Confidential Distributed Ledgers for Online Syndicated Lending

by

Le Wang

A Thesis

presented to

The University of Guelph

In partial fulfilment of requirements
for the degree of

Master of Science

in

Computer Science

Guelph, Ontario, Canada

ABSTRACT

CONFIDENTIAL DISTRIBUTED LEDGERS FOR ONLINE SYNDICATED LENDING

Le Wang
University of Guelph, 2022

Advisor(s):
Dr. Xiaodong Lin

With the boom of Fintech in emerging markets, online syndicated lending enables Internet borrowers to apply for loans through an agent bank and receive the result within seconds. It is of great significance to build a trusted, fair, and regulation-complaint collaboration model while maintaining confidentiality to protect the sensitive financial information of individual syndicate lenders. In this paper, we present a novel collaborative financial ledger for online syndicated lending. The proposed ledger-enabled MPC leverages homomorphic encryption to enable the efficient reuse of intermediary data without breaking the privacy during the full life-cycle of a loan. Our system further minimizes the privacy leakage in the settlement phase by carefully aggregating daily loan share and repayment information. Besides, our framework enables the regulator to efficiently verify the encrypted transactional activities. We streamline the framework design to optimize performance for the real-world requirements and develop a prototype system on a permissionless blockchain platform.

ACKNOWLEDGEMENTS

I would like to thank the following people, without whom I would not have been able to complete this research, and without whom I would not have made it through my master's degree!

First, I wish to thank my dissertation committee. Without their guidance, I would not have made it. Dr. Xiaodong Lin and Dr. Rozita Dara served as wise committee members who went above and beyond to help me reach my goal.

To my advisor, Dr. Xiaodong Lin, I would like to express my deepest appreciation. His rigorous academic attitude taught me to be conscientious and hardworking, his profound academic knowledge guided me to find out the right research questions, and his insightful advice made me understand how to be an excellent researcher.

To my friends, my parents, and my siblings: you put up with my distractions and missing many events. I am forever grateful for your patience and understanding. I think I had more time to spend with each of you now.

Finally, to my wife, Wenjing Zhang, and my son, Alex: your love and understanding have been a ray of light in my dark times, illuminating me and making me move forward. Without your trust in me, I never would have made it. Now is the time to celebrate, you have earned this degree with me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Introduction

In this chapter, we briefly describe the research background of online syndicated lending firstly, then the research motivation is to be introduced. At last, we mainly present the objectives and contributions of our work.

1.1 Overview

Finance has become an essential part of human society and money plays an important role in every part of people's life. Nevertheless, most of the capital is generated from trade and lending. That is why; lending has been a domain of rapid development and growth over the years, and morphing into more efficient and creative lending forms. On the other hand, due to the advent of fintech, there has been exponential growth in the lending sector. Furthermore, with the emergence of mobile funding lenders, online lending institutions, and P2P platforms, people can borrow money faster and easier than before, and have the ability to live a more comfortable life [1].

Specially, with the development of ICT technologies and the increasing requirement of capital security, the form of lending has changed a lot. when a borrower requires an amount (US$1 million or more) that is too large for a single lender or when the loan is outside the scope of a lender's risk exposure levels [2]. Multiple lenders pool together and form a syndicate to provide the borrower with the requested capital. When the borrower repays the loan, the distribution of profits is completed in proportion to the contribution. This model is known as syndicated lending (or traditional syndicated lending), which refers to the process of involving a group of lenders that fund various portions of a loan for a single borrower. Obviously, syndicated lending is often used in corporate financing. Firms seek corporate loans for a variety of reasons, including funding for mergers, acquisitions, buyouts, and other capital expenditure projects. These capital projects often require large amounts of capital that typically exceed a single lender's resource or underwriting capacity.

Generally, the syndicated lending is organized in the following pattern: a borrower (such as corporation, large project, government, etc.) makes a loan request to an agent institution who coordinates a group of financial institutions to jointly provide funding to the borrower; the agent will also distribute the received repayment to the syndicate lenders according to their respective loan shares [3,4]. As a collaborative financial activity, the global syndicated lending reached US$5.5 trillion during full year 2021, representing a 51% increase in total proceeds compared to the full year 2020 and the strongest annual period for lending since record began. By number of deals, more than 10,000 loans reached financial close in full-year 2021, an increase of 21% compared to 2020 [5].

With the boom of FinTech, online syndicated lending market has been rapidly growing and surged to US$283 billion in emerging market economies (e.g., China). Contrary to the above traditional syndicated lending market, Internet users in online syndicated lending can ubiquitously access the loan services through an online financial agent. With

a massive user base, the agent is able to intrigue and connect to hundreds of lending sources to secure the requested loans and remarkably improve the efficiency of financial delivery and settlement [6].

In the traditional syndicated lending, lending information flows only among stakeholders and is subject to strict confidentiality clauses and regulatory protections. However, compared with it, agents of online syndicated lending collect a large amount of personal information and lending information of users and lenders, and due to the novelty and versatility of this lending form, it is difficult to be effectively supervised by regulators, which creates great potential risks for the information and capital safety of users and lenders. Therefore, we propose a comprehensive and innovative solution based on distributed ledger and cryptography primitives, and demonstrate the effectiveness and security of the solution through rigorous experiments.

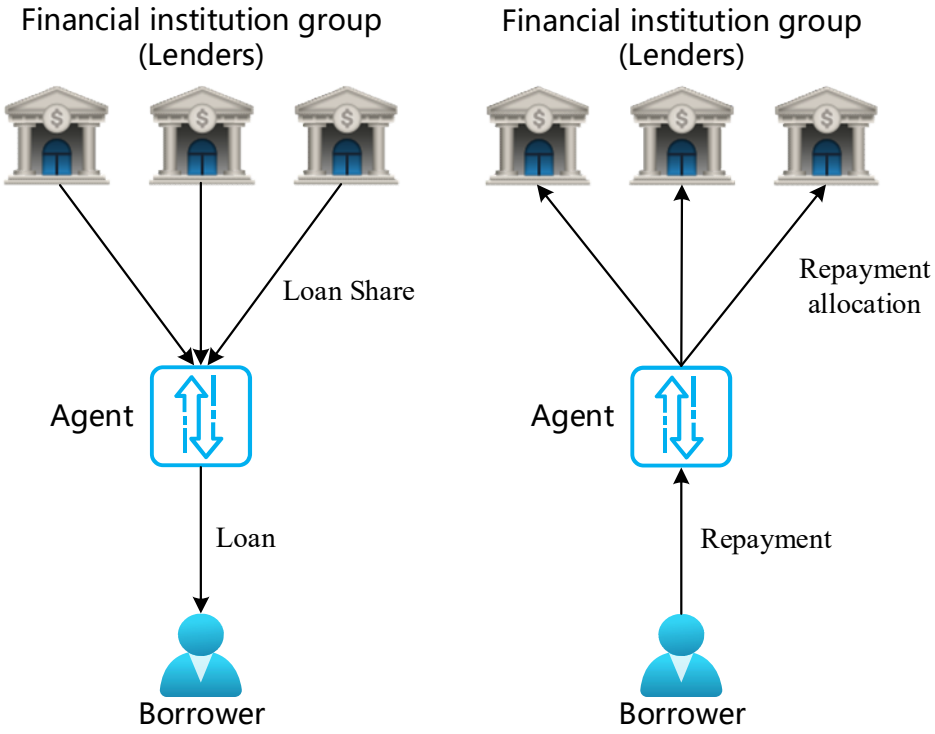
1.2 Motivation

In contrast to traditional banks, relying on the rapid development of technologies, many licensed fintech companies have emerged in several countries, such as WeBank, the largest online syndicated lending finance company in China, which has reached 321 million users and managed a total of 782.9 billion CNY in loans by the end of 2021, up 34% from the end of 2020 [7]. Obviously, as an online syndicated lending intermediary, WeBank controls a huge amount of personal data and borrowing and repayment data. If it wants to do any evil, it will bring huge losses to borrowers and lenders. On the other hand, it is clear from a variety of media that regulatory agencies are becoming aware of the security risks of online syndicated lending. As a result, various financial security bills have been introduced, requiring fintech companies to provide online syndicated lending solutions with both privacy protection capabilities and friendly regulatory capabilities.

From the participant's perspective, the online syndicated lending is performed by borrowers (users), agent and multiple lenders. From the process composition perspective, the online syndicated lending is composed of two phases: **collection** and **distribution**. In the collection phase as shown in Figure 1.1(a), a borrower makes a loan request to an agent institution who coordinates a group of financial institutions (or lenders) to jointly provide funding to the borrower; when the borrower repays capital, the agent will also distribute the received repayment to the syndicate lenders according to their respective loan shares (distribution phase as shown in Figure 1.1(b)).

Above all, it is clear that different roles have different requirements for the online syndicated lending system. First of all, borrowers want to protect their personal information, loan information and loan amount from others. One lender wishes to keep its contribution and repayment distributions for each loan safe from other lenders and agents to avoid speculation and attacks on its own risk model. On the contrary, if an agent is malicious, it has an incentive to profit from analyzing the behavior of particular borrowers and lenders over the multiple lending and repayment. Besides, all participants of online syndicated lending expect to establish an effective mechanism to ensure that they can

quickly view, track and verify their own lending or repaying process and information, and that the information is open and transparent to all of the participants. Finally, online syndicated lending was built without much consideration for regulation, leading to regulatory compliance issues that can easily induce greater financial risks and bring greater property losses to lenders and borrowers.



(a) Loan collection **(b) Repayment distribution**

Figure 1.1: Example of a caption for a figure.

Therefore, a comprehensive solution for online syndicated lending should satisfy the following main requirements.

1. **Transparent and Unbiased**. The process of online syndicated lending should be transparent and unbiased to all participants. This is crucial to build trust among the borrower, agent and syndicate lenders.

2. **Confidentiality**. Confidentiality must be achieved to protect valuable financial information, i.e., loan shares (or “contributions”, we use them interchangeably) and repayment distributions in each lending service. Considering that a large number of loan requests and responses would enable to launch model extraction attacks [8,9,10] to the lenders' risk models, thus we expect that even the agent bank should not be able to gain advantages by playing the role of the loan coordinator.

3. **Auditable**. The Internet financial services are required to be auditable to regulatory entities, such that all transactions and related activities could be verified by stakeholders.

1.3 Objectives and Contributions

Recently, distributed ledgers for blockchain technology have been considered a promising fintech solution [11,12,13]. The trust enforced by the distributed infrastructure creates a fair environment for all players. Transactions can be verifiably recorded/executed by the financial ledgers. However, confidentiality is missing in the current design, which limits the types of the financial products that can be delivered. Concerns for retaining sensitive information on chain may further kill auditability and diminish the advantages brought by blockchain. Great efforts have been made to study confidential ledgers and privacy-preserving transaction execution [14,15,16,17,18,19,20,21]. However, the provided transactional privacy is either insufficient for our requirements [15,16,17,22], or suitable for specific financial products [19,20,21]. For example, privacy-preserving value transfer [19,20] can be realized by using zero-knowledge proof (ZKP) [23,24], but they only fit applications in peer-to-peer models and do not support collaborative financial activities that are the focus of this work. i.e., transactional collaboration among multiple financial institutions.

Intuitively, secure multi-party computation (MPC) [25,26,27] can enable such collaborative financial paradigm. However, by further looking into the concerned "collection-then-distribution" application, there are at least three main reasons why the existing MPC designs are unsuitable for the online syndicated lending.

First, the MPC does not provide privacy guarantee beyond the computation paradigm. In an online syndicated lending application, the loan share of each lender is extremely sensitive as discussed above. Using MPC for computing collection is effective in protecting these private inputs. Nevertheless, the corresponding funds collection from each lender to the agent will still expose the sensitive loan shares. Similar information leakage occurs in the repayment distribution phase. Second, the MPC is always a self-governing process, that is, the intermediate states in both garbled-circuit [25] and share-based [27] MPC are unable to be reused. However, the online syndicated lending consisting of two logically associated phases, where each loan collection will be followed by a repayment distribution in a subsequent period. A trivial solution is to design MPCs for these two phases and instantiate a dedicated proof to verify the correct association, which adds non-negligible workloads to the participants. Further, another important reason that makes a traditional MPC solution even more unattractive is the inability of keeping track of the transactional activities for a mandatory auditing function in a privacy-preserving way.

In this work, we present a confidentiality-centered collaborative financial ledger for the online syndicated lending service. The core innovation is a new MPC framework that allows us to keep and reuse the intermediate states during the computation while not undermining the security. To this end, we generate homomorphic public commitments for

lenders' loan contributions and repayment allocations for each loan. By aggregating these homomorphic commitments, daily settlement between each lender and the agent can be achieved with minimized privacy leakage. Similar to prior work [21,28,29,30,31,32], participating institutions collectively maintain a distributed ledger for fairness and providing verifiable data input for the off-chain MPC-enabled decision making. The entire framework is streamlined for the real-world online lending model and fully auditable by being able to securely record and cryptographically reveal the transactional evidence.

We formalize and prove the security of the proposed scheme, which may be useful to inform designs in a similar model. We implement the application in a consortium blockchain due to regulations and inherent difficulty of permissionless lending services [33]. But the proposed core technical framework is agnostic to the ledger types and can be adapted to a permissionless setting. Therefore, this study may benefit a wide range of public distributed applications. We summarize our contributions as follows:

1. **Confidential collaborative financial ledger**. We present a new ledger-based MPC framework for the online syndicated lending. The system supports confidential transaction recording on-chain and secure computation over transactions off-chain during the full life cycle of the loan. We formally define and prove the security of this new type of ledger. The underlying MPC framework may be of independent interest to applications in a similar model.

2. **Regulation-friendly auditing**. The audibility is supported in the proposed ledger system. A regulator with the associated private key can examine loan details and lending activities that are recorded on the blockchain to ensure the regulatory compliance. We accomplish this goal by allowing the regulator to publicly check the private states of the MPC for loan request processing.

3. **Prototype implementation and evaluation**. We develop a prototype system running on top of a leading consortium blockchain [34]. The thorough system evaluation shows a satisfactory performance that meets the real-world requirements, i.e., a minimal 75 millisecond delay for loan application processing and 31.6 times greater throughput versus the practical demand.

1.4 Thesis Outline

The rest of the thesis is organized as follow: the traditional syndicated lending, online syndicated lending, blockchain architecture, the difference between permissionless and permissioned blockchain platforms, consensus protocol, smart contract, distributed ledger privacy method, distributed ledger for cryptographic protocols and cryptographic building blocks covered in this work will be firstly presented in Chapter 2. Chapter 3 will go through the primary methodology, where we introduce the problem statement, proposed algorithms and the confidentiality-centered collaborative financial ledger for online syndicated lending, along with the analysis of the security and privacy of the designed solution. The performance evaluation will be discussed in Chapter 4, which

demonstrates that our proposed system has satisfactory performance on a real workload with a large syndicated of lenders. Chapter 5 will present the conclusion and future work.

2 Related Work and Background

In this Chapter, we will provide a comprehensive introduction to the basics involved in this article, including but not limited to traditional syndicated lending, online syndicated lending, the architecture of blockchain technology, the relationship of blockchain and distributed ledger, blockchain basic technologies, existing approaches to blockchain data privacy protection, and the related cryptographic building blocks.

2.1 Syndicated Lending

To establish a clearer understanding of online syndicated lending, we provide a comprehensive comparison of traditional syndicated lending and online syndicated lending in terms of definitions, participants, processes, advantages and disadvantages in this section. It is to be highlighted that online syndicated lending is referred to as opposed to traditional syndicated lending in order to add distinction in this thesis, which is often called syndicated lending/loan in industry and research circles. The detailed differences between online syndicated lending and traditional syndicated lending are shown below.

2.1.1 Traditional Syndicated Lending (TSL)

In general, traditional syndicated lending is served for large amount financing or loans. The borrowers always are corporations, large projects or sovereign governments and the lenders commonly are well-known banks or financial institutions.

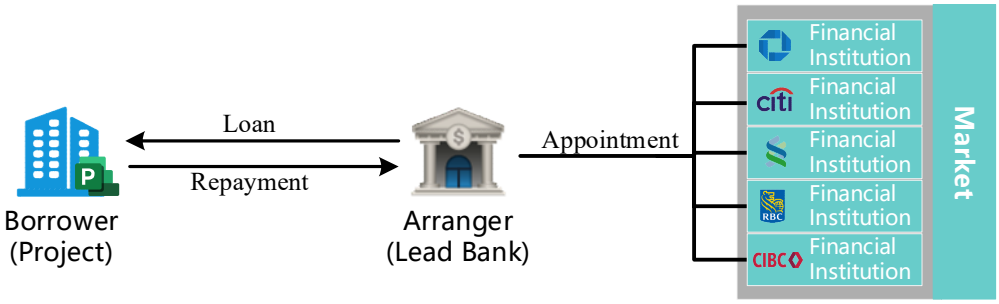


Figure 2.1: An example of traditional syndicated lending.

1. Definition

The traditional syndicated lending is also named as syndicated loan or syndicated bank facility. In short, it's defined as a loan facility tendered by a bunch of lenders to a borrower that needs a lot of money. Generally, as shown in Figure 2.1, traditional syndicated lending means a group of lenders (referred to as a syndicate) works together to provide funds for a specific borrower. The borrower can be a company, an individual and large project or a sovereign nation. The loan is usually composed of a fixed amount of money, a credit line, or a combination of both. The traditional syndicated lending occurs when a project requires a loan that is too large for a single lender, or when a project necessitates

a specialized lender who must have extensive experience in a particular asset type. A portion of the loan amount is contributed by an individual lender in the syndicate, who all share in the risk of the loan. One of the lenders acts as the arranger (lead bank) and manages the loan by representing the other lenders in the syndicate. As mentioned above, the syndicate is likely to be a mix of various kinds of loans, each with different repayment terms which are determined by negotiations between the lenders and the borrower [35].

2. Participants

The participants in a traditional loan syndication might differ from deal to deal, but typical participants consist of the borrower, agent, lead bank, and participating banks [36], as detailed below.

- **Borrower**. As mentioned above, the borrower is usually a company, large project or sovereign that initiates a syndicated loan due to the need for a large capital investment for project expansion, transaction acquisition, state investment, etc. The borrower is primarily responsible for the loan, repayment of principal and interest, proof and pledge of collateral, etc.

- **Agent**. In a syndicated loan, the agent is the link between the borrower and the lenders and serves as a bridge for information sharing and management between the two parties. In addition, the agent has contractual obligations to both the borrower and the lenders. Specifically, the agent's role to the lenders is to provide them with accurate information to enable them to exercise their authority under the syndicated loan agreement. However, the agent has no fiduciary responsibilities and offers no recommendations to the borrower or lenders.

- **Lead Bank**. The lead bank, also referred to as arranger, is entrusted by the borrower to organize the funds according to specific agreed loan terms. The bank must confirm that other lenders are ready to participate in the loan syndicate and share the lending risks together. The responsibilities and obligations between the lead bank and the borrower will be recorded in the contract, which describes the size of the loan, the repayment timetable, the rate of interest, the duration of the loan, and any other costs associated with the loan. It is noteworthy that the lead bank holds a significant portion of the loan and has a responsibility to allocate the flow of funds among the other participating lenders.

- **Participating Banks**. Based on the borrower information, collateral and other relevant information shared by the agent, the participating banks consider the contract terms agreed between the lead bank and the borrower, their own cash flow, risk model, loan interest rate and other factors to determine their own contribution ratio. Once both the lead bank and the participating banks' contribution ratios are determined, all lenders contribute to the borrower in accordance with the contractual requirements.

