returning phase can be generated and verified throughout the month as the blocks are added to the blockchain. The Lending phase is the only set of proofs where the proofs can not be created beforehand and there is no other work that can be done in the protocol while the proofs are being verified. To reduce the delay at this point, the consolidation phase, can be spread out across the months. The experimental results demonstrate Protocols 3 and 4 scale linearly in construction, verification time and proof size to its inputs: the proof of Protocol 2 scales with number of repayments. the proof of Protocol 3 and 4 scales with number of repayments and number of lenders. The verification of the protocol require more time to the construction of the proof.

8.2.3 Returning Phase. In this phase, we simulate the execution of a single month of returning transactions. To do this, we use a set of transactions and a list of committed values representing the amount each investor is owed. We then execute Protocol 5 on each account and investor. The results are shown in Figure 6. These graphs showcase the linear scaling with both the number of investors and the number of transactions. It should be noted that, in the context of ZeroLender as a whole, this portion can be executed over the course of an entire month as the blocks come in. ZeroLender can choose the anonymity set by choosing transactions from the blocks as they are created, given enough time to be assured that the block isn't on a terminating branch.

9 RELATED WORK

In Bitcoin, if an address is ever linked to a user's identity, all related transactions will be linked to the user as well. To avoid such problem, mixing service [21] are used to mix bitcoins and obscure the trail back to the fund's original source. Even though ZeroLender is not a mixing service, ZeroLender closely related work. In the lending phase, ZeroLender is related to mixing services [2] [9] [17], where a mixing address receives coins from multiple clients and forwards them to fresh addresses. Mixcoin [2] uses a Trusted Third Party (TTP), which could steal users' bitcoins. TumbleBit [9] uses RSA-puzzle solving protocol which enables sender to pay the receiver one denomination unit. However, TumbleBit requires the sender and the receiver to interact with each other and to exchange puzzle information. Moreover, to achieve anonymity, every transaction has to be set to an equal denomination. CoinShuffle [17] is secure against thefts but suffering the Bitcoin's maximum transaction size (100KB) limits, which has scalability issue. In our lending



(a) Construction Time **(b) Verification Time** **(c) Proof Size**

Figure 6: Performance for Protocols 5 being run with 20, 40, 60, and 80 investors with a range of transactions.

phase, the activity of ZeroLender collecting investment and forwarding to the borrower is similar to a mix service, but all these aforementioned mixing services either need interactive communication between senders and receivers or the mixing service has to be trusted. Furthermore, some mix services such as TumbleBit has unusual transaction script which may expose Bitcoin addresses involved in mixing activities through blockchain.

Zero-knowledge proof (ZKP) allows a prover to assure a Verifier that they have knowledge of a secret or statement without revealing the secret itself [10]. Participated parties prove they are following the protocol properly without revealing their secrets [3]. Schnorr proposed first efficient Σ-*protocol* [4], which is simple to prove security with a simulator. Comparing to ZeroLender, Zcash uses zero-knowledge succinct non-interactive argument of knowledge (zk-SNARKs) [19] to provide enhanced privacy and recently, later, Ethereum integrated zk-SNARKs in the form of smart contracts.

To mitigate or avoid vulnerability of the exchange system, Dagher et al. [3] presents *Provision* which is a privacy-preserving proof of solvency for Bitcoin exchanges without revealing Bitcoin addresses, total holding and liabilities. Inspired by *Provision*, ZeroLender uses ZKP in all three phases. In the negotiation phase, ZKP is used to prove the raw repayment plan. In the lending phase, ZKP is used to prove final (consolidated) repayment plan. Finally, in the returning phase, ZKP is used to prove proper repayment distribution from ZeroLender.

10 CONCLUDING REMARKS

In this paper, we have presented ZeroLender, a trustless P2P lending platform in Bitcoin. We developed a proof-of-concept implementation to test the performance of our protocol. Our experiments demonstrate that the computation time and storage size overhead are linear with respect to number of payments. Finally, although we have focused on the P2P lending platform in this paper, our model is compatible with other sharing economy, e.g. group buying and Crowdfunding, with minor modifications. Also, these concepts can be used on other similar cryptocurrencies.