3. Process

The successful completion of the traditional syndicated lending requires seamless cooperation between multiple financial institutions and the borrower based on a series of supporting documents, and it's a complex and delicate financial operation coordinated by the lead bank. As shown in the Figure 2.2, the syndication process includes steps such as borrower application, lead bank evaluation, participating banks invitation, contract conclusion, loan payment and repayment, interest distribution between lenders, etc. [37] The processing details of each step are as follows.

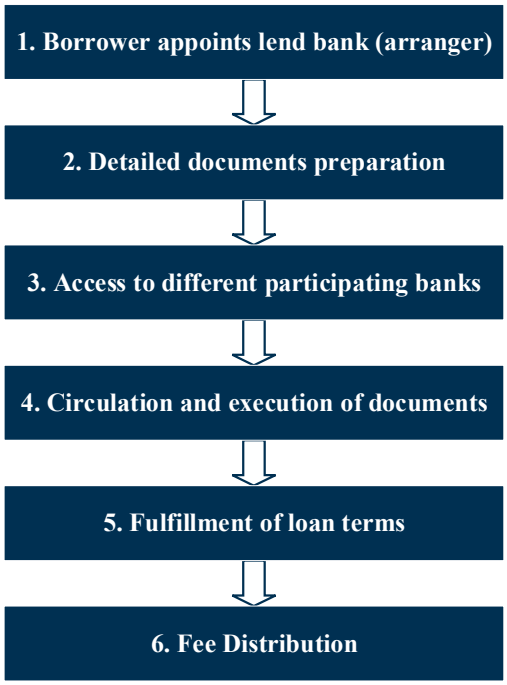


Figure 2.2: The procedure of traditional syndicated lending.

- **Borrower appoints lead bank (Arranger)**. Borrower should choose a lead bank or a long-term partner bank from the market. After completing the initial negotiation of preliminary terms and the structure of the loan, the lead bank arranges the loan for the borrower. As a result, the lead bank receives a fee as a percentage of the amount of the arranged loan. The fee relies on such parameters as the complexity of the transaction, the existing loans, the credit rating of the borrower, and the financial strength of the borrower.

- **Detailed documents preparation**. After reviewing and studying the borrower's documents, the lead bank produces the detailed loan report within a required format. Finally, the lead bank and borrower submit the loan request to the financial institution. Particularly, the documents that may be required for this process include, but are not limited to company profile along with financial data, collateral security and guarantees

provided, company's legal documents, and Directors' detailed information and their credit scores.

- **Access to different participating banks**. The lead bank then approaches different banks and invites them to participate in the loan syndication transaction, specifying the share of the loan syndication that they are willing to undertake. After that, the lead bank prepares a document that sets out the terms and conditions for the approval of the loan. This document is also known as the “term sheet”. The term sheet contains the loan amount, interest rate, collateral, repayment schedule, and other terms and conditions.

- **Circulation and execution of documents**. Once the syndicate has made its commitment, the loan documents are circulated among the banks for review and execution. Additionally, the lead bank is required to assist in completing the execution of the documents, such as payment of stamp duty, submission of documents, etc. It is important to highlight that the execution of the documents depends upon the type of assets secured as collateral. Collateral is shared on an equal basis, i.e., each lender is ranked according to their contribution to the syndicate.

- **Fulfillment of loan terms**. After the loan documents are signed, the borrower must meet the conditions set forth in the loan agreement. Once met, all lenders will disburse the loan. Also, there may be conditions that follow the agreement. The borrower must satisfy these conditions after the loan is disbursed in order to receive the subsequent loan.

- **Fee Distribution**. After the loan is disbursed, the lead bank will charge the borrower a transaction fee (service fee) based on a percentage of the total loan amount. After the borrower repays the loan, each lender will distribute the principal and interest according to the contract. In addition, any lender has no authority to take enforcement action in any syndicated loan transaction, and any change to the loan contract requires the support of a majority of all lenders.

4. Pros and Cons

At this point, we have completed a thorough introduction to traditional syndicated lending in terms of definition, participants, and process. To further clarify its main features, the advantages and disadvantages will be described in this part, as shown in the Figure 2.3.

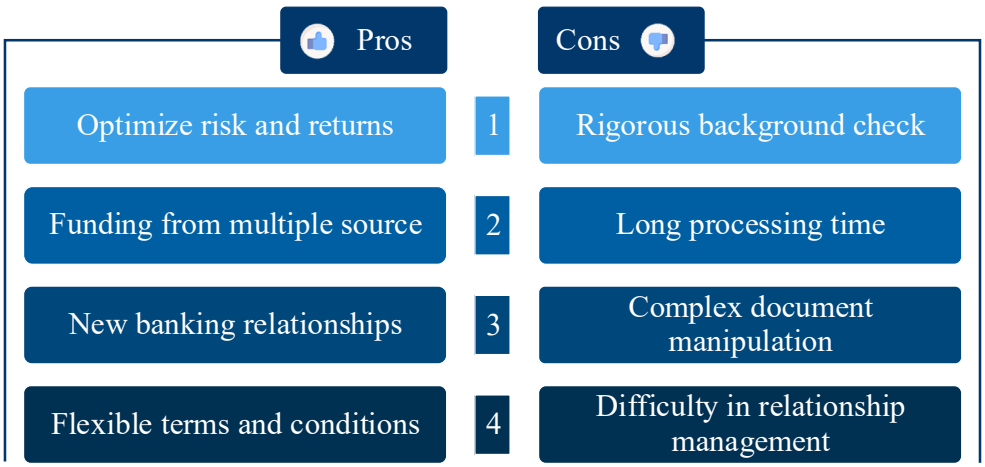


Figure 2.3: The pros and cons of traditional syndicated lending.

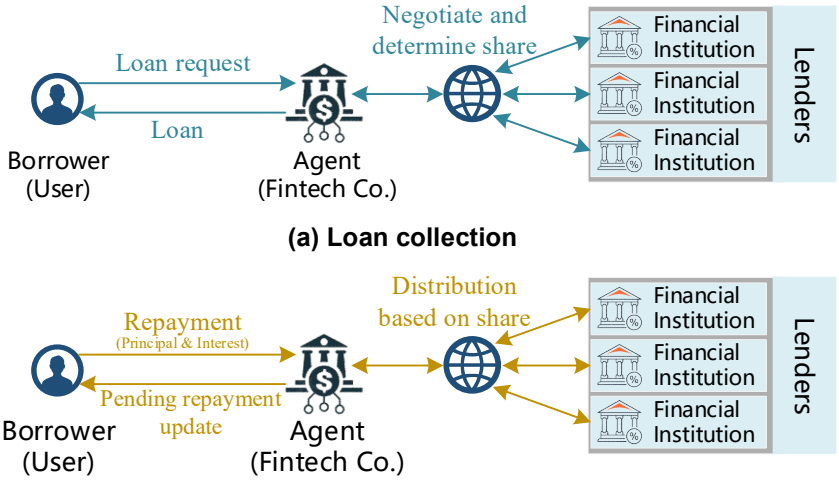
2.1.2 Online Syndicated Lending (OSL)

In contrast with the traditional syndicated loan that aims for securing a large loan amount at a time, online syndicated loan has emerged with the development of financial technology as an Internet-based financial service. Its main service group is individuals. Large Internet companies complete user credit risk assessment as agents through a large amount of user data deposited, and cooperate with banks to realize syndicated loans to users. Usually, the loan amount is within tens of thousands of dollars.

1. Definition

The online syndicated lending is also named as online syndicated loan. Generally, online syndicated lending means a group of lenders (referred to as a syndication) work together to provide loans for ordinary people through a fin-tech company that acts as an intermediary. Compared with traditional syndicated lending, the borrowers of online syndicated lending are ordinary person who are long-term users of products of Internet company (such as Tencent, Alibaba) and the agent is the above Internet company or its subsidiary (such as WeBank, which is a subsidiary of Tencent). Moreover, the online syndicated loan process is much simpler, taking only a few minutes from the time the user submits a loan application to the time he or she receives the money.

As we have mentioned, the online syndicated lending is composed of two phases: collection and distribution. A user (borrower) makes a loan request to an agent institution who coordinates a group of financial institutions to jointly provide funding to the borrower (collection phase as shown in Figure 2.4(a)); the agent will also distribute the received repayment to the syndicate lenders according to their respective loan shares [3,4] (distribution phase as shown in Figure 2.4(b)). As a collaborative financial activity, the online syndicated lending reached US$2.1 trillion in US in 2019 and has totaled around US$900 billion globally in the first quarter of 2020 [4,5].



(a) Loan collection

(b) Repayment distribution

Figure 2.4: Two phases of online syndicated lending.

2. Participants

The constituent members of online syndicated lending are relatively stable and mainly composed of borrowers, agents and lenders. The tasks of each role are shown below.

- **Borrower**. In online syndicated lending, the borrower is usually ordinary people. As a rule, they are long-term users of an Internet company (agent) and have accumulated a large amount of behavioral data and consumption data, based on which the agent is able to achieve credit assessment of them to ensure the security of the online syndicated loans.

- **Agent**. The agent's role is similar to that of the lead bank in traditional syndicated lending. As mentioned above, the agent is usually played by the Internet company or its subsidiary in charge of the financial business. Its main responsibilities are to evaluate the borrower's credit, determine the borrower's loanable amount based on the above evaluation results, receive the borrower's loan application, negotiate with lenders on the share of contribution, sign the loan contract with the borrower, and form a syndicated loan with lenders for transfer to the borrower's account, etc.

- **Lenders (Financial Institutions)**. Based on the syndicated loan agreement between the agent and lenders, when the agent receives a loan request from a user who meets the credit requirements, it needs to negotiate with lenders on the syndicated loan contribution ratios and provides the loan to the user based on the ratios. When the loan matures, the agent will distribute the principal and interest according to the contribution ratios of different lenders and refund to the fund accounts of different lenders.

3. Process

The online syndicated loan processing is all done online and does not contain any paper documents. The user only needs to perform a simple operation within a specific app (such as WeChat [38]) to issue a loan application, and the agent and lenders will complete the negotiation of the contribution percentages based on the user's creditworthiness, the generation of the syndicated loan, and the transfer of funds to the user's designated account. The processing details of each step are as shown in the Figure 2.5.

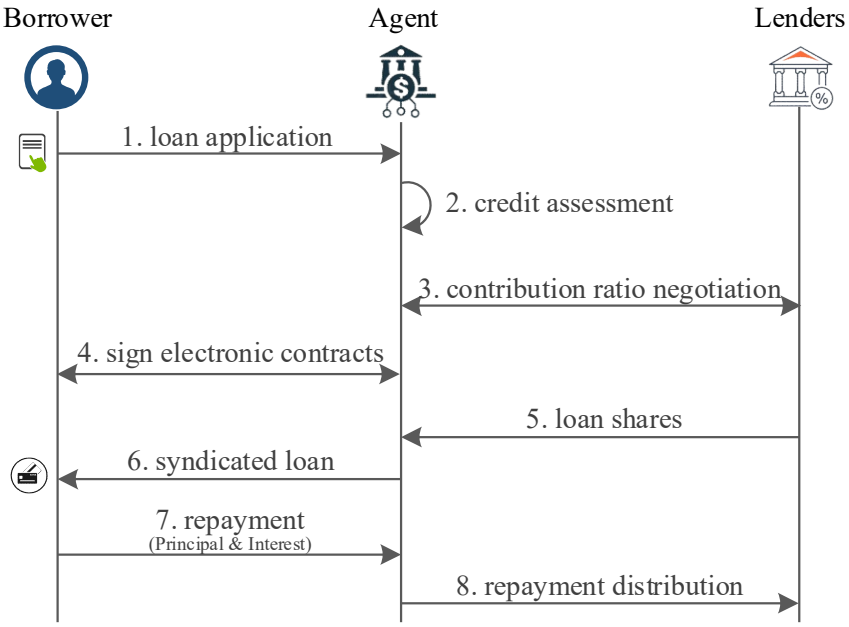


Figure 2.5: The procedure of online syndicated lending.

- Borrower launches loan application.

As illustrated above, borrowers can initiate syndicated loan applications with just a few simple actions on a specific app (such as WeChat [38], Alipay [39], etc.). In particular, when a user wants to take out a loan, he or she first needs to complete an assessment of the loan amount available and then can initiate an online syndicated loan with an agent.

- Agent assesses borrower's credit.

Upon receiving the syndicated loan application from the borrower, the agent evaluates the user's credit based on a large amount of user data accumulated in the backend of the app. Once approved, the agent notifies the user that it will accept the application.

- Agent and lenders negotiate their respective loan ratios.

Based on the borrower's loan amount and credit assessment, the agent will contact and negotiate with the appropriate lenders to finalize the respective contribution percentages and contractual terms required.

- Agent and borrower sign electronic contracts.

Once the agent has determined the percentage of each lender's contribution, it will generate an electronic contract and send it to the user for inspection. If the contract is no problem, the user has to send back to the agent the result of the check named “Contract is correct”.

- Lenders transfer the share of the loan assumed to the agent.

Upon receiving the user's confirmation of the contract results, the agent needs to contact the lenders participating in the syndicated loan and inform them to transfer the amount of the percentage of contribution assumed to the appointed agent's account.

- Agent transfers the syndicated loan to the borrower's account.

After the above steps completed, the agent will receive the loan funds from each lender and confirm that every lender's contribution ratio meets the contractual requirements. The agent then pools its own loan funds and those of all lenders together to form a syndicated loan and transfer it to the account designated by the borrower.

- Borrower repays principal and interest to the agent.

In accordance with the contract, the borrower repays the principal and interest to the account designated by the agent upon maturity of the loan. The agent confirms that there is no error and notifies the borrower that the loan has been repaid.

- Agent allocates the principal and interest in proportion to the lenders' contributions.

In this step, the agent makes distributions of principal and interest according to the percentage of each lender's contribution as specified in the contract and transfers them to the bank accounts designated by lenders.

4. Pros and Cons

At this point, we have made a thorough introduction to online syndicated lending in terms of definition, participants, and process. To further clarify its main features, the advantages and disadvantages will be described in this part, as shown in the .

Figure 2.6: The pros and cons of online syndicated lending.

2.1.3 The differences between traditional and online syndicated lending

With the description above, this section has provided a relatively comprehensive and detailed introduction to traditional and online syndicated lending models. In order to more clearly understand the differences between them, a comparison in terms of borrower, loan method, loan amount, agent, and lender is presented as shown in Table 2.1.

Table 2.1: The comparison of traditional and online syndicated lending.

Compare Items	TSL	OSL
Borrower	Company/Project/Country	Internet user
Loan Method	Mainly offline	Fully online
Paperless	Yes	No
Loan Amount ($)	Millions/Billions	Thousands
Agent Type	Large banks	Fin-tech companies
Lender Type	Large Banks/Financial institutions	Banks
Time Consumption	Weeks	Minutes

2.2 Blockchain Technology

Cryptocurrencies are gaining increasing attention from academia and industry in recent years. Cryptocurrencies are based on the blockchain underlying platform (e.g., Ethereum, Solana, etc.) that stores all current and historical transactions and is maintained by distributed miners [40]. The blockchain is stored in the form of chain of blocks, and all miners agree on its state through a consensus protocol. Due to its special data structure, the blockchain enables *decentralization*, *transaction immutability*, *traceability*, and *transparency*. Briefly speaking, the blockchain technique is a distributed and oftentimes public digital ledger, which consists of records called blocks that is used to record transactions across miners [41,42,43]. On the other hand, along with the continuous combination of blockchain technology and other industry applications, it is gradually divided into two major categories: permissionless blockchain and permissioned

blockchain. In short, the permissionless blockchain means that nodes are free to join and leave. The system is “loosely” organized by the asynchronous consensus protocols, which is suitable for applications, such as the cryptocurrency of Bitcoin and Ethereum [44]. In contrast, permissioned blockchains have a strict permission policy for participates, and is often used within alliances (e.g., financial service, supply chain, energy management, etc.) of different organizations/institutions that have a certain trust base.

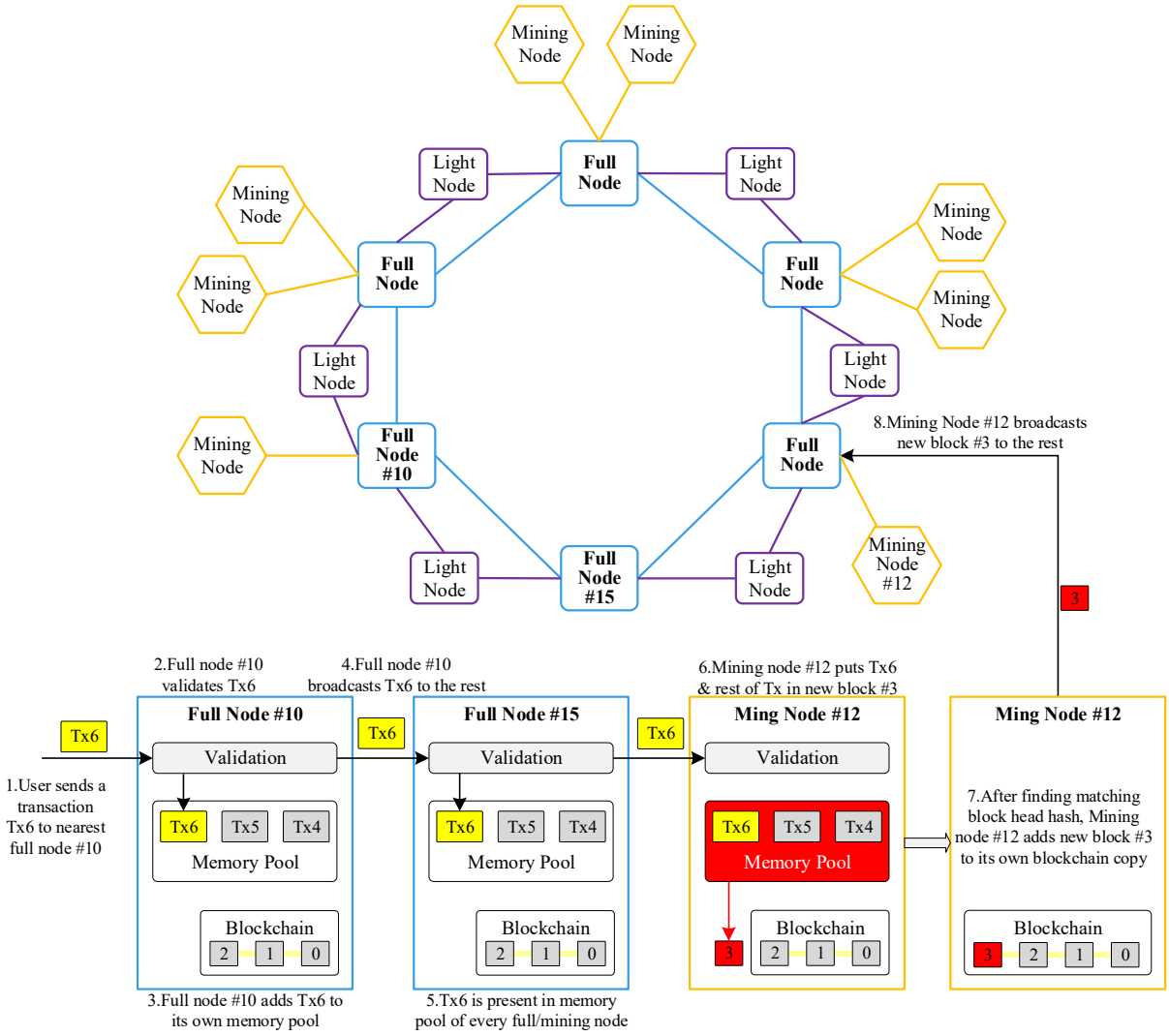


Figure 2.7: The blockchain network composition and transaction process.

Along with the development of blockchain technology, the structure of blockchain networks has become more and more complex. Generally, as shown in Figure 2.7, a representative blockchain network is composed of several types of nodes, such as full nodes, mining nodes, and light nodes. Specially, full nodes validate each transaction and block on the blockchain network by confirming the accuracy of the proof of work or proof of stake. If the block is verified, it is accepted and added to the blockchain. If not, the

block is rejected. Full nodes also download and store a copy of the entire blockchain from its origin to the present. Having multiple copies of the blockchain ensures that no one can influence the network. Mining nodes, also called block-producing nodes, validate each transaction and produce new blocks based on the verified transactions. Moreover, mining nodes achieve this objective through solving complex mathematical problems, and this process is called proof-of-work or proof-of-stake. Light nodes are similar to full nodes in function. However, rather than storing the entire blockchain in their memory, they store only the part of blockchain that is associated with the transaction being carried out. This implies that light nodes are more efficient and user-friendly than full nodes since they are cheaper to maintain and provide an easier way to access the blockchain without having to maintain a fully synchronized copy of the blockchain.

In order to understand better, as shown in Figure 2.7, the processing of a specific transaction in the blockchain network is used to illustrate the functions of the different nodes. First of all, a user sends a transaction Tx6 to nearest full node with number #10. Next, full node #10 validates the transaction Tx6. If it's correct, it adds Tx6 to its own memory pool, and then broadcasts Tx6 to the rest full and mining nodes. When transaction Tx6 is received by full node #15, it does the same work as full node #10, which results in transaction Tx6 is present in memory pool of full node #15. Unlike full nodes, when transaction Tx6 reaches mining node #12, it firstly validates Tx6 and then puts Tx6 and rest of transactions in its memory pool in new block #3. After finding matching block head hash, mining node #12 adds new block #3 to its own blockchain copy. Next, mining node #12 broadcasts new block #3 to the rest. The rest will repeat the process validate, check, add, and broadcast.

2.2.1 Blockchain Architecture

From Bitcoin, which was the first to apply blockchain technology, to Ethereum, which was the first to introduce smart contracts in blockchain, to Fisco Bcos and Hyperledger Fabric, which are the widely used permissioned blockchain platforms, they have many commonalities in the architecture, despite their different implementations. As shown in Figure 2.8, the blockchain architecture can be divided into network layer, consensus layer, data layer, smart contract layer, and application layer.

(1) Network layer

P2P-based blockchain can realize financial applications of digital asset trading. There is no central node in the blockchain network, and any two nodes can directly conduct transactions with each other, and each node is free to join or leave the network at any moment. Blockchain network nodes are equal, autonomous and distributed, and all nodes are connected to each other in a flat topology without any centralized authority nodes or hierarchical structure, and each node has the capabilities of route discovery, broadcast transactions, broadcast blocks, discover new nodes, etc. In more detail, the P2P protocols of blockchain networks are mainly used to transfer transaction data and blocks between nodes. The P2P protocols of Bitcoin and Ethereum are realized based on TCP protocol,

while the P2P protocol of Hyperledger Fabric and Fisco Bcos are implemented based on HTTP/2 protocol. In a blockchain network, nodes always listen to the data broadcasted in the network, and when receiving new transactions and new blocks from neighborhood nodes, they first verify whether these transactions and blocks are valid, including digital signatures in transactions, proof of work in blocks, etc. Only transactions and blocks that pass the verification are processed (new transactions are added to the blocks being built, and new blocks are linked to the blockchain) and forwarded to prevent the continued propagation of invalid data.

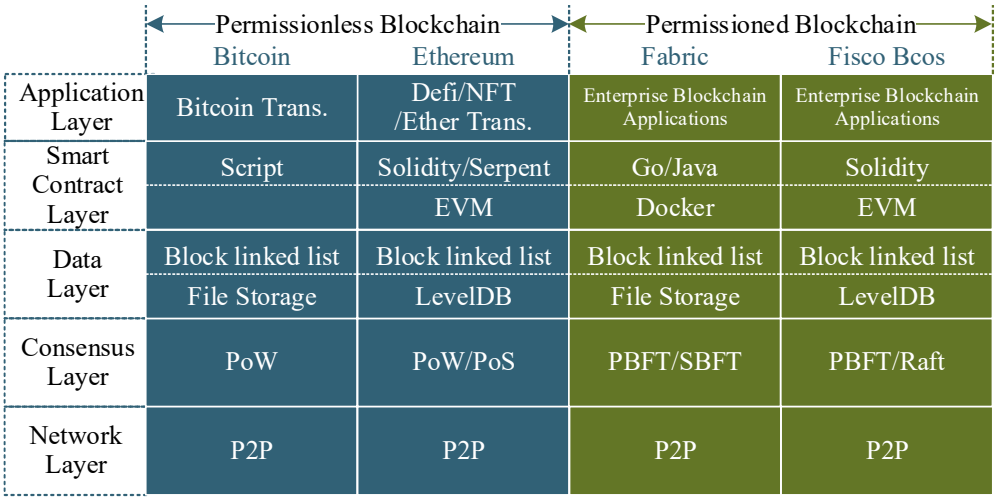


Figure 2.8: The architecture of blockchain.

(2) Consensus layer

Distributed databases mainly use Paxos and Raft algorithms to solve the distributed consistency problem. These databases are managed and maintained by a single organization, all nodes are trusted, and the algorithm only needs to support crash fault-tolerant (CFT). A decentralized blockchain is jointly managed and maintained by multiple parties, and its network nodes can be provided by any party, some of which may not be trusted, thus requiring support for the more complex Byzantine Fault Tolerant (BFT). Assuming that the network contains at most f untrustworthy nodes in a total of n nodes, the Byzantine Generals Problem can be solved for $n \geq 3f + 1$ for a network with synchronous communication and reliability. In contrast, in case of asynchronous communication, Fischer, Lynch and Paterson et al. show that the deterministic consensus protocol cannot tolerate any node failure [45]. Castro and Liskov proposed PBFT (Practical Byzantine Fault Tolerance), which reduces the complexity of Byzantine protocols from exponential to polynomial level, making it possible to apply Byzantine protocols in distributed systems [46].

In order to solve the problem of sybil attack [47], Bitcoin applies PoW (Proof of Work) consensus protocol, which is derived from Dwork's work used to prevent spam [40]. Particularly, Bitcoin requires that only nodes that have completed a certain amount of

computational work and provided proofs can generate blocks. Each network node uses its own computational resources to perform hashing operations to compete for the right to generate new block and earn Bitcoin reward, and as long as the computational resources controlled by trusted nodes across the network are higher than 51%, the entire network can be proven to be secure. In order to avoid the power consumption caused by the high reliance on node computing power, researchers propose some mechanisms that can reach consensus without relying on computing power. Peercoin applies a PoS (Proof of Stake) protocol in which the difficulty of block generation is inversely proportional to the node's stake [48]; Bitshares applies a DPoS (Delegated Proof of Stake) protocol in which the representatives with the highest number of votes from shareholders take turns to generate blocks in a given time period [49]. Hyperledger Sawtooth applied PoET (Proof of Elapsed Time) protocol based on Intel SGX trusted hardware. In conclusion, proof-based consensuses are usually applied to permissionless blockchains with free access to nodes, such as Bitcoin and Ethereum using the PoW mechanism, while voting-based consensuses are usually applied to permissioned blockchains where nodes are authorized to join, such as Hyperledger Fabric and Fisco Bcos using the PBFT algorithm.

(3) Data layer

Each block in the blockchain contains two parts: the block body stores bulk transaction data, and the block header stores Merkle root, previous block hash, timestamp and other data. Specially, the Merkle root, which is generated based on the hash of the transaction data within the block, achieves tamper-evidence and SPV (Simple Payment Verification) abilities; the pre-block hash, which is generated based on the content of the previous block, links the isolated blocks together to form the chain of blocks; and the timestamp indicates the time when the block was generated. Besides, Bitcoin's block header also contains data such as difficulty target and Nonce variables to support the mining operations of the PoW consensus protocol.

In terms of data model design, Bitcoin adopts a transaction-based data model, where each transaction consists of an input address indicating the source of the transaction and an output address indicating the destination of the transaction, and all transactions are linked together through the input and output, this mechanism makes each transaction traceable; Ethereum, Hyperledger Fabric, and Fisco Bcos need to support feature-rich general-purpose applications, so they adopt an account-based data model to support the current balance or status of the account can be queried quickly.

In terms of data storage design, as blockchain data is similar to the pre-written logs of traditional databases, it is usually stored in log file format; as the system requires a lot of hash-based key-value retrieval (e.g., transaction data retrieval based on transaction hash, block data retrieval based on block hash), index data and state data are usually stored in key-value databases, such as Bitcoin, Ethereum, Hyperledger Fabric, and Fisco Bcos all use LevelDB database to store index data.

(4) Smart contract layer

Smart contract is a digital agreement that uses algorithms and programs to write contractual terms, which are always deployed on a blockchain and can be automatically executed according to terms. The idea of smart contract was firstly introduced by Nick Szabo in 1994 [50], it was initially defined as a set of digitally defined commitments, with the intention of embedding smart contracts into physical entities to create a variety of flexible and controllable smart assets. However, due to the lack of application scenarios, smart contracts did not receive much attention from researchers until they were redefined with the advent of blockchain technology. Briefly, blockchain enables decentralized storage and smart contracts enable decentralized computation on its basis.

Bitcoin script, a set of instructions embedded in Bitcoin transactions, can only be considered a prototype of smart contract due to its single instruction type and limited functionality. Ethereum provides the Turing-complete scripting languages (such as Solidity and Serpent) and the sandbox environment EVM (Ethereum Virtual Machine) [51] for users to write and run smart contracts. The smart contract for Hyperledger Fabric, called Chaincode [52], uses the docker container as the sandbox environment with a set of signed base disk images and SDKs for Go and Java to run Chaincode written in them.

(5) Application layer

The differences between permissionless and permissioned blockchain platforms in the application layer are relatively large. Concretely, Bitcoin transactions are the primary application on the Bitcoin network. In addition to Ether transactions, the Ethereum also supports a lot of decentralized applications (such as DeFi and NFT), which are a web front-end application built by JavaScript that communicates with smart contracts running on Ethereum nodes via JSON-RPC. Hyperledger Fabric and Fisco Bcos are mainly for enterprise-level blockchain applications and does not provide cryptocurrency. Their applications can be built based on SDKs in Go, Java, Python, Node.js and other languages, and communicate with smart contracts running on Hyperledger Fabric nodes or Fisco Bcos nodes through gPRC or REST.

2.2.2 Permissionless Blockchain v.s. Permissioned Blockchain

What initially developed was the permissionless blockchain, also known as public blockchain, because anyone has the ability to access on-chain data and there is no central authority to control them. Bitcoin and Ethereum are well-known examples of this type blockchains. As more and more use cases are developed, industries consider blockchain as an important means for the industry to evolve again. However, the totally public nature of permissionless blockchain hinders its large-scale adoption, as companies cannot afford the dire consequences of opening up on-chain information to everyone. To sort this issue, permissioned blockchain came into the picture. It's also known as consortium blockchain, allow consortium blockchain management committees to limit the participation of blockchain members. In addition, the level of right to access on-chain data should be predefined in permissioned blockchains. Hyperledger Fabric and Fisco Bcos are the typical examples of permissioned blockchain.

As described at the beginning of this section, there are significant differences in participant access policies and application scenarios between permissionless blockchain and permissioned blockchain. Thus, their detailed differences and the reasons for why we choose permissioned blockchain will be discussed below.

Permissionless blockchain is a completely distributed blockchain. As the name implies, anyone can participate in the permissionless blockchain with no conditions or permissions. All members have the authority to carry out all actions, including adding and verifying transactions, smart contracts, etc. Moreover, any user has the right to decide the size and shape of such type of blockchain.

However, permissioned blockchain is a consortium blockchain established by a consortium, organization or institution that allows a few specific participants to limit the participation of other members. As a result, autonomy is transferred from one person in charge to multiple people in charge. Only those members who are predefined have the right to perform certain actions at different levels. This means that a number of participants have the privilege to make transactions, while some have the right to validate them.

Table 2.2: The detailed comparison of permissionless and permissioned blockchains.

Compare Items	Permissionless Blockchain	Permissioned Blockchain
Access restriction	No	Yes
Participant	Free access for anyone	Authorized users
Reader/Writer	Anyone	Authorized users
Consensus Protocol	PoW/PoS/DPoS	PBFT/Raft/SBFT
Consensus Determination	Mining nodes	Selected set of nodes
Consensus Process	Permissionless	Needs Permission
Incentive Mechanism	Required	Optional
Centralization	No	Partial
Transactions per Second	Hundreds	Tens of thousands
Efficiency	Low	High
Immutability Level	Impossible to tamper	Hard to be tampered
Affiliation	No	Consortium
Typical Applications	Cryptocurrencies	Traditional industries

As shown in the Table 2.2, the detailed comparison between these two blockchain systems is provided. Obviously, the permissioned blockchain is a better match for what we do, so we leverage a consortium blockchain in our work, which is required by the regulation to limit the exposure of sensitive financial data to public. Our confidential distributed ledger system consists of a group of financial institutions who are equally partnered with each other in the business of an online syndicated lending. We propose solutions to support smart contract execution over sensitive financial data in a privacy-preserving manner.

2.2.3 The relationship between Blockchain and Distributed Ledger

In reality, people are usually confused about the concepts of both blockchain and distributed ledger technologies (DLT). Therefore, we will give a brief introduction to their concepts and nature. To be honest, a distributed ledger is a database which is decentralized, that is, distributed over a number of computers or nodes. Specially, each node maintains the ledger and if any data change occurs, the ledger is updated independently at each node. Also, all the nodes are equal in status. There is no central authority to govern this database, which makes the technology transparent. Every node has the ability to update the ledger, while other nodes will validate its correctness. The update process of the intra-node ledger is relatively simple, and nodes will try to validate transactions using consensus algorithms. However, which nodes take on the responsibility of maintaining the ledger depends on the rules of that ledger. Therefore, sometimes all nodes maintain the ledger together, at other times only selected nodes are involved in maintaining the ledger.

Blockchain is actually just a type of distributed ledger. You could think of DLT as the mother technology of blockchain. However, with the explosive growth of cryptocurrencies, it has become more prevalent than distributed ledger technology. Technically speaking, blockchain is one of the distributed ledger technologies where each node maintains its own copy of the ledger. Every time new transactions are added by nodes, the update will happen on all copies of the ledger.

2.2.4 Consensus Protocol

Traditional distributed databases mainly use Paxos and Raft protocols to solve the distributed consistency problem. They assume that each node in the system is loyal and non-evil, but problems such as message loss and delay may occur. This assumption holds when all nodes of a distributed database are uniformly maintained by a single institution. In a distributed blockchain network, the nodes are provided and maintained by multiple participants who do not know and trust each other, and are driven by various interests. Therefore, Paxos and Raft protocols cannot be directly used for blockchain consensus. As the typical representative, this section introduces the PoW consensus [40] for permissionless blockchain and the PBFT consensus [53] for permissioned blockchain, and Table 2.3 shows an elaborate comparison of PoW and PBFT in terms of a series of characteristics.

Table 2.3: The comparison between PoW and PBFT.

Compare Items	PoW	PBFT
Node Access Mechanism	Anyone	Authorized users
Node Throughput (TPS)	Hundreds	Thousands
Transaction Confirmation Time	Minutes	Milliseconds
Scalability	<100000 nodes	<100 nodes
Byzantine Fault Tolerance	<50% Computing Power	<33% Votes

Compare Items	PoW	PBFT
Fork Possibility	Yes	No
Ultimate Consistency	No	Yes

(1) PoW

The identity of each user in the permissionless blockchain is anonymous and they can enter and leave freely, which makes the consensus protocol based on node voting mechanism unsuitable for such type of blockchain, because malicious attackers can create any number of nodes to increase voting rights, thus launching Sybil attack and misleading the system. PoW protocol can effectively deal with Sybil attack. It relies on the computing power competition between distributed nodes to ensure the consistency and security of distributed ledger throughout the network. PoW requires each node to solve the complicated but easy to verify SHA256 calculation problem based on its own computing power, that is, to find a suitable random number $Nonce$, so that the SHA256 hash value of the block header metadata is less than the set value of the difficulty target in the block header: $H(n \parallel h) \leq t$, where H is the SHA256 hash function, n is the random number $Nonce$, and h is the block header data, which mainly contains the previous block hash, Merkle root, etc. t is the difficulty target, the smaller the t value, the harder the n value is to find. The first node to find it gets the right of generating new block and the consensus process of PoW in the blockchain network is as follows.

- Each new transaction is broadcast to all nodes of the blockchain network.
- To construct a new block, each node collects all transactions received since the previous block was generated and computes the Merkle root of the block header based on these transactions. The random number $Nonce$ in the block header is incremented by 1 from 0 until the two SHA256 hashes in the block header are less than or equal to the set value of the difficulty target.
- Nodes all over the network participate in the calculation at the same time. If any node finds the correct random number first, it will get the right of generating the new block and the reward (the reward includes the Genesis coins in the new block and the transaction fee for each transaction), and broadcast the new block to the whole network.
- After receiving the new block, other nodes verify the validity of the transactions and the random number $Nonce$ in the block, and if it is correct, add the block to the local copy of distributed ledger and start building the next block based on that block.

As you can see, the PoW consensus protocol integrates economic incentives with the consensus process, prompting more nodes to participate in mining and maintain integrity, thus actively enhancing the reliability and security of the network, which is not available in other consensus algorithms.

(2) PBFT

The PoW consensus relies on computing power competition to guarantee the consistency and security of blockchain network, but computing power competition consumes a lot of computing resources and power energy meaninglessly, and is not suitable for the permissioned blockchain for enterprise-level applications. The PBFT consensus tolerates no more than $1/3$ of the number of nodes in the network, which means that if there are more than $2/3$ of normal nodes, on-chain data consistency and security can be guaranteed. The consensus procedures of PBFT are as follows.

- A master node is elected from the whole network nodes, and the new blocks are generated by the master node.
- Each node broadcasts the new transactions to the whole network, and the master node sorts the multiple transactions collected from the network that need to be placed in the new block into a list and broadcasts that list to the whole network.
- After each node receives the transaction list, it simulates the execution of transactions based on the sequence. After all transactions are executed, the hash digest of the new block is calculated based on the transaction results and broadcasted it to the whole network.
- If a node receives $2f$ (f is the number of tolerable malicious nodes) digests from other nodes that are all the same as its own, it broadcasts a commit message to the whole network.
- If a node receives $2f + 1$ commit messages, it can formally submit the new block and its transactions to the local copy of ledger and state database. At this point, the permissioned blockchain network has completed a round of PBFT consensus. At the beginning of the next round, the master node will be re-elected randomly.

2.2.5 Smart Contract

Smart contracts are business contracts written in a programming language which can automatically enforce the terms of the contract when predefined conditions are fulfilled, realizing the goal of “code as law”. The decentralization of blockchain makes it possible for smart contracts to run simultaneously on all nodes of the network without the participation of a central administrator, and no institution or individual can force them to stop.