REFERENCES

[1] Krishnan S. Anand and Ravi Aron. 2003. Group Buying on the Web: A Comparison of Price-Discovery Mechanisms. *Management Science* (2003), 1445–1615.

[2] Joseph Bonneau, Arvind Narayanan, Andrew Miller, Jeremy Clark, Joshua A. Kroll, and Edward W. Felten. 2014. Mixcoin: Anonymity for Bitcoin with Accountable Mixes. In *18th Int. Conf. Financial Cryptography and Data Security.* 486–504.

[3] Gaby G. Dagher, Benedikt Bünz, Joseph Bonneau, Jeremy Clark, and Dan Boneh. 2015. Provisions: Privacy-preserving Proofs of Solvency for Bitcoin Exchanges. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security (CCS).* 720–731.

[4] Ivan Damgard. 2010. On Σ-protocols. http://www.cs.au.dk/~ivan/Sigma.pdf.

[5] U. Feige and A. Shamir. 1990. Witness Indistinguishable and Witness Hiding Protocols. In *the Twenty-second Annual ACM Symposium on Theory of Computing.* 416–426.

[6] Maxwell Gregory. 2013. CoinSwap: Trasnsaction graph disjoint trustless trading. https://bitcointalk.org/index.php?topic=321228.0/.

[7] Olena Havrylchyk and Marianne Verdier. 2018. The Financial Intermediation Role of the P2P Lending Platforms. *Comparative Economic Studies* (2018), 115–130.

[8] Carmit Hazay and Yehuda Lindell. 2010. *Efficient Secure Two-party Protocols.* Springer.

[9] Ethan Heilman, Leen Alshenibr, Foteini Baldimtsi, Alessandra Scafuro, and Sharon Goldberg. 2017. TumbleBit: An Untrusted Bitcoin-Compatible Anonymous Payment Hub. In *24th Annual Network and Distributed System Security Symposium.*

[10] Markus Jakobsson and Ari Juels. 2000. Mix and Match: Secure Function Evaluation via Ciphertexts. In *Advances in Cryptology — ASIACRYPT 2000.* 162–177.

[11] Leslie Jane, Federer Vaaler, and James Daniel. 2009. *Mathematical Interest Theory.* The Mathematical Association of America.

[12] Venkat Kuppuswamy and Barry L. Bayus. 2018. *Crowdfunding Creative Ideas: The Dynamics of Project Backers.* 151–182.

[13] Legion of the Bouncy Castle Inc. [n.d.]. *Bouncy Castle Crypto APIs.*

[14] Wenbo Mao. 1998. Guaranteed correct sharing of integer factorization with off-line shareholders. In *Public Key Cryptography.* 60–71.

[15] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. http://www.bitcoin.org/bitcoin.pdf.

[16] Torben Pryds Pedersen. 1992. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In *Advances in Cryptology - CRYPTO '91.* 129–140.

[17] Tim Ruffing, Pedro Moreno-Sanchez, and Aniket Kate. 2014. CoinShuffle: Practical Decentralized Coin Mixing for Bitcoin. In *19th European Symposium on Research in Computer Security.* 345–364.

[18] Nicolas van Saberhagen. [n.d.]. CryptoNote v2.0. https://github.com/monero-project/research-lab/blob/master/whitepaper/whitepaper.pdf.

[19] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. 2014. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy.* 459–474.

[20] C. P. Schnorr. 1990. Efficient Identification and Signatures for Smart Cards. In *Advances in Cryptology - CRYPTO' 89.* 239–252.

[21] Usman W. Chohan. 2017. The Cryptocurrency Tumblers: Risks, Legality and Oversight. *SSRN Electronic Journal* (2017).

[22] Gavin Wood. 2014. Ethereum: A secure decentralised generalised transaction ledger. http://gavwood.com/paper.pdf.