The Ethereum is one of the most popular permissionless blockchain platforms [54]. There exist two types of accounts in the Ethereum network: externally owned accounts and smart contract accounts. An externally owned account is associated with a unique public-private key pair, owned by users who has an Ether balance, and the private key can sign transactions from that account. Contract accounts do not have key pairs. A smart contract

account maintains an Ether balance and stores a contract code that determines the flow of Ether in the account. A smart contract account must be activated by an externally owned account. In order to execute a smart contract used to transfer cryptocurrencies, a user-controlled account must pay a certain amount of gas using Ether. The gas fee is used to encourage miners to incorporate the code execution of the smart contract into the blockchain. Gas can be seen as an indicator of the cost of a regulated smart contract, with each assembly operation having a fixed gas cost based on its expected execution time. However, as more and more Ethereum-enabled applications become available, transaction activities become more frequent and active, leading to ever higher transaction gas fees and lower transaction consensus efficiency.

Hyperledger Fabric [55] is another type of blockchain, the permissioned blockchain, it is designed for use in an enterprise context and offers some key differentiating features compared to the permissionless blockchain. One of the most important different factors is the support for pluggable consensus protocols, allowing the platform to be more effectively customized to fit specific use cases and trust models. Additionally, Fabric can use consensus protocols that do not require native cryptocurrency to incentivize expensive mining or drive smart contract execution. Absence of cryptographic mining operations means that the platform can be deployed at roughly the same operating cost as other distributed systems, meaning that users do not have to afford high gas costs for transactions. Finally, miner nodes are mutually known and not anonymous in Fabric network, this means that while miner nodes may not fully trust each other, the network can operate under a governance model that builds on the trust that does exist between miner nodes, such as a legal agreement or a framework for handling disputes. The combination of these differentiated design features makes Fabric one of the better performing platforms today in terms of transaction processing and transaction confirmation latency.

In a nutshell, a smart contract is a self-executing computer program and consists of functions (executable units of code within the contract) and data (the state of the smart contract). The program code captures the logical contract terms between multiple parties and predefines trigger conditions and response actions. The execution of functions in a smart contract is triggered by time or events. Ideally, smart contracts can be regarded as executed by a distributed and trusted global machine that will faithfully execute every instruction. With the help of smart contracts, the rules of fair exchange can be enforced without trusted third-parties.

The mechanism of operation of smart contract is shown in Figure 2.9. After completing the smart contract code according to the business logic, it needs to be published to the blockchain network nodes. In the Ethereum, the deployed contract is stored on the blockchain and is loaded by the EVM each time when it is invoked. External applications implement various transactions by invoking smart contracts. If the invocation involves modification operations, a consensus needs to be reached across the network first, after which the modification operations are recorded in the blockchain and the modification results are stored in the state database (e.g., the transfer amount of a transfer transaction

is recorded, the increase or decrease of the account balance, etc.). If the invocation only contains a query operation, no consensus is required and it does not need to be recorded on the blockchain. Additionally, smart contracts also support registration and notification mechanisms for events within the contract, so that critical events occurring within the contract can be proactively notified to external applications. Smart contracts currently only have access to on-chain data and cannot actively listen and respond to off-chain events.

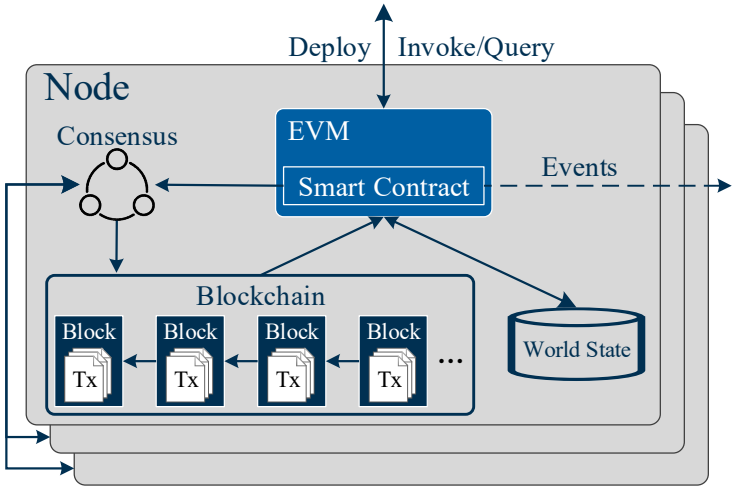


Figure 2.9: The operation mechanism of smart contract.

Furthermore, smart contracts also define the business rules for transaction logic and access to state data. External applications need to invoke smart contracts to execute transactions and access state data according to the contracts. The relationship between external applications and smart contracts is very similar to the relationship between traditional database applications and stored procedures, where stored procedures run in the database management system and access relational database data, while smart contracts run in the blockchain system and access block and state data.

2.2.6 Distributed Ledger Privacy

Bitcoin [40] realized pseudonymity by breaking the link between the public address of a user and her/his real-world identity but the transaction privacy is not considered. To further preserve the privacy, a joint transaction with multiple users can be performed to avoid the execution of their respective transactions [14,15,16,17]. As a result, individual payment is concealed.

Another line of research to protect the transaction privacy is using zero-knowledge proof, which is a protocol that a prover can prove to a verifier that a given statement is true while the prover avoids conveying any additional information other than the fact that the statement is indeed true. Maxwell proposed the Confidential Transactions scheme [18], where coins are in the form of Pedersen commitment and balance security is achieved

from the homomorphism of the commitment. In addition, range proof is used to prove that transaction value is non-negative. The scheme in [19,20] also adopted ZKP to conceal balances, the amount of the transfer, and even the transaction graph. All the mentioned prior work only considers a peer-to-peer financial model and cannot be used for the online syndicated lending, where collaboration is required among multiple financial institutions.

2.2.7 Distributed Ledger for Cryptographic Protocols

Great efforts have been devoted to improving the performance of secure multiparty computation. It is known that MPC does not guarantee fairness with no-honest-majority assumption, which hampers its applications in financial industry. To overcome the weakness, secure computation schemes with penalties are proposed. Aborting parties are forced to pay a pre-agreed penalty [28,29,30,31]. Targeting at the problem from a different angle, Choudhuri et al. in [32] achieved the standard notion of fairness based on the blockchain that was used as a public bulletin board, i.e., either all parties get the output or none does. In [21], Hawk was proposed as a compiler to build privacy-preserving smart contracts. A programmer writes a private smart contract, and Hawk compiles the program to a cryptographic protocol between the blockchain and the users. Anonymous decentralized crowdsourcing with data privacy via the blockchain were considered in [56,57]. These work are focused on their respective research objectives. Their MPC components are incompatible with our needs, such as the collection-then-distribution setting and the fund flow leakage beyond the MPC. In other words, although their MPC components guarantee the input privacy, participants have to reveal their loan contributions (the input of the MPC) to the agent in order to for the agent correctly distribute the repayment in the settlement phase. Besides, fully auditable is of the essence in online syndicated lending, while all these work do not take it into consideration.

2.3 Cryptographic Building Blocks

Let $G = \langle g \rangle$ be a cyclic group of a large prime order q. We assume that the discrete-logarithm problem is hard in G.

2.3.1 Additively-homomorphic ElGamal

ElGamal cryptosystem [58] is an asymmetric-key encryption algorithm based on the difficulty of computing discrete logarithms over finite fields. An elegant property of the original ElGamal cryptosystem is the multiplicative homomorphism. Specifically, let $E(m)$ denote the ciphertext of message m under ElGamal encryption. Given two ciphertexts $E(m_1)$ and $E(m_2)$ under the same public key, multiplicative homomorphism allows us to compute the ciphertext $E(m_1 * m_2)$ for the message $m_1 * m_2$ by multiplying $E(m_1)$ and $E(m_2)$. In our work, we make use of a variant of ElGamal cryptosystem with additive homomorphism. Assuming g_0 and g_1 are two random elements in G, additive homomorphic ElGamal cryptosystem (AH-ElGamal) consists of three algorithms as follows:

- $KeyGen(1^k)$: on the input of security parameter k, this algorithm outputs a secret key $s \in Z_q^*$ and a public key $pk = g_0^s$.

- $Enc_{pk}(m, r)$: on the input of plaintext m in the message space $[0, T]$, a random number $r \in Z_q^*$ and the public key pk, this algorithm outputs ciphertext $c = (g_1^m pk^r, g_0^r) = (g_1^m g_0^{rs}, g_0^r)$.

- $Dec_{sk}(c)$: This algorithm takes as input the ciphertext c and secret key s, and computes $g_1^m = g_1^m g_0^{rs} / (g_0^r)^s$ first. To further recover m, it suffices to compute the discrete log of g_1^m base g_1, which takes polynomial time in the size of the message space T, where T needs to be far less than q, which signifies that only short message is suitable. In practice, T is always set to be 32 bits. Note that one can also speed-up the decryption by precomputing a (polynomial-sized) table of powers of g_1 so that decryption can be done in constant time [59].

To compute the sum of m_1 and m_2, given ciphertexts $Enc_{pk}(m_1, r_1)$ and $Enc_{pk}(m_2, r_2)$, the secret key owner can get

$$Enc_{pk}(m_1 + m_2, r_1 + r_2) = Enc_{pk}(m_1, r_1) \cdot Enc_{pk}(m_2, r_2) \qquad (2.1)$$

and then decrypt it.

2.3.2 Distributed Key Generation (DKG)

Distributed Key Generation (DKG) [60] enables multiple players to jointly generate a shared group public key without any fully trusted third-party. An appealing characteristic of DKG-based cryptosystems is that it does not need to compute, reconstruct, or store the secret key in any single location. In other words, any single party in the group is prohibited from accessing the system private key. This technique is widely used to enforce threshold-based decryption policy, i.e., the message protected by the group public key can be cooperatively revealed if the number of users who intend for decryption is greater than a pre-defined threshold. Fairness is thus accomplished in this regard. In our work, we adopt a DKG protocol to complement the AH-ElGamal encryption to enable a fair decryption among multiple users.

Specially, we combine the DKG protocol [60] with AH-Elgamal to compute the sum of multiple messages while maintaining the confidentiality of individual message. The former goal is achieved from AH-Elgamal with first ciphertext aggregation then decryption manner, and DKG accomplishes the latter one since less than or equal to the pre-fixed threshold plays cannot collude to decrypt any ciphertext successful.

For discrete-log based cryptosystems, a series of players $\{P_1, P_2, \cdots, P_n\}$ with private inputs $\{x_1, x_2, \cdots, x_n\}$ engages a DKG scheme [60]. Besides a public group key g^s, each player

P_i in the group will also generate a pair of public and private individual keys (g^{s_i}, s_i), where $s = \sum_{i=1}^{n} x_i$ and $s_i \in Z_q^*$ is a share of s using (n, t) secret sharing technique [61].

2.3.3 Perdersen Commitment

A commitment scheme is a cryptographic primitive with both binding and hiding properties. Intuitively, commitment is digital equivalent of a locked box, where sender puts a value in the locked box and gives the box to receiver. Binding property ensures that once a value has been locked in the box, it cannot be changed anymore. Hiding property means that nobody can tell what is inside the box without the key. Pedersen commitment [62] is a special kind of commitment scheme with additive homomorphic, which includes three algorithms as follows:

- $Setup(1^k)$: on input security parameter k, this algorithm outputs $g, h \in G$.
- $Com(m, r)$: to commit a message $x \in Z_q^*$, sender chooses a random number $r \in Z_q^*$ and sends a commitment $com = g^m h^r$ to verifier.
- $Open(com, m, r)$: to open the commitment com, sender reveals x and r, and verifier accepts it if equation $com = g^x h^r$ holds.

In Pedersen commitment, hiding property is unconditional, binding property relies on the discrete logarithm problem. The additive homomorphic property is illustrated as follows:

- Homomorphic addition of committed message: the product of two commitments is the commitment for the sum of their committed messages.

$$Com(x_1, r_1) \cdot Com(x_2, r_2) = Com(x_1 + x_2, r_1 + r_2) \quad (2.2)$$

- Homomorphic multiplication committed value: any commitment raised to a constant $k \in Z_q^*$ is the commitment for the product of the committed message and the constant.

$$Com(x, r)^k = Com(k \cdot x, k \cdot r) \quad (2.3)$$

2.3.4 Non-interactive Zero-knowledge Proof (NIZP)

A zero-knowledge proof of protocol enables a prover $\mathcal{P}$ to convince a verifier $\mathcal{V}$ that a statement holds without revealing any extra information other than the fact [63]. In general, the protocol achieves three properties: *completeness*, *soundness* and *zero-knowledge*. NIZKP extends ZKP by removing the need of interactions between $\mathcal{P}$ and $\mathcal{V}$ via the Fiat-Shamir (FS) transform [64]. NIZKP is more desirable in cryptographic protocols to force participants to comply with protocol specification. Five types of NIZKPs are intensively used in this work:

- **Discrete logarithm proof system**. $PoK\{(y,x): y = g^x\}$ [65]: it allows $\mathcal{P}$ to prove the knowledge of the discrete logarithm x of a group element y to the base g, which includes three algorithms as follows:

  - ✓ $PROOFGEN(\cdot) \rightarrow Proof$: to generate the proof: (i) compute a commitment $t = g^v$, where v is a random number in Z_q^*; (ii) compute a response $s = v - cx$ and construct a proof (s,t), where $c = H(g,y,t)$ and $H: \{0,1\}^* \rightarrow Z_q^*$ is a secure one-way hash function.

  - ✓ $PROOFCHECK(\cdot) \rightarrow 1\ or\ 0$: if equation $g^s y^{H(g,y,t)} = t$ holds, output 1. Otherwise, 0.

  - ✓ $BATCHCHECK(\cdot) \rightarrow 1\ or\ 0$ given y_j and its proofs (s_j, t_j) for $1 \leq j \leq n$, the verifier chooses n 32-bit random numbers $\eta_1, \cdots, \eta_n$ and accepts the proofs if equation $g^{\sum_{i=1}^n \eta_i s_i} \prod_{i=1}^n y_i^{\eta_i H(g,y_i,t_i)} = \prod_{i=1}^n t_i^{\eta_i}$ holds.

- **Representation proof system**. $PoK\{(y, x_i, \cdots, x_l): y = \prod_{i=1}^l g_i^{x_i}\}$ [66]: it enables $\mathcal{P}$ to prove the knowledge of the representation $\{x_1, \cdots, x_l\}$ of an element y to the bases $g_1, \cdots, g_l \in G$, which includes three algorithms as follows:

  - ✓ $PROOFGEN(\cdot) \rightarrow Proof$: to generate this proof: (i) compute a commitment $t = \prod_{i=1}^l g_i^{v_i}$, where $v_1, \cdots, v_l$ are random numbers in Z_q^*; (ii) compute a response $s_i = v_i - cx_i$ and construct a proof $(s_1, \cdots, s_l, t)$, where $c = H(g_1, \cdots, g_l, y, t)$ is a challenge.

  - ✓ $PROOFCHECK(\cdot) \rightarrow 1\ or\ 0$: if the equation $y^c \cdot \prod_{i=1}^l g_i^{s_i} = t$ holds it outputs 1, where $c = H(g_1, \cdots, g_l, y, t)$. Otherwise 0.

  - ✓ $BATCHCHECK(\cdot) \rightarrow 1\ or\ 0$: given y_i and its proof $(s_{j,1}, \cdots, s_{j,l}, t_j)$ for $1 \leq j \leq n$, the verifier chooses n 32-bit random numbers $\eta_1, \cdots, \eta_n$ and accepts the proofs if the equation $\prod_{j=1}^n y_j^{\eta_j c_j} \cdot \prod_{i=1}^l g_i^{\sum_{j=1}^n s_{j,i}\eta_j} = \prod_{j=1}^n t_j^{\eta_j}$ holds, where $c_j = H(g_1, \cdots, g_l, y_j, t_j)$.

- **Equality proof system**. $PoK\{(y_1, y_2, x): y_1 = g_1^x \wedge y_2 = g_2^x\}$ [67]: it is used by $\mathcal{P}$ to convince $\mathcal{V}$ of the equality of the discrete logarithms of the elements y_1 and y_2 to the bases $g_1, g_2 \in G$, respectively, i.e., $log_{g_1} y_1 = log_{g_2} y_2$. It which includes three algorithms as follows:

  - ✓ $PROOFGEN(\cdot) \rightarrow Proof$: to generate a proof, the prover will (i) compute two commitments $t_1 = g_1^v, t_2 = g_2^v$, where v is a random number in Z_q^*, (ii) set $c = H(g_1, g_2, y_1, y_2, t_1, t_2)$ as the challenge and (iii) compute the responses $s = v - cx$ and construct the $Proof$ as (s, t_1, t_2).

- ✓ $PROOFCHECK(\cdot) \rightarrow 1\ or\ 0$: If both the equations $g_1^s \cdot y_1^c = t_1, g_2^s \cdot y_2^c = t_2$ hold, outputs 1; Otherwise, outputs 0, where $c = H(g_1, g_2, y_1, y_2, t_1, t_2)$.

- ✓ $BATCHCHECK(\cdot) \rightarrow 1\ or\ 0$: given $(y_{j,1}, y_{j,2})$ and their proofs $(s_j, t_{j,1}, t_{j,2})$ for $1 \leq j \leq n$, the verifier chooses n 32-bit random numbers $\eta_1, \cdots, \eta_n$ and accepts the proofs if the equations $g_1^{\sum_{j=1}^n \eta_j \cdot s_j} \cdot \prod_{j=1}^n y_{j,1}^{c_j \cdot \eta_j} = \prod_{j=1}^n t_{j,1}^{\eta_j}$ and $g_2^{\sum_{j=1}^n \eta_j \cdot s_j} \cdot \prod_{j=1}^n y_{j,2}^{c_j \cdot \eta_j} = \prod_{j=1}^n t_{j,2}^{\eta_j}$ hold, where $c_j = H(g_1, g_2, y_{j,1}, y_{j,2}, t_{j,1}, t_{j,2})$.

- **Linear equation proof system**. $PoK\{(y, a_i, b, x_i): y = \prod_{i=1}^l g_i^{x_i} \wedge \sum_{i=1}^l a_i x_i = b\}$ [68]: it enables $\mathcal{P}$ to prove that given $y = \prod_{i=1}^l g_i^{x_i}$, the linear equation $b = \sum_{i=1}^n a_i x_i$ holds, where $a_i, b \in Z_q$ are public and $g_1, \cdots, g_l \in G$. It includes three algorithms as follows:

  - ✓ $PROOFGEN(\cdot) \rightarrow Proof$: this algorithm contains two steps. (i) $\mathcal{P}$ computes a commitment $t = \prod_{i=1}^l g_i^{v_i}$, where $v_1, \cdots, v_l$ are random numbers in Z_q^* satisfying the linear relation $\sum_{i=1}^l a_i v_i = 0$; (ii) $\mathcal{P}$ computes $s_i = v_i - cx_i$ for $i = 1, \cdots, l$ and constructs a proof $(s_1, \cdots, s_j, t)$, where $c = H(g_1, \cdots, g_l, y, t)$.

  - ✓ $PROOFCHECK(\cdot) \rightarrow 1\ or\ 0$: $\mathcal{V}$ verifies the proof by checking whether $t = y^c \prod_{i=1}^l g_i^{s_i}$ and $\sum_{i=1}^l a_i s_i = -cb$ hold, where $c = H(g_1, \cdots, g_l, y, t)$. If the verification succeeds, output 1; Otherwise output 0.

  - ✓ $BATCHCHECK(\cdot) \rightarrow 1\ or\ 0$: given y_j and its proof $(s_{j,1}, \cdots, s_{j,l}, t_j)$ for $1 \leq j \leq n$, the verifier chooses n 32-bit random numbers $\eta_1, \cdots, \eta_n$ and accepts the proofs if the equations $\prod_{j=1}^n t_j^{\eta_j} = \prod_{i=1}^l g_i^{\sum_{j=1}^n s_{j,i} \eta_j} \cdot \prod_{j=1}^n y_j^{c_j \eta_j}$ and $\sum_{j=1}^n \sum_{j=1}^l a_i \cdot s_{j,i} \cdot \eta_j = -\sum_{j=1}^n c_j \cdot b \cdot \eta_j$ hold, where $c_j = H(g_1, \cdots, g_l, y_j, t_j)$.

- **Range proof system**. $PoK\{(y, x, l): y = g_1^x g_2^r \wedge x \in [0, 2^l - 1]\}$ [69]: it enables $\mathcal{P}$ to prove that a secret committed value lies in a given interval. Here $x \in Z_q$ is the committed value of l-bit and $y \in G$ is a Pedersen commitment to x using randomness r. We use the efficient Bulletproofs in our work. Specifically, the time cost of proof generation and verification is linear in l. The proof contains $2log_2(l) + 9$ group and field elements. We refer readers to [69] for the detailed discussion.

In what follows, we use $PROOFGEN$ and $PROOFCHECK$ to denote the respective algorithms for proof generation and verification in related proof systems.

2.4 Concluding Remarks

In this chapter, we provide a detailed description of the relevant works covered in our work. First of all, in order to more clearly illustrate the background and application scenarios of the research problem, we make a thorough comparison between traditional

syndicated lending and online syndicated lending in four aspects: definition, participants, process, and advantages and disadvantages, and we can see that there are significant differences between them and lenders worry about the security of their information and risk model. Furthermore, blockchain, as one of the fundamental technologies of this work, we provide a detailed introduction of blockchain from the perspectives of architecture, the difference between permissionless and permissioned blockchain platforms, the relationship between blockchain and distributed ledger, consensus protocol and smart contract. At the same time, we analyze existing distributed ledger privacy methods and the MPC-based cryptographic protocol of distributed ledger. We stress that all proposed work is not adaptable to the collection-then-distribution model of online syndicated lending. Finally, in order to make the distributed ledger confidential, regulatory and auditable, the cryptographic building blocks involved in our work are presented in detail, including additively-homomorphic ElGamal, distributed key generation, Pedersen commitment and non-interactive zero-knowledge proof.

3 Methodology

3.1 Problem Statement

Before going into technical details, we provide an overview of our work, including system model, trust assumptions, and design goals. To facilitate understanding, we here start with an introduction to the formal business process of online syndicated lending.

3.1.1 Online Syndicated Lending

As mentioned above, online syndicated lending typically offers micro-loan services in seconds through mobile apps to Internet users and the loan amount is between a few and tens of thousands of dollars. Due to the convenience, this innovative financial product has been gaining momentum and achieved unprecedented success in emerging markets. For example, Weilidai had an accumulated outstanding loan balance of US$ 14.88 billion from its launch in May 2015 to August 2017, which is comparable to the loan book of leading commercial banks [70]. Next, we briefly illustrate the formalization process of online syndicated lending.

There are four entities involved in the online syndicated lending: **borrower**, **agent bank**, **syndicate lenders** and **regulator**. An Internet user or a micro-enterprise can be the borrower to initiate a loan request to an agent bank (e.g., WeBank). On receiving the request, the agent bank first negotiates the interest rate with the borrower. Next, the agent transmits the borrower information to the syndicate lenders. In practice the agent bank can join the syndicate to become a lender as well. For ease of discussion, we only consider a pure agent that serves as a proxy to provide the lending service on behalf of the syndicate lenders in this work.

Having received the loan application, all the syndicate lenders start to activate a joint negotiation process to determine their respective capital contribution. Specifically, based on their local risk management model and the applicant information, every lender i calculates and publishes its preferred contribution proposal α_i. Then a decision can be made by comparing the total amount $loan_\alpha = \sum \alpha_i$ that the group is willing to lend collectively with the requested amount $loan_r$ by the borrower and the minimum amount $loan_{min}$ the borrower would accept (i.e., $loan_{min} \leq loan_r$). In detail, if $loan_\alpha \leq loan_{min}$, the application will be rejected as the result does not meet the expectation of the borrower; when $loan_{min} \leq loan_\alpha \leq loan_r$, the total real amount $\widehat{loan}_\alpha = loan_\alpha$ will be provided to the borrower with the actual loan share $\hat{\alpha}_i = \alpha_i \cdot \frac{\widehat{loan}_\alpha}{loan_\alpha} = \alpha_i$ for lender i; Otherwise, the borrower will get his/her requested amount (i.e., $\widehat{loan}_\alpha = loan_r$) and the actual share of each lender is $\hat{\alpha}_i = \alpha_i \cdot \frac{\widehat{loan}_\alpha}{loan_\alpha}$. To expedite the service, $\widehat{loan}_\alpha$ is usually provided by the agent bank under an agreement with all lenders.

A repayment $\hat{p}$ of the loan, for both principal and interest, is made by the borrower periodically or at one time to the agent bank. After claiming a portion $\widehat{rev}$ of the repayment as the upfront cost compensation and profit, the agent bank begins the settlement by sending the repayment allocation $\hat{p}_i = (\hat{p} - \widehat{rev}) \cdot \frac{\widehat{\alpha}_i}{\widehat{loan}_\alpha} = (\hat{p} - \widehat{rev}) \cdot \frac{\alpha_i}{loan_\alpha}$ to each lender. The repayment disbursement will carry on until the loan is paid off.

Importantly, it is mandatory for the regulator to oversee the financial service. It is not necessary to get the regulator involved in the loan processing, but expected that a posterior auditing function is available for the regulator to access the transaction details, such as the loan shares and the repayment allocation to ensure the regulation compliance.

3.1.2 Goals and Assumptions

With a clear service model in mind, we would like to discuss the expected functional and security objectives, and the threat model and assumptions used in our work.

(1) Design Goals

In this work, we aim at achieving the following four objectives.

- **Financial ledger functions.**

The key factor for the correctness of the online syndicated lending protocol is to maintain a financial ledger that all parties can refer to and take actions accordingly. For instance, the ledger should keep a transparent record of the total loan amount and individual contribution to facilitate the decision making, settlement and dispute resolution. The ledger should also be deemed a legal contractual agreement with high availability and confidence of temper-proof.

- **Privacy-preserving collaborative lending activities.**

From the privacy perspective, participating financial institutions naturally intend to fend off unnecessary information disclosure during the service. For instance, the risk management model ideally should be kept private and applied locally as it may leak sensitive financial information of its owner, such as investment strategies, debit/credit sensitivity, internal risk control policies. The inference could occur when publishing the loan contributions of lenders, by observing a series of loan request-result tuples, etc. Therefore, we need additionally make the ledger confidential and concerned information only accessible to intended parties. To this end, we are aiming to design a confidential collaborative financial ledger that is beyond the pure value transfer and enables privacy-preserving collaborative financial activities. To be specific, the capital contribution and repayment allocation of a lender associated to a special loan should be secret even to the agent.

- **Non-obtrusive auditing.**

A posterior auditing function is required by the regulation to assess the potential risks of certain activities, prevent damages to the financial systems and penalize non-compliant behavior. In this work, we explicitly allow an auditor to efficiently examine transactional details associated with lending activities to ensure the regulatory compliance such as "know your customer (KYC)", "anti-money laundering (AML)" [71], etc. The auditing process should be non-obtrusive and cost no extra burdens for the rest of the participants, including agent and lenders.

- **Performance.**

The performance of the proposed solution should be aligned with the expectations of the financial industry. We need to streamline the system by coordinating on-chain and off-chain computations and optimizing algorithm design.

(2) Threat Model and Assumptions

We adopt the blockchain as an abstraction to provide the needed ledger functions, such as liveness, safety, and smart contract execution. For implementation, we utilize a permissioned blockchain platform. So we can assume a synchronous communication environment. No extra incentives are assumed to motivate the participation, because the agent and lenders are working together to provide the loan services to customers for profits.

We leverage MPC over blockchain for secure lending service. Thus, the adversary should be clearly defined in this hybrid setting. Simultaneously, the reliability of distributed ledger is mainly determined by the used consensus mechanism to tolerate faulty nodes. From the perspective of security, these faulty nodes can be considered to be corrupted and controlled by one adversary in the worst case. In our system, a lender plays both roles of the blockchain participant and secure computation party. So it is reasonable to constrain the power of an MPC adversary to be in line with security implication of the underlying consensus protocol used in the ledger platform. This assumption is also consistent with prior research [21,28,29,30,31]. Table 3.1 lists the common consensus protocols with their fault tolerance threshold t, which also serves the upper bound of the allowed number of corrupted MPC parties. The agent bank does not take part in ledger functions and MPC process. We assume that the agent is semi-honest and interested to infer lender information from the confidential ledger and MPC, but does not have strong motivation to sabotage due to the reputation risk and harsh punishment in reality.

Table 3.1: Fault tolerance in hybrid setting of secure computation and blockchain.

Consensus algorithms	Fault / corruption tolerance t
PoW [40]	$t < n/2$
PBFT [53]	$t < n/3$
BFTRAFT [72]	$t < n/3$
Stellar [73]	$t < n/3$

We formally define the security and introduce an ideal functionality $\mathcal{F}$ that we would like to achieve in Figure 3.1. Specifically, let $F = (F_1, \cdots, F_n)$ be an n-party polynomial function, and the ideal execution of F is illustrated as $\mathcal{F}$. Let I with $|I| < t$ be a group of lenders corrupted by a non-uniform probabilistic polynomial-time adversary S. We use $Ideal_{\mathcal{F},S(Z),I}(\vec{\alpha}, k)$ to denote the output of $\mathcal{F}$ for both honest parties and S on input $\vec{\alpha}$, auxiliary input Z to S and security parameter $k \in \mathcal{N}$. Let π be an n-party protocol computing F, and $\mathcal{A}$ is a non-uniform probabilistic polynomial-time adversary in protocol π. $Real_{\pi,\mathcal{A}(Z),I}(\vec{\alpha}, k)$ denotes the output of the real execution of π for both honest parties and $\mathcal{A}$.

1. **Parameters**: Let $\vec{\alpha} = \alpha_1, \cdots, \alpha_n$ denote the prescribed input of lenders $\mathcal{B}_1, \cdots, \mathcal{B}_n$ respectively. Z is adversary S's auxiliary input, and TTP is a trusted third party.

2. **Send input to TTP:** Honest lender $\mathcal{B}_i$ sends her prescribed input α_1 to TTP; the corrupted party $\mathcal{B}_i \in I$ by S sends either her prescribed input or false input of the same length to TTP. Besides, corrupted parties are also allowed to reject this protocol by sending a special message $abort$.

3. **Early abortion:** For the input with the abort message, TTP broadcasts a message to indicate that those lenders will not participate in the protocol.

4. **Send computation result to adversary:** TTP computes $F_1, \cdots, F_n$ on input from both honest and malicious lenders. TTP sends the result of F_j to $\mathcal{B}_j$ for $j \in I$.

5. **Instruct TTP to continue or halt:** If TTP receives a "continue" instruction, it sends the result of F_i to lender $\mathcal{B}_i$ for $i \in I$. If "abort" instruction, TTP sends abort to $\mathcal{B}_i$.

6. **Output:** Honest lenders output their receipts from TTP, and the corrupted lenders output nothing. The adversary outputs any function of the prescribed input of the corrupted lenders, the auxiliary input and the result of F_j received from TTP.

Figure 3.1: Ideal functionality $\mathcal{F}$.

Definition 1. Protocol π is said to securely compute F with abort in the presence of static t-limited malicious adversaries if for every $\mathcal{A}$ in the real world, there exists an adversary S in the ideal world, such that

$$\{\text{Ideal}_{F,\mathcal{S}(z),I}(\vec{\alpha},\kappa)\}_{\kappa\in\mathcal{N}} \overset{c}{\equiv} \{\text{Real}_{\pi,\mathcal{A}(z),I}(\vec{\alpha},\kappa)\}_{\kappa\in\mathcal{N}}. \quad (3.1)$$

Remark 1. Informally, we aim to protect the confidentiality of the transactions to make sure the information leakage is aligned with the proposed MPC framework. Specifically, the agent obtains only the allowed message, i.e., daily settlement data for each lender. Here, daily settlement data represents the balance for total expenditure (contributions of a lender for all loan requests occurs in the day) and gross income (distributions of a lender for all loan repayments occurs in the same day).

3.2 Designed Building Blocks

Here we describe three fundamental algorithms that will serve as cryptographic building blocks for our work.

3.2.1 Verifiable Encryption under a Public Key

This scheme allows a user to encrypt a message under a public key with AH-ElGamal encryption algorithm and generate a proof for the validity. It is described by the following algorithms.

- ✓ $(pk, sk) \leftarrow Gen(1^{\mathcal{K}})$ takes as input a security parameter $\mathcal{K}$ and outputs a public key and a private key $sk \in Z_q^*$.

- ✓ $(c_1, c_2) \leftarrow Enc_{pk}(m, r)$ takes as input public key $pk = g_0^{sk}$, message m and random number $r \in Z_q^*$, outputs a two-part ciphertext $c_1 = g_1^m pk^r$ and $c_2 = g_0^r$.

- ✓ $P.ElGamal \leftarrow PrfGen(pk, g_0, g_1, c_1, c_2, m, r)$ takes as input public key pk, parameters g_0, g_1, two-part ciphertext c_1, c_2, message m and the random number r, calls three NIZKPs: (i) $PoK\{(c_1, m, r): c_1 = g_1^m pk^r\}$ to prove that c_1 is a representation to the bases g_1 and pk; (ii) $PoK\{(c_2, r): y = g_0^r\}$ to prove that c_2 is a representation to the base g_0; (iii) $PoK\{(y, a_i, b, x_i): y = \prod_{i=1}^{l} g_i^{x_i} \wedge \sum_{i=1}^{l} a_i x_i = b\}$ to prove that the representation of $y = c_1 \cdot c_2$ to the bases g_1, pk, g_0 satisfying the linear equation $0 \cdot m + 1 \cdot r + (q-1) \cdot r = 0 \, mod \, q$, i.e., the exponent of pk in c_1 is equal to that of g_0 for c_2. In the end, this algorithm outputs $P.ElGamal$ consisting of the generated representation proof, discrete logarithm proof and linear equation proof.

- ✓ $0/1 \leftarrow PrfCheck(P.ElGamal, pk)$ takes as input the proof, calls the $PrfCheck$ algorithms of representation proof, discrete logarithm proof and linear equation

proof systems, and outputs 1 if all the $PrfCheck$ algorithms success, otherwise 0.

Note that the AH-ElGamal is adopted due to three reasons: first, the additively homomorphic property that supports to compute the sum of different lender's loan share under ciphertext, second, small ciphertext size, and third, low computational overhead.

3.2.2 Verifiable Plaintext Consistency for Two Ciphertexts under Different keys

It enables a user to prove that two ciphertexts under two different public keys are produced from an identical plaintext that lies in a desired domain. Let $E_{pk_1}(m_1, r_1) = \left(E_{pk_1} \cdot c_1, E_{pk_1} \cdot c_2\right)$ (resp. $E_{pk_2}(m_2, r_2) = \left(E_{pk_2} \cdot c_1, E_{pk_2} \cdot c_2\right)$) denote message m_1 (resp. m_2) encrypted under public key pk_1 (resp. pk_2) with randomness r_1 (resp. r_2). The aim is to prove $m_1 = m_2 \in [0, 2^l)$. It is described by the following algorithms:

- ✓ $P.Consistency \leftarrow PrfGen(pk_1, pk_2, m_1, m_2, r_1, r_2, g_1)$ takes as input the two public keys, the two messages, the two random numbers and the system parameters. It first computes $y = \left(E_{pk_1} \cdot c_1\right) \cdot \left(E_{pk_1} \cdot c_2\right)^{-1}$ and then calls a representation proof system to state y is a representation to bases pk_1 and pk_2, i.e., $y = g_1^{m_1 - m_2} pk_1^{r_1} (pk_2)^{-r_2}$ with $m_1 - m_2 = 0$. To be specific, $PoK\{(y, r_1, -r_2): y = pk_1^{r_1} (pk_2)^{-r_2}\}$ is used and outputs a proof $P.Consistency$.

- ✓ $P.Range \leftarrow RangePrfGen\left(E_{pk_1} c_1, m_1, r_1\right)$ takes as input the first-part ciphertext, the message and the random number, calls range proof system $PoK\{(y, m_1, l): y = g_1^{m_1} pk_1^{r_1} \wedge x \in [0, 2^l - 1]\}$ and outputs a proof $P.Range$.

- ✓ $0/1 \leftarrow PrfCheck(P.Consistency, P.Range, pk_1, pk_2)$ takes as input the proof, calls the $PrfCheck$ algorithms of representation proof and range proof systems, and outputs 1 iff all the verification algorithms success, otherwise 0.

3.2.3 Verifiable Joint Decryption

It enables t or more honest users with their group key shares to jointly decrypt a ciphertext. Let $E_{pk}(M, R) = (c_1, c_2) = (g_1^M pk^R, g_0^R)$ be a ciphertext under the public key $pk = g_0^X$. There are n users jointly owning the secret key X, where each user owns a secret share x_i with the related public key $pk_i = g_0^{x_i}$ for $i \in [1, n]$.

- ✓ $y_i \leftarrow PratialDec(x_i, c_2)$ takes as input the secret share x_i and the second-part ciphertext c_2, outputs $y_i = c_2^{x_i}$.

- ✓ $P.Equlity \leftarrow PrfGen(x_i, c_2)$ takes as input the secret key share and the second-part ciphertext, calls equality proof system $PoK\{(pk_i, c_2^{x_i}, x_x): pk_i = g_0^{x_i} \wedge y_i = c_2^{x_i}\}$ and outputs the generated proof as $P.Equlity$.

- ✓ $0/1 \leftarrow PrfCheck(P.Equlity, pk_i, y_i)$ takes as input the proof, the public key pk_i and y_i, calls the $PrfCheck$ algorithms of equality proof system, outputs 1 iff it successes, otherwise 0.

- ✓ $M \leftarrow JointDec(y_{i_1}, \cdots, y_{i_t}, c_1)$ takes as input t valid y_{i_j} and the first-part ciphertext c_1, computes $c_2^X = \prod_{j=1}^{t} \left(c_2^{x_{i_j}}\right)^{\prod_{k=1,k\neq j}^{t} \frac{i_k}{i_k - i_j}} = g_0^{x \cdot R}$ and $g_1^M = c_1 \cdot (c_2^X)^{-1}$. Finally, this algorithm outputs M from a pre-computed table of the powers of g_1 since M is far less than q.

Remark 2. The above three scheme satisfies correctness, zero-knowledge and soundness under the security of the involved NIZKPs. Correctness is straightforward since anyone encrypts m under pk will generate valid proofs. Zero-knowledge and soundness are inherited from the involved NIZKPs. Specifically, zero-knowledge is achieved since for every verifier there exists a simulator with the statement can produce a well-formed transcript interaction between the honest prover and the verifier. The soundness is proved by using standard rewinding techniques that make the adversary produces two accepting conversations for different challenges with the same message [66,65,68], which can be used to solve the hard problem.

3.3 Proposed Approach

We first introduce the research questions, overview the system design and then dive into the specifics.

3.3.1 Research Questions

The motivation of using the distributed ledger is aligned with prior work [20] to maintain transparent transactional records that unambiguously indicate the relevant financial activities and the reached agreement. Therefore, the auditing can be conducted on top of the logged evidence. Blockchain also helps us build and automate the dispute resolution mechanism by smart contract function. On the flip side, the current financial ledgers [11,12,13,20] cannot satisfy the privacy needs when performing collaborative activities, such as the online syndicated lending. To this end, we employ secure multi-party computation to build a novel confidential ledger for the required frequent interactions among different financial institutions. In this work, we need to answer the following research questions.

Q1: How to design an MPC framework suitable for "collection-then-distribution" financial model?

Traditionally, MPC has been focused on the input privacy and reveals nothing except for final computation result. This is perfect for the collection decision of the online syndicated lending, i.e., hiding the individual loan share of each lender. However, the agent needs to subsequently distribute the repayment of the borrower to the syndicate as per their

corresponding loan contributions that have been "discarded" upon the end of the computation [26,27]. Due to this "one-off" feature, we need to re-feed the loan shares to another MPC instance for the repayment distribution, which has to recompute the loan contributions, thus wasting time and resources. Further, the agent can still infer the share and repayment information by executing the transfer and observing the changes of its balance. Another solution is to aggregate loan requests during a defined time period for privacy. However, longer the time, more delay added to the system, which is not acceptable for time-sensitive internet financial services (see **Q2**). As the result, the desired MPC in our framework should be "adaptable" in the sense that intermediate states in the collection phase can be efficiently reused without disclosing the initial loan shares.

Q2: How to minimize the operational cost to meet the low-latency requirement of online financial products?

Online syndicated lending, similar to other Internet financial services, needs to provide a near-instantaneous response to a user request. Therefore, any added cost should be minimal without affecting the user experience. Answering this question involving rethinking the MPC design, optimizing the on-chain and off-chain workload for blockchain, etc.

Q3: How to realize the auditing task practically?

Due to the "non-adaptability" of the traditional MPC, transactional activities cannot be recorded and presented for the auditing purpose. With the sheer amount of loan requests daily, it is not realistic to have the regulator being always online for real-time auditing and manage a large number of system secret keys associated with the requests. Therefore, it is desirable for the regulator to use its own private key for an on-demand auditing task.

Next, we will overview our system and briefly answer the above questions.

3.3.2 System Overview

We consider a group $\mathbb{B}$ of the syndicate lenders $\mathcal{B}_1, \mathcal{B}_2, \cdots, \mathcal{B}_n$ who work together to provide the loan service. Here we only consider that the agent bank $\mathcal{B}_{agent}$ serves as a middleman to help coordinate and manage the loan.

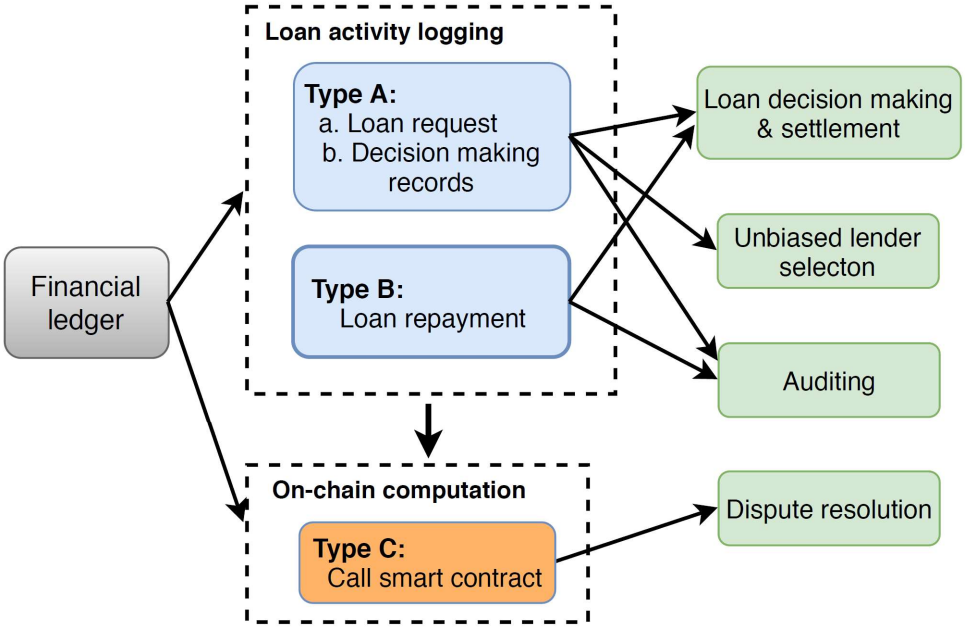


Figure 3.2: Our financial ledger.

(1) Transaction Types

There are three types of transactions in our financial ledger, illustrated in Figure 3.2. Type A transaction is for recording the loan request of the borrower, invoking an MPC instance for the decision making and logging the result. Because of the confidential nature of Type A transactions, the lenders will be treated and selected fairly by the agent to form the syndicate. The repayment information along with loan identifiers and the agent's signatures, are recorded in Type B transactions. Besides facilitating the repayment processing, the regulator can fully audit the loan by looking into both Type A (through its associated private key) and Type B transactions. Type C transaction is specifically used to call the smart contract function when a dispute appears, which could be resolved by referring to Type A and B records.

(2) Loan Request Processing

This is for the collection phase. Upon receiving a loan application from a borrower, the agent bank checks the background information of the borrower. The agent records the information along with the request details and its signature in a Type A transaction and sends it to the blockchain. This subsequently activates a joint decision-making function by performing an off-chain MPC among the syndicate lenders. The main idea of the proposed MPC is to leverage the homomorphic AH-ElGamal to compute the total loan amount and make any encrypted intermediate states reusable for the settlement in repayment distribution. We consider a malicious MPC model. As such, the lenders need to submit the ZKPs along with their encrypted loan shares. As a result, the received ciphertexts of loan shares can be verified against the lending policy without being

decrypted. Any invalid proof will be regarded as a “complaint” to invoke an on-chain dispute resolution function. On behalf of the lenders, the agent bank computes over the ciphertext and informs the borrower of the decision.

The blockchain here creates a fair play environment and serves as a trusted bulletin board (storage and broadcast channel) to record the private input, intermediate states and final output of the MPC. We further optimize the NIZKP protocol to support the batch proof verification in the presence of a large number of lenders. We carefully separate the on-chain and off-chain operations to minimize the overhead of the distributed infrastructure. The technical details will be provided in Section 3.3.3.

(3) Net Settlement between Agent and Lender

Net settlement allows each lender to accumulate debits (i.e., real loan shares) and credits (repayment allocations) with the agent throughout a business day. At the end of the business day, the totals are calculated and only the net differential needs to be transferred from the lender to the agent.

To this end, the intermediate ciphertext and computation result from the preceding decision-making process can be reused to publicly generate Pedersen commitments of the loan shares. Likewise, the commitment of the repayment allocations for each lender can be produced publicly as well. All the related computations are placed off-chain. Note that, the agent usually receives the repayments for multiple loans in one day. Therefore, a settlement is expected to be carried out over all the associated loans for each lender. In this case, the commitments for individual loans can be aggregated, which will be verified by the agent. A verification failure will trigger the on-chain dispute resolution function. See Section 3.3.3 for technical details.

Remark 3. The proposed MPC for loan collection is “adaptable” in the sense that the intermediate states are kept encrypted but reusable for repayment distribution. As a result, our solution does not sacrifice the privacy and reuses information to streamline the computation. Further, we design the system to minimize the on-chain computational burden by taking the expensive cryptographic operations off-chain and only letting blockchain for logging and “lightweight” contract execution. Therefore, our design satisfies the requirements laid out by the research questions **Q1** and **Q2**. Details will be discussed in Sections 3.3.3 and 4.

(4) Auditing

For the auditing function in the question **Q3**, each lender encrypts its preferred loan contribution under the regulator's public key. We adopt NIZKP to prove the consistency such that a lender cannot generate a valid proof unless it is compliant. The ciphertexts and their proofs are also recorded as Type A transactions in the blockchain. As a result, the regulator is able to audit the loan service at any time with its private key by accessing the distributed confidential ledger. In the end, the preferred contributions of lenders for a

particular loan request are obtained, from which both the actual loan shares and repayment distributions can be deduced. The technical details are provided in Section 3.3.4.

3.3.3 System Details

(1) System Initialization

In this phase, we initialize and prepare the system with appropriate parameters. For example, every participating institution $\mathcal{B}_i$ needs to generate a signature key pair and broadcast the public key as its identity. Then these participants initialize the intended blockchain platform in an agreement of various configurations (e.g., the consensus protocol, parameters, etc.). Suppose there are a cyclic group G with a large prime order q and two random elements $g_0 \in G$ and $g_1 \in G$. Let Z_q denote $(0,1,2,\cdots,q-1)$. For an distributed AH-ElGamal algorithm, we adopt the DKG protocol [60] to produce a group public key for all the lenders $pk_{\mathcal{B}} = g_0^x \ mod \ q$. This key is usually generated once for long-term use due to the stable composition of the syndicate in practice, but can be updated if necessary. Each lender also gets a share secret x_i of the secret key x and the corresponding public key share $pk_{\mathcal{B}_i} = g_0^{x_i} \ mod \ q$. Further, we need a secure one-way hash function $H: \{0,1\}^* \rightarrow Z_1^*$ to compute the challenge in NIZKPs. Besides, let $pk_{\mathcal{FR}}$ be the regulator's public key that will be used for the auditing. Note that the cost for this step is minimal because the parameters can be produced by individual banks rather than relying on a trusted third party or being bootstrapped from a joint MPC procedure. In addition, the cost is one-off and will be further amortized over the future use. The parameter setup is also designed to be flexible to support membership update.

(2) MPC for Joint Lending Decision

We design an MPC framework to enable a collective decision making process among the syndicate lenders for a loan request. This step contains three fundamental algorithms as shown in Figure 3.3 and we briefly introduce them below.

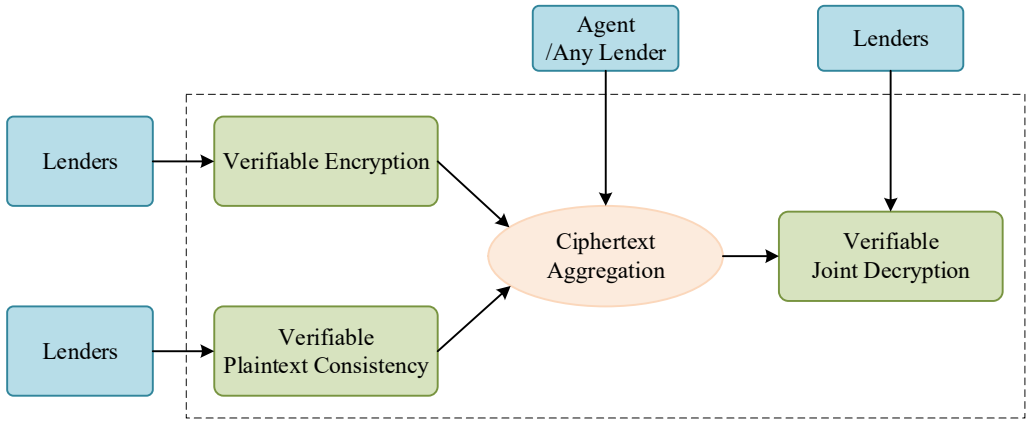


Figure 3.3: Building blocks of the proposed secure computation by multiple lenders.

- **Verifiable encryption**: It allows a user to encrypt a message under a public key and generate a proof for some specific properties (see Section 3.2.1).

- **Verifiable plaintext consistency**: It enables a user to prove that two ciphertexts under two different public keys are produced from an identical plaintext that lies in a desired domain (see Section 3.2.2).

- **Verifiable joint decryption**: It enables t or more honest users with their group key shares to jointly decrypt a ciphertext (see Section 3.2.3).

The main idea is that when the agent bank submits a Type A transaction to the blockchain, which records a new loan request with id, an off-chain MPC for the decision making will be triggered. Specifically, each lender runs verifiable encryption algorithm to encrypt its preferred contribution using two public keys respectively: the group public key of the syndicate and the regulator's public key (for the auditing purpose, see Section 3.3.4), and broadcasts its ciphertext through blockchain. Due to the additively homomorphic property of AH-ElGamal, all the encrypted contributions of lenders can be summed up to get an aggregated ciphertext. Finally, t or more honest lenders use verifiable joint decryption algorithm to decrypt this ciphertext to get the total amount the syndicate would like to lend.

The detail of the proposed MPC-enable decision making is presented in Figure 3.4. Here, we explain the used symbols and notations in the MPC. In step 2, $P.ElGamal(pk, m)$ is to prove that $E_{pk}(m)$ is indeed a ciphertext of the message m under the public key pk. In step 3, $P.Consistency(\alpha_i, pk_{\mathcal{B}}, pk_{\mathcal{FR}})$ is a proof to demonstrate that the plaintexts of $E_{pk_{\mathcal{B}}}(\alpha_i, r_{i,1})$ and $E_{pk_{\mathcal{FR}}}(\alpha_i, r_{i,2})$ are identical. We use $P.Range(\alpha_i)$ to prove that α_i lies in a desired range $[0, 2^l - 1]$. At the end of step 4, other lenders can check these proofs to verify the compliance of $\mathcal{B}_i$. All the computations are off-chain and only the computation results are properly signed and recorded on the blockchain as Type A transactions. The proofs that have been verified valid can be safely discarded to further save the on-chain space without affecting the subsequent operations. Any invalid proof will trigger dispute resolution function (Section 3.3.5). Note that the total of the proposed loan shares $loan_\alpha$ is revealed in the end. With this public value, the actual loan total $\widehat{loan}_\alpha$ is computed by both the agent and lenders. Besides, lenders can also compute their loan shares privately (see Section 3.1.1). Therefore, the agent bank is not able to discriminatively choose the participating lenders by their respective sheer amount of contribution. To expedite the service, the agent provides the $\widehat{loan}_\alpha$ to the borrower immediately on behalf of the syndicate, and $\widehat{loan}_\alpha$ will be compensated by the lenders in the settlement at the same day.

1. Each lender $\mathcal{B}_i$ runs its local risk management model to determine the preferred loan share α_i.

2. $\mathcal{B}_i$ performs the **verifiable encryption** with the input of $(\alpha_i, pk_i, r_{i,1})$ to obtain the ciphertext under the public key $pk_{\mathcal{B}}$ and its proof, i.e., $E_{pk_{\mathcal{B}}}(\alpha_i, r_{i,1})$and $P.ElGamal(pk_{\mathcal{B}}, \alpha_i)$. In addition, the **verifiable encryption** is called again with input $(\alpha_i, pk_{FR}, r_{i,2})$ to compute the ciphertext under the public key pk_{FR} and its proof, i.e., $E_{pk_{FR}}(\alpha_i, r_{i,2})$ and $P.ElGamal(pk_{FR}, \alpha_i)$.

3. $\mathcal{B}_i$ calls **verifiable plaintext consistency** algorithm to obtain $P.Consistency(\alpha_i, pk_{\mathcal{B}}, pk_{FR})$ and $P.Range(\alpha_i)$.

4. $\mathcal{B}_i$ submits the request id and the above output along with her signature to the blockchain.

5. $\mathcal{B}_i$ locally aggregates the valid ciphertexts recorded in the blockchain to get the encrypted total amount of the preferred loan contributions, i.e., $loan_\alpha = \sum \alpha_i$.

6. t or more users collectively execute the **verifiable joint decryption** algorithm derive $loan_\alpha$.

Figure 3.4: MPC-enable decision making.

(3) Daily Settlement

For ease of understanding, we re-summarize the notations used for this section in Table 3.2. Note that $Com(\alpha_i^j, r_i^j)$ and $loan_\alpha^j$ are recorded in the Type A transaction for L^j. $\widehat{loan}_\alpha^j$ is publicly known by the syndicate. Similarly, $Com(\alpha_i^k, r_i^k)$ and $loan_\alpha^k$ are also public.

Table 3.2: Notations.

Notations	Description
U	The set of identifiers for lent loans recorded in Type A transactions during a day
L^j	A loan L^j with the identifier $j \in U$
α_i^j	$\mathcal{B}_i$'s initially offered loan share for L^j

Notations	Description
$Com(\alpha_i^j, r_i^j)$	Pedersen commitment of α_i^j with randomness r_i^j
$loan_\alpha^j$	The preferred total amount for loan L^j
$\widehat{loan}_\alpha^j$	The actual total amount of L^j
V	The set of identifiers of repayments in Type B transactions throughout a business day
$\hat{p}^k$	The repayment amount of loan $L^k, k \in V$
$\widehat{rev}^k$	The reserved portion of p^k for the agent
α_i^k	The initially preferred loan share of $\mathcal{B}_i$ for loan L^k
$Com(\alpha_i^k, r_i^k)$	The commitment of $\mathcal{B}_i$'s on α_i^k
$loan_\alpha^k$	The proposed total amount for loan L^k

We design the daily settlement mechanism to mitigate the leakage of the individual loan share or repayment allocation. It includes four steps in Figure 3.5. In particular, at the end of each business day, the following operations will be executed between agent and each lender.

(1) **Publicly compute commitments of a lender's loan shares**: The agent with access to the ledger can publicly compute all the Pedersen commitments of the lender $\mathcal{B}_i$'s loan shares during the day.

(2) **Publicly compute commitments of a lender's repayment allocations**: With the blockchain record, the agent can compute all the Pedersen commitments of $\mathcal{B}_i$'s repayment allocations during the day.

(3) **Publicly compute commitments of the net differential of a lender**: With the commitments from above two steps, the agent can generate the Pedersen commitment of the actual amount that needs to be transferred from the lender $\mathcal{B}_i$'s to the agent.

(4) **Open the commitment of the net differential**: It is performed by lender $\mathcal{B}_i$'s to disclose its net differential that needs to be transferred to the agent.

1. **Publicly compute commitments of lender $\mathcal{B}_i$'s loan shares**: For all $j \in U$, compute the commitment of $\mathcal{B}_i$'s loan share $\hat{\alpha}_i^j$:

$$Com\left(\hat{\alpha}_i^j, r_i^j \cdot \frac{\widehat{loan}_\alpha^j}{loan_\alpha^j}\right) = Com\left(\alpha_i^j, r_i^j\right)^{\frac{\widehat{loan}_\alpha^j}{loan_\alpha^j}}$$

2. **Publicly compute commitments of lender $\mathcal{B}_i$'s repayment allocations:** For all $k \in V$, compute the commitment of $\mathcal{B}_i$'s repayment allocation $\hat{p}_i^k$, i.e.,

$$Com\left(\hat{p}_i^k, r_i^k \cdot \frac{\hat{p}^k - \widehat{rev}^k}{loan_\alpha^k}\right) = Com\left(\alpha_i^k, r_i^k\right)^{\frac{\hat{p}^k - \widehat{rev}^k}{loan_\alpha^k}}$$

3. **Publicly compute commitments of the net differential of lender $\mathcal{B}_i$:** Compute the total loan shares of $\mathcal{B}_i$ throughout the business day

$$Com_{out} = \prod_{j \in U} Com\left(\hat{\alpha}_i^j, r_i^j \cdot \frac{\widehat{loan}_\alpha^j}{loan_\alpha^j}\right)$$

Compute the total repayment allocations of $\mathcal{B}_i$ during the business day

$$Com_{in} = \prod_{k \in V} Com\left(\hat{p}_i^k, r_i^k \cdot \frac{\hat{p}^k - \widehat{rev}^k}{loan_\alpha^k}\right)$$

Compute the commitment of the actual payment

$$C = Com_{out} \cdot (Com_{in})^{-1}$$

4. **Open the commitment of the net differential:** $\mathcal{B}_i$ computes the net repayment

$$\sum_{j \in U} \hat{\alpha}_i^j - \sum_{k \in V} \hat{p}_i^k$$

It also computes the aggregated random number

$$\sum_{j \in U} r_i^j \cdot \frac{\widehat{loan}_\alpha^j}{loan_\alpha^j} - \sum_{k \in V} r_i^k \cdot \frac{\hat{p}^k - \widehat{rev}^k}{loan_\alpha^k}$$

Finally, $\mathcal{B}_i$ sends the results to the agent, and its validity can be checked via the commitment C.

Figure 3.5: Settlement details.

The actual loan share $\hat{\alpha}_i^j$ or repayment allocation $\hat{p}^k$ of lender $\mathcal{B}_i$ is proportional to its initial share proposal α_i^j or α_i^k for $j \in U$ and $k \in V$. Thus, by reusing the Pedersen commitment of α_i, the commitments of $\hat{\alpha}_i^j$ and $\hat{p}^k$ can be directly computed. In theory, the homomorphism of Pedersen commitment enables computing and opening an aggregated commitment of the net differential rather than disclosing individual loan share or repayment allocation. The agent only obtains the net differential from the settlement protocol in the end, mitigating the leakage of a lender's actual contribution in a specific loan. In practice, considering the sheer number of loans that need to be settled daily [74], it is challenging for the agent to deduce private information of individuals. As a result, the related concerns in the research question **Q1** have been addressed. In addition, a forged net differential will be detected. Other lenders can jointly decrypt their initial share proposals to restore the settlement function.

Dealing with decimals and negative numbers. In the computations, $\frac{\widehat{loan}_\alpha^j}{loan_\alpha^j}$ and $\frac{\hat{p}^k - \widehat{rev}^k}{loan_\alpha^k}$ may be decimals. We keep two decimal places such that loan shares/repayment allocations could be measured by cent. Besides, $\frac{\widehat{loan}_\alpha^j}{loan_\alpha^j}$ and $\frac{\hat{p}^k - \widehat{rev}^k}{loan_\alpha^k}$ are enlarged by 100 times to make them being elements in Z_q. To support negative numbers resulted from net payment calculation, we use $\left[\frac{q-1}{2}+1, q-1\right]$to denote negative numbers and let $\left[0, \frac{q-1}{2}\right]$ be non-negative.

3.3.4 Auditing

Type A transaction auditing:

- ✓ **Compute the preferred contributions:** For $1 \le i \le n$, the regulator uses its private key to decrypt the ciphertext $E_{pk_{FR}}(\alpha_i, r_{i,2})$ and obtains lender $\mathcal{B}_i$'s preferred contribution α_i.
- ✓ **Compute the actual loan shares:** Let $loan_\alpha = \sum_1^n \alpha_i$, if $loan_\alpha < loan_{min}$, the application will be rejected as the result does not meet the expectation of the borrower and each loan share $\hat{\alpha}_i = 0$; if $loan_{min} \le loan_\alpha \le loan_r$, $\hat{\alpha}_i = \alpha_i$ for lender $\mathcal{B}_i$; Otherwise, $\hat{\alpha}_i = \alpha_i \cdot \frac{loan_r}{loan_\alpha}$.

Type B transaction auditing:

- ✓ **Compute the repayment allocations:** For $1 \le i \le n$, given the recorded repayment $\hat{p}$ and the profit $\widehat{rev}$ of the agent, the regulator computes the allocation of each lender $\mathcal{B}_i$, i.e., $\hat{p}_i = (\hat{p} - \widehat{rev}) \cdot \frac{\alpha_i}{loan_\alpha}$.

Figure 3.6: Auditing details.

The auditing process is illustrated in Figure 3.6, which is achieved by letting each lender encrypt its preferred loan share under the regulator's public key. NIZKPs are used to

make sure that the lenders follow the protocol honestly. By accessing the blockchain, the regulator uses its private key to get the preferred loan share α_i for $1 \leq i \leq n$ from type A transactions, which enables the regulator to compute the loan shares and the related repayment allocations in the form of plaintexts.

3.3.5 Dispute Resolution

Disputes appear if there exists inconsistency among the views of participants throughout the life cycle of a loan, which are faithfully recorded by the blockchain. For example, a malicious lender submits a valid signature of a message to the blockchain but attaches an invalid NIZK proof, or it produces a false repayment amount that does not satisfy the aggregated commitment for the settlement. Then any other participants can generate a Type C transaction to file a complaint that calls the smart contract function for dispute resolution. Figure 3.7 shows an example of the deployed smart contract to identify a lender who submits an invalid proof for its initial loan share proposal. Once the misbehavior is verified, the dishonest party will be punished accordingly.

3.3.6 Batch Verification for Proofs

NIZKPs are intensively used in our MPC-based loan request processing. It is worth noting that all the NIZKPs support batch verification [75] to significantly reduce the number of expensive exponentiation operations. The main idea is that verifying $g^x = 1 \wedge g^y = 1$ can be replaced by selecting two random scalars a, b from a sufficiently large domain to efficiently check $g^{ax+by} = 1$. Besides, considering the fact that most generators are fixed in the public parameters, fast fixed-base exponentiation with pre-computation [76] can be adopted as well to speed up all the multi-exponentiation. The detailed batch verification algorithms for the NIZKPs are listed in Subsection 2.3.4.

```
contract Cdl is ParallelContract {

// Contract events.
event VerifyPass(string transactionInfo,
                 uint badLenderId);
event VerifyFail(string transactionInfo,
                 uint badLenderId);

constructor() public {
  // Loads WeDPR precompiled tool contract from
  // address 0x5018 of  blockchain VM.
  wedpr = WedprPrecompiled(0x5018);
  registerParallelFunction(
      "verifyCdlProof(string)", 0);
}

// Verifies the validity of a CDL proof.
function verifyCdlProof(
    string transactionHeader
) public {
  string memory cdlProof=
      wedpr.getCdlProof(transactionHeader);
  string memory encryptedLoan =
      wedpr.getEncryptedLoan(transactionHeader);
  string memory transactionInfo =
      wedpr.getTransactionInfo(transactionHeader);
  // Returns 0 if the verification passed;
  // Otherwise returns  the ID of the lender
  // who caused the failure.
  uint badLenderId = wedpr.verifyCdlProof(
      encryptedLoan, cdlProof);
  if (badLenderId == 0) {
    // Stores the encrypted transaction.
    wedpr.storeEncryptedLoan(encryptedLoan);
    emit VerifyPass(transactionInfo, 0);
  } else {
    emit VerifyFail(transactionInfo, badLenderId);
  }
}

}  // contract Cdl
```

Figure 3.7: Auditing details.

3.4 Security and Privacy Analysis

Before providing the formal security proof, we first describe the sequential composition security [77] and GMW compiler [78]. The former guarantees that security holds for protocols that call functionalities in a sequential manner, and the latter takes as input any protocol security under semi-honest adversaries and outputs a new protocol for the same functionality that is secure in presence of malicious adversaries.

3.4.1 Sequential Composition Security

We use the hybrid model [77] to define the security. In the hybrid model, n parties run a protocol π to evaluate function g with $ideal\ calls$ to a trusted party (as in the ideal world) computing n-party functions $f_1, f_2, \cdots, f_m$. Here, we consider that each $ideal\ call$ for a

functionality f_i is performed sequentially. That is, all parties send their f_i-input to the trusted party and wait until the trusted party sends the output back. In addition, ideal calls to the trusted party can be done many times, even for the same function, but each call is independent. In the real protocol, $ideal\ calls$ are substituted by secure protocols $\rho_1, \rho_2, \cdots, \rho_m$ with the functions $f_1, f_2, \cdots, f_m$.

Sequential composition security states that if a protocol is secure in the stand-alone model under definition X, then it remains secure under X under *sequential composition*. Here, X denotes the security model, e.g., semi-honest model or malicious model. *Sequential composition* indicates that protocols are composited as long as the executions are run sequentially, i.e., each execution concludes before the next begins.

3.4.2 GMW Compiler

Let π be a semi-trusted protocol. The main idea of the GMW compiler is that each party proves that she executes a protocol π on the consistent inputs with honest execution of π. Specifically, commitments to the inputs are used to prevent parties from running π honestly, but with different inputs in different rounds in the GMW compiler. Besides, ZK-Proof combined with coin tossing technique is utilized to guarantee that each party's random tape for ZK-Proof is chosen uniformly. As a result, the malicious adversary either follows π correctly or cheats with an invalid proof.

3.4.3 Security Proof

Theorem 4. DKG protocol used in our system is secure under the t-limited malicious adversary model.

We refer readers to [60] for details. With the sequential composition security, DKG algorithm can be used securely instead of ideal call to a functionality for a distributed AH-ElGamal cryptographic system generation provided by a fully-trusted third-party.

Theorem 5. MPC for joint decision making in our system is secure under t-limited malicious adversary model.

Algorithm 1: Semi-trusted protocol π of our MPC

Input: Each lender $\mathcal{B}_i$'s preferred loan contribution α_i, secret share x_i, and lender group public key $pk_{\mathcal{B}}$

Output: $\alpha = \sum_{i=1}^{n}$

1 Each lender $\mathcal{B}_i$ computes and broadcasts its ciphertext $E_{pk}(\alpha_i, r_i)$, where r_i is a random in Z_q^*;

2 All lenders aggregate the ciphertexts $E_{pk}(\sum_{i=1}^{n}\alpha_i, \sum_{i=1}^{n} r_i) = (c_1, c_2) = (g_1^{\alpha} pk^r, g_0^r)$, where $\alpha = \sum_{i=1}^{n}\alpha_i$ and $r = \sum_{i=1}^{n} r_i$;

3 Each lender $\mathcal{B}_i$ decrypts the aggregated ciphertext by sk_i and broadcasts $c_2^{sk_i} = g_0^r x_i$;

4 Each lender $\mathcal{B}_i$ uses Lagrange interpolation to compute c_2^x and recover α from $g_1^{\alpha} = c_1 \cdot (c_2^x)^{-1}$;

We first give the semi-trusted protocol π of our MPC as shown in Algorithm 1 and then demonstrate its security. The auditing can be easily achieved in π by letting each lender encrypt α_i under the regulator's public key, which does not break the security. And we omit the regulation functionality for simpleness since it is achieved by the AH-ElGamal encryption that is secure under the adaptively chosen plaintext attack. Due to the GMW compiler, the security of our MPC under t-limited malicious adversary model is deduced. The security of the semi-trusted protocol under t-limited semi-trusted adversary model is simple, and we give a brief analysis here.

The ideal protocol of π is as follows. Each lender $\mathcal{B}_i$ submits α_i to the fully-trusted server, who then computes $\alpha = \sum_{i=1}^{n} \alpha_i$ and sends it back to all parties. Let Adv denote the adversary who corrupts $t-1$ lenders $\mathcal{B}_1, \cdots, \mathcal{B}_{t-1}$. The simulator $\mathcal{S}$ (i.e., ideal adversary in the ideal protocol) owns the outputs α, the private inputs α_i and their shares sk_i for $i \in [1, t-1]$. In order to simulate $\mathcal{B}_t, \cdots, \mathcal{B}_n$ in the real protocol, $\mathcal{S}$ sets $\alpha_t, \cdots, \alpha_{n-1}$ as random numbers and computes $\alpha_{n-1} = \alpha - \sum_{i=1}^{n-1} \alpha_i$, so that $\alpha = \sum_{i=1}^{n} \alpha_i$ still holds. Then $\mathcal{S}$ does the follows.

Step-1: Simulation for encryption. For $t \le i \le n$ encrypts α_i under pk and broadcasts the ciphertext $E_{pk}(g_1^{\alpha}, r_i) = \left(g_1^{\alpha_i} pk^{r_i}, g_0^{i}\right)$ to Adv, where r_i is a random number in Z_q^*. This step is indistinguishable from the real protocol due to the randomness of r_i.

Step-2: Ciphertext aggregation does not need simulation since no message is needed to send. $\mathcal{S}$ receives $E_{pk}\left(g_1^{\alpha_i}, r_i\right) = \left(g_1^{\alpha_i} pk^{r_i}, g_0^{i}\right)$ for $1 \le i \le t-1$. Besides, both $\mathcal{S}$ and Adv compute the aggregated ciphertext $E_{pk}(\sum_{i=1}^{n} \alpha_i, \sum_{i=1}^{n} r_i)$, i.e., $(c_1, c_2) = (g_1^{\alpha} pk^{r}, g_0^{r})$ locally.

Step-3: Simulation for decryption. $\mathcal{S}$ first computes $c_2^{x} = pk^{r} = c_1 \cdot (g_1^{\alpha})^{-1}$ and $c_2^{x_i}$ for $1 \le i \le t-1$. Then, $\mathcal{S}$ is able to compute valid $c_2^{x_t}, \cdots, c_2^{x_n}$ by utilizing the Lagrange interpolation. Finally, $\mathcal{S}$ sends $c_2^{x_t}, \cdots, c_2^{x_n}$ to Adv. This step is also indistinguishable from the real protocol because of the validity of $c_2^{x_i}$ for $t \le i \le n$.

Now we observe that π is secure under the semi-trusted adversary model. Our MPC for joint decision can be regarded as the output of the GMW compiler taken as input protocol π, where each transferred message is committed and attached with NIZK proofs. In the end, we conclude the security of our MPC under the malicious adversary model.

Theorem 6. The net settlement solution is secure under t-limited malicious adversary model.

The daily settlement achieved in the manner of multiple commitment aggregation then open can be regarded as a special kind of MPC. To be specific, the inputs of this MPC include public commitments that will be aggregated and $\mathcal{B}_i$'s the value opened from the aggregated commitment. The public commitments are generated from the meta-commitments (i.e., commitments of the preferred contributions) and public information

recorded in the blockchain. The output is only available to the agent bank, i.e., $\mathcal{B}_i$'s net differential.

This MPC should satisfy two requirements: (i) public commitment hides the secret value, i.e., the intermediate states of the MPC do not leak any information; (ii) the net differential cannot be forged, i.e., the output the MPC is correct.

We suppose that meta-commitments are generated from ideal calls to a third party in the hybrid model. The requirements are fulfilled by the used Pedersen commitment with perfect-binding and computation-hiding. In the real protocol, meta-commitments are provided by the MPC for the joint decision and stored in the blockchain. Both the MPC and blockchain is t-limited secure. With the sequential composition security, the solution to net settlement is secure under t-limited malicious adversary model.

3.5 Concluding Remarks

In this chapter, in order to portray the research problem more clearly, we first describe the research problem in detail through the perspectives of problem formalization, research objectives, threat model and assumptions. Next, the designed cryptographic algorithms used in our system are described in detail. After that, we provide a thorough description of the scheme of the proposed system, which mainly contains modules and phases such as transaction types, MPC for joint lending decision, daily settlement, auditing, dispute resolution, and batch verification for proofs. Finally, the capabilities of security and privacy of the proposed system are analyzed and proved focusing on the perspectives of sequential composition security, GMW compiler and security proof.

4 Performance Evaluation

We develop a prototype system of the proposed scheme running on top of a leading consortium blockchain, an open-source and non-profit consortium blockchain platform, Fisco Bcos [34], which is a reliable, secure, efficient and portable blockchain platform with proven success from many partners and successful financial-grade applications. It has established research projects in areas including credit, equity, loyalty points system, insurance, commercial bills, cloud service, digital assets, and wealth management issuance and trading and has been widely adopted by many financial applications including inter-institutional reconciliation, supply chain finance, etc. We report the performance of the prototype system in this section.

4.1 Setup

We consider a real-world loan scenario where the number of lenders may vary from 10 to 100. All the lenders are deployed on Tencent Cloud and each lender node machine runs CentOS Linux release 7.2.1511 with 8-core Inter(R) Xeon(R) CPU E5-2600 V4, 16GB memory, and 8Gpbs bandwidth. PBFT [53] is adopted as the distributed consensus protocol. We use Rust and C++ with curve25519-dalek library [79] for the used cryptographic primitives while ECDSA [80] is used as the signature algorithm with curve secp256k1 [81] due to its efficiency. Both curves provide 128-bit security guarantee. The result is averaged on 10 rounds of tests.

4.2 Transaction Size

Table 4.1: Size of Type A and B transactions.

Number of lenders	Type A size	Type B size
10	2.96 KB	96 B
30	8.68 KB	
50	14.4 KB	
70	20.1 KB	
100	28.7 KB	

Table 4.1 shows the transaction size. One Type A transaction stores one loan request, associated decision record and signatures. We discard the proof that has been verified valid to reduce the transaction size. Each lender contributes two ciphertexts (i.e., under the lender group public key and the regulator public key) and a partial ciphertext for joint decryption, which are attached with two signatures for authenticity. In the experiment, we set the size of a request to be 32 bytes. Curve secp256k1 [81] results in 64-byte ECDSA signatures and a point in curve25519 [79] is 33 bytes with compression. Thus, the size of a Type A transaction is $32 + 64 + (33 \cdot 5 + 64 \cdot 2) \cdot n = 96 + 293 \cdot n$, where n denotes the number of lenders.

Each Type B transaction is generated for a repayment to a specific loan for all the lenders, including 32-byte repayment information and the agent's signature. Its size is constant, i.e., 96 bytes. Suppose there is a syndicate of 10 lenders to process average 200,000 loan requests and 200,000 repayments per day. The accumulated size of both Type A and Type B transactions is about 0.6GB, which is common for financial industry and not a burden due to the cheap storage.

Type C transaction is used to trigger smart contract to identify misbehavior and resolve dispute that occurs infrequently in reality. The size mainly depends on the type of proof involved in the complaint. For instance, discrete logarithm proof includes two points in curve25519 [79] and thus the transaction size is about 66 bytes.

4.3 Transaction Throughput

We measure the number of transactions per second (*TPS*) to show the throughput of the proposed confidential ledger and compare it with practical demand. Figure 4.1 exhibits the throughput for Type A transactions, i.e., the capacity of collaboratively processing loan requests by a group of syndicate lenders. The throughput is determined by the used PBFT consensus and cryptographic operations in the MPC for joint lending decision making. It is expected that the MPC with the batch proof verification outperforms the individual proof checking. With 50 lenders, batch verification shows 2.3 times throughput increase. Compared to the commercial need to process 200,000 loan requests in a day, which approximates to $2.31 = 200{,}000/(24 \cdot 3600)$ *TPS* on average, our system throughput can be 31.6 times greater (73 *TPS*) with a large syndicate of 100 lenders.

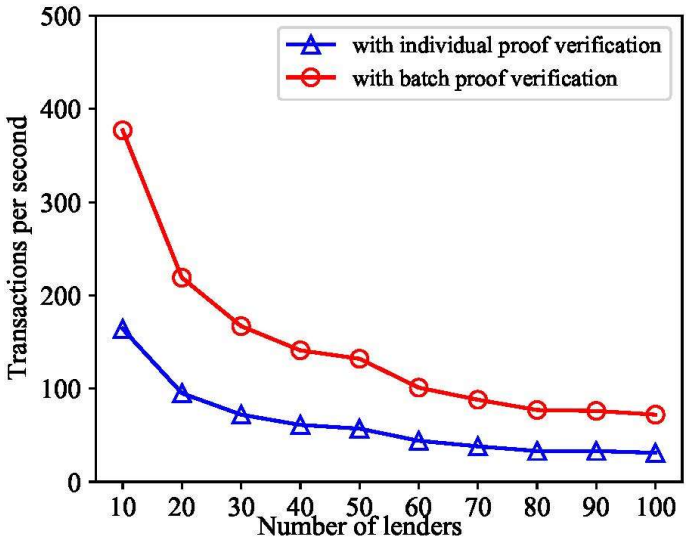


Figure 4.1: Type A transaction throughput (*TPS*).

Since the cost of signature verification in a Type B transaction is negligible, its throughput is mainly reliant on the consensus protocol. The experiment reports 1,737 *TPS* with 50 lenders and 954 *TPS* with 100 lenders respectively. Type C transactions are triggered by

occasional complaints from participants. Thus, they are a concern here and we do not report its throughput.

4.4 Time Cost

We report the average time cost for different steps in proposed the scheme.

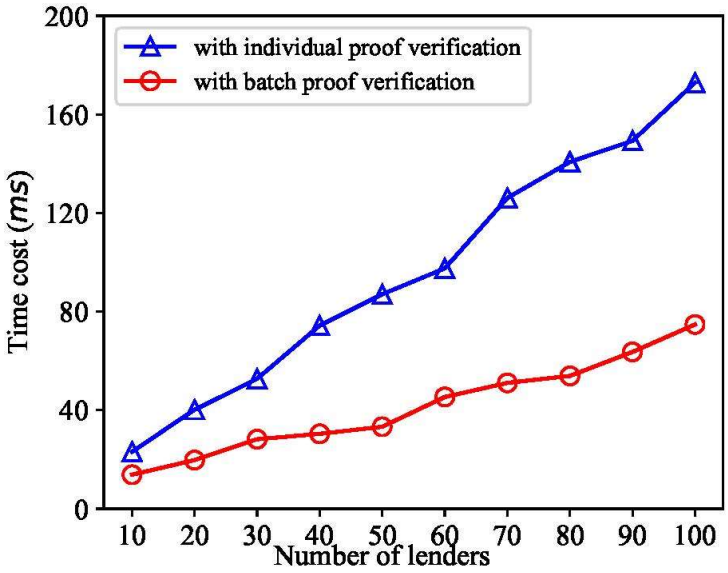


Figure 4.2: Time cost for joint lending decision (*ms*).

4.4.1 System Initialization

DKG protocol [60] is exploited to initialize the threshold cryptosystem for the lender group. Note that this is usually one-time or low-frequency costs due to the stable composition of the syndicate in practice. Our experiment shows that it takes about 25 seconds to execute the DKG protocol for a syndicate of 100 lenders, where the threshold is set to be $67 = [100 \cdot 2/3] + 1$.

4.4.2 MPC for Joint Lending Decision

In practice, a borrower usually gets the result within seconds on submitting an online syndicated loan request. We measure the time cost that is introduced by our ledger-based MPC and added to the application processing. Specifically, the extra time is spent on the three presented fundamental algorithms, i.e., verifiable encryption, verifiable plaintext consistency and verifiable joint decryption, which can be executed concurrently by different lenders. On the other hand, the number of NIZKPs that needs to be verified is increasingly proportional to the number of lenders in a syndicate. Figure 4.2 shows the expected linear growth of the operational overhead with the increased lenders. We may adopt batch verification to further reduce the delay. Considering a large syndicate of 100 lenders and a typical 2.4 s loan processing time with the current online lending system,

the proposed MPC with batch proof verification only contributes to additional 75 ms , i.e., 3.1% overhead increase.

4.4.3 Daily Settlement

Daily settlement is performed off-chain between the agent and each lender during a day. The time cost is independent of the number of lenders and mainly relies on the number of Type A and Type B transactions. Specially, a lender computes the net differential and the associated random number. The operations of the agent include: computing Pedersen commitments of the lender's loan shares and repayment allocations, aggregating all the commitments to obtain Pedersen commitment of the net differential, plus verifying the net differential from its commitment. For example, to deal with 200,000 and 200,000 Type A and Type B Transactions per day, the system takes about 181.6 ms to finish the settlement between the agent and a lender.

4.5 Concluding Remarks

In this Chapter, we streamline the framework design to optimize performance for the real-world requirements (e.g., the number of lenders, loan request throughput, loan processing time, etc.) and develop a prototype system on a famous permissionless blockchain platform utilized in a lot of financial applications. In terms of transaction throughput, our system throughput can be 31.6 times greater (73 *TPS*) compared with the commercial need to approximately process 2.31 *TPS* on average with a large syndicate of 100 lenders. In terms of time consumption, the typical online lending system need 2.4 s to process a loan, while our proposed MPC with batch proof verification only contributes to additional 75 ms. As a result, our experiment shows satisfactory performance on a real workload with a large syndicate of 100 lenders.

5 Conclusion And Future Work

5.1 Summary and Research Work

With the boom of FinTech in emerging markets, a whole new and fast-growing type of financial services named online syndicated lending enables Internet borrowers to apply for loans through an agent bank and receive the money with seconds, which has served hundreds of millions of people. This has raised concerns about data security, capital security and regulability of this business. so in this paper, we investigate two existing types of syndicated lendings: traditional syndicated lending and online syndicated lending, and compare them in terms of participants, processes, and advantages and disadvantages. We found that online syndicated lending is “collection-then-distribution” transaction model due to the presence of agents as intermediaries, which leads to problems in fairness, transparency, privacy protection and regulation, and poses a major security risk to funds, loan request, share and repayment data. Based on this, we study the existing financial privacy protection solutions and found that they mainly solve the fairness and data privacy protection problems under the “peer-to-peer” transaction model, and are not suitable for the “collection-then-distribution” transaction model of online syndicated lending.

In order to address this problem, the blockchain technology and several cryptographic building blocks are researched in depth. As a result, this thesis presents a novel collaborative financial ledger for online syndicated lending, which may also be of independent interest. The proposed ledger-enabled MPC leverages homomorphic encryption/commitment to enable the efficient reuse of intermediary transactional states without breaking the privacy promise during the full life-cycle of a loan. Our system further minimizes the privacy leakage in the settlement phase by carefully aggregating daily loan share and repayment information. Besides, our framework enables the regulator to efficiently verify the encrypted key transactional activities. Moreover, we perform a complete analysis and demonstration of the security and privacy protection capabilities of the proposed framework. Finally, we streamline the framework design to optimize performance for the real-world requirements and develop a prototype system on a popular and non-profit consortium blockchain. Our experiment shows satisfactory performance on a real workload with a large syndicate of 100 lenders.

5.2 Summary of Outcomes

The outcomes of this thesis are summarized as follows: (1) we present a new ledger-based MPC framework for the online syndicated lending. The system supports confidential transaction recording on-chain and secure computation over transactions off-chain during the full life cycle of the loan. We formally define and prove the security of this new type of ledger. The underlying MPC framework may be of independent interest to applications in a similar model. (2) the audibility is supported in the proposed ledger system. A regulator with the associated private key can examine loan details and lending

activities that are recorded on the blockchain to ensure the regulatory compliance. We accomplish this goal by allowing the regulator to publicly check the private states of the MPC for loan request processing. (3) we develop a prototype system running on top of a leading permissionless blockchain platform. The thorough system evaluation shows a satisfactory performance that meets the real-world requirements, i.e., a minimal 75 ms delay for loan application processing and 31.6 times greater throughput versus the practical demand.

5.3 Potential Applications

As we see, the proposed scheme is suitable for application scenarios in which participants have the following characteristics or requirements: first, equal relationships and distributed deployment, second, data interaction needs, third, data privacy protection needs, and fourth, regulatory needs. Once the participants of any scenario have all or part of the above characteristics and requirements, the feasibility of the designed solution in that scenario can be considered. For instance, in supply chain finance, the participants in this scenario include banks, financial institutions, and different enterprises upstream and downstream of the supply chain. Besides, there are very complex debtor-creditor relationships among enterprises, and enterprises within the whole supply chain may also join together to lend to banks or financial institutions. The proposed solution can protect the debtor-creditor relationship between enterprises and the amount of bank loans from being known by other enterprises in the supply chain, and support the normal settlement between enterprises under the condition of privacy protection. Furthermore, Regulatory agencies can audit the supply chain financial activities in a friendly and low-cost manner. Similarly, the proposed scheme can be adopted in scenarios such as joint power supply and electricity settlement in smart grid, traditional syndicated loan, etc.

5.4 Future Work

Strong fairness is a key feature of the realized platform by exploiting the proposed ledger-based MPC. Financial institutions are equally treated in terms of lending participation. On the one hand, a fair and data confidentiality bidding function could be considered to be introduced in the proposed system, where all lenders encrypt and commit to their own rates, and submit them to the blockchain. The agent verifies the correctness of the ciphertext and commitment based on the data downloaded from blockchain, and completes a privacy-preserving ciphertext ranking as a basis for selecting the appropriate lender in the current round of syndicated loans. We will work on developing this function while preserving privacy and carrying out correct settlement. On the other hand, there are still a mass of processes that rely on paper documents in the traditional syndicated lending. By combining OCR recognition, Natural Language Processing (NLP) and other technologies, the architecture proposed in this thesis may be applied to traditional syndicated lending, which could solve the duplicate financing fraud issues in addition to the problems of privacy leakage and difficult supervision. This is also an important direction for our future research.

BIBLIOGRAPHY

[1] Thomas A Durkin and Gregory E Elliehausen. Truth in lending: Theory, history, and a way forward. Financial Management Associati, 2011.

[2] CFI. Syndicated loan: A loan facility offered by a group of lenders to a large borrower, April 2022. https://corporatefinanceinstitute.com/resources/knowledge/finance/syndicated-loan/.

[3] Federal Reserve. Syndicated loan portfolios of financial institutions, March 2020. https://www.federalreserve.gov/releases/efa/efa-project-syndicated-loan-portfolios-of-financial-institutions.html.

[4] Bridget Marsh and Tess Virmani. Loan syndications and trading: An overview of the syndicated loan market 2020, March 2020. https://iclg.com/practice-areas/lending-and-secured-finance-laws-and-regulations/01-loan-syndications-and-trading-an-overview-of-the-syndicated-loan-market.

[5] REFINITIV. Global syndicated loan review, full year 2021. https://www.refinitiv.com/en/financial-data/market-data/lpc-loan-pricing.

[6] Hongyuran Wu, Yue Hu, and Wei Han. In depth: Cheers and fears in $283 billion bank-tech lending tie-up. https://www.caixinglobal.com/2019-10-27/in-depth-cheers-and-fears-in-283-billion-bank-tech-lending-tie-up-101475874.html.

[7] Webank 2021 annual report. [Online], 2022. https://tctp.webankcdn.net/owb-res/owb-res/www/2.0/pdf/annual_report_2021.pdf.

[8] Seong Joon Oh, Bernt Schiele, and Mario Fritz. Towards reverse-engineering black-box neural networks. In Explainable AI: Interpreting, Explaining and Visualizing Deep Learning, pages 121–144. Springer, 2019.

[9] Florian Tramer, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In 25th {USENIX} Security Symposium ({USENIX} Security 16), pages 601–618, 2016.

[10] BinghuiWang and Neil Zhenqiang Gong. Stealing hyperparameters in machine learning. In 2018 IEEE Symposium on Security and Privacy (SP), pages 36–52. IEEE, 2018.

[11] Digital Asset Platform. [Online], 2020. https://www.digitalasset.com/.

[12] Stellar - an open network for money. [Online], 2020. https://www.stellar.org/.

[13] JP Morgan Chase. Quorum whitepaper. [Online], 2016. https://github.com/ConsenSys/quorum/blob/master/docs/Quorum%20Whitepaper%20v0.2.pdf

[14] Ethan Heilman, Leen Alshenibr, Foteini Baldimtsi, Alessandra Scafuro, and Sharon Goldberg. Tumblebit: An untrusted bitcoin-compatible anonymous payment hub. In Network and Distributed System Security Symposium, 2017.

[15] Greg Maxwell. Coinjoin: Bitcoin privacy for the real world. In Post on Bitcoin forum, 2013.

[16] Tim Ruffing, Pedro Moreno-Sanchez, and Aniket Kate. Coinshuffle: Practical decentralized coin mixing for bitcoin. In European Symposium on Research in Computer Security, pages 345–364. Springer, 2014.

[17] Luke Valenta and Brendan Rowan. Blindcoin: Blinded, accountable mixes for bitcoin. In International Conference on Financial Cryptography and Data Security, pages 112–126. Springer, 2015.

[18] Gregory Maxwell. Confidential transactions, technical report (2015). https://epic.tech/glossary/gregory-maxwell/.

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

[20] Ethan Cecchetti, Fan Zhang, Yan Ji, Ahmed Kosba, Ari Juels, and Elaine Shi. Solidus: Confidential distributed ledger transactions via pvorm. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pages 701–717, 2017.

[21] Ahmed Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In 2016 IEEE symposium on security and privacy (SP), pages 839–858. IEEE, 2016.

[22] Shen Noether and Sarang Noether. Monero is not that mysterious. Technical report, 2014.

[23] Jan Camenisch, Aggelos Kiayias, and Moti Yung. On the portability of generalized schnorr proofs. In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pages 425–442. Springer, 2009.

[24] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct noninteractive zero knowledge for a von neumann architecture. In Proceedings of the

23rd USENIX Conference on Security Symposium, SEC'14, page 781–796, USA, 2014. USENIX Association.

[25] Andrew C. Yao. Protocols for secure computations. In Proceedings of the 23rd Annual Symposium on Foundations of Computer Science, SFCS '82, page 160–164, USA, 1982. IEEE Computer Society.

[26] Andrew Chi-Chih Yao. How to generate and exchange secrets. In Proceedings of the 27th Annual Symposium on Foundations of Computer Science, SFCS '86, page 162–167, USA, 1986. IEEE Computer Society.

[27] Shafi Goldwasser. How to play any mental game, or a completeness theorem for protocols with an honest majority. Proc. the Nineteenth Annual ACM STOC'87, pages 218–229, 1987.

[28] Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Lukasz Mazurek. Secure multiparty computations on bitcoin. In 2014 IEEE Symposium on Security and Privacy, pages 443–458. IEEE, 2014.

[29] Iddo Bentov and Ranjit Kumaresan. How to use bitcoin to design fair protocols. In Annual Cryptology Conference, pages 421–439. Springer, 2014.

[30] Aggelos Kiayias, Hong-Sheng Zhou, and Vassilis Zikas. Fair and robust multi-party computation using a global transaction ledger. In Marc Fischlin and Jean-S´ebastien Coron, editors, Advances in Cryptology – EUROCRYPT 2016, pages 705–734, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[31] Ranjit Kumaresan and Iddo Bentov. How to use bitcoin to incentivize correct computations. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, CCS '14, page 30–41, New York, NY, USA, 2014. Association for Computing Machinery.

[32] Arka Rai Choudhuri, Matthew Green, Abhishek Jain, Gabriel Kaptchuk, and Ian Miers. Fairness in an unfair world: Fair multiparty computation from public bulletin boards. InProceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS '17, page 719–728, New York, NY, USA, 2017. Association for Computing Machinery.

[33] Cameron Harwick. Incentives in blockchain design and applications. Available at SSRN, 2020.

[34] FISCO BCOS. [Online], 2020. http://www.fisco-bcos.org/.

[35] Blaise Gadanecz. The syndicated loan market: structure, development and implications. BIS Quarterly Review, December, 2004.

[36] Jongha Lim, Bernadette A Minton, and Michael S Weisbach. Syndicated loan spreads and the composition of the syndicate. Journal of Financial Economics, 111(1):45–69, 2014.

[37] Steven A Dennis and Donald J Mullineaux. Syndicated loans. Journal of financial intermediation, 9(4):404–426, 2000.

[38] Christian Montag, Benjamin Becker, and Chunmei Gan. The multipurpose application wechat: a review on recent research. Frontiers in psychology, 9:2247, 2018.

[39] Lerong Lu. Decoding alipay: mobile payments, a cashless society and regulatory challenges. Butterworths Journal of International Banking and Financial Law, pages 40–43, 2018.

[40] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Technical report, Manubot, 2019.

[41] Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Xiangping Chen, and Huaimin Wang. Blockchain challenges and opportunities: A survey. International Journal of Web and Grid Services, 14(4):352–375, 2018.

[42] Michael Nofer, Peter Gomber, Oliver Hinz, and Dirk Schiereck. Blockchain. Business & Information Systems Engineering, 59(3):183–187, 2017.

[43] Chris Dannen. Introducing Ethereum and solidity, volume 318. Springer, 2017.

[44] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper, 151(2014):1–32, 2014.

[45] Michael J Fischer, Nancy A Lynch, and Michael S Paterson. Impossibility of distributed consensus with one faulty process. Journal of the ACM (JACM), 32(2):374–382, 1985.

[46] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance and proactive recovery. ACM Transactions on Computer Systems (TOCS), 20(4):398–461, 2002.

[47] John R Douceur. The sybil attack. In International workshop on peer-to-peer systems, pages 251–260. Springer, 2002.

[48] Sunny King and Scott Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. self-published paper, August, 19(1), 2012.

[49] Daniel Larimer. Delegated proof-of-stake (dpos). Bitshare whitepaper, 81:85, 2014.

[50] Nick Szabo. Formalizing and securing relationships on public networks. First monday, 1997.

[51] Joshua. Ethereum virtual machine (evm). [Online], 2022. https://ethereum.org/en/developers/docs/evm/.

[52] Chaincode tutorials. [Online], 2018. https://hyperledger-fabric.readthedocs.io/en/release-1.3/chaincode.html.

[53] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In OSDI, pages 173–186, 1999.

[54] Vitalik Buterin et al. A next-generation smart contract and decentralized application platform. white paper, 3(37), 2014.

[55] The fabric tutorials. [Online], 2021. https://hyperledger-fabric.readthedocs.io/en/latest/tutorials.html.

[56] Yuan Lu, Qiang Tang, and Guiling Wang. Dragoon: Private decentralized hits made practical. In 2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS), pages 910–920, 2020.

[57] Yuan Lu, Qiang Tang, and Guiling Wang. Zebralancer: Private and anonymous crowdsourcing system atop open blockchain. In 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), pages 853–865, 2018.

[58] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE transactions on information theory, 31(4):469–472, 1985.

[59] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In Theory of Cryptography Conference, pages 325–341. Springer, 2005.

[60] Rosario Gennaro, Stanis law Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. In International Conference on the Theory and Applications of Cryptographic Techniques, pages 295–310. Springer, 1999.

[61] Adi Shamir. How to share a secret. Communications of the ACM, 22(11):612–613, 1979.

[62] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Annual International Cryptology Conference, pages 129–140. Springer, 1991.

[63] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. SIAM Journal on computing, 18(1):186–208, 1989.

[64] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Conference on the theory and application of cryptographic techniques, pages 186–194. Springer, 1986.

[65] C. P. Schnorr. Efficient signature generation by smart cards. Journal of Cryptology, 4(3):161–174, 1991.

[66] Stefan Brands. Untraceable off-line cash in wallet with observers. In Annual international cryptology conference, pages 302–318. Springer, 1993.

[67] David Chaum and Torben Pryds Pedersen. Wallet databases with observers. In Annual International Cryptology Conference, pages 89–105. Springer, 1992.

[68] Stefan Brands. Rapid demonstration of linear relations connected by boolean operators. In International Conference on the Theory and Applications of Cryptographic Techniques, pages 318–333. Springer, 1997.

[69] Benedikt Bunz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In 2018 IEEE Symposium on Security and Privacy (SP), pages 315–334. IEEE, 2018.

[70] Webank: China's financial unicorn, July 2019. https://www.euromoney.com/article/b1h9dbtpshwh66/webank-chinas-financial-unicorn.

[71] Anti-money laundering, 2020. https://www.finra.org/rules-guidance/key-topics/aml.

[72] Christopher Copeland and Hongxia Zhong. Tangaroa: a byzantine fault tolerant raft, 2016.

[73] David Mazieres. The stellar consensus protocol: A federated model for internet-level consensus. Stellar Development Foundation, 32, 2015.

[74] WeBank, 2020. https://www.webank.com/.

[75] Mihir Bellare, Juan A Garay, and Tal Rabin. Fast batch verification for modular exponentiation and digital signatures. In International Conference on the Theory and Applications of Cryptographic Techniques, pages 236–250. Springer, 1998.

[76] Daniel M Gordon et al. A survey of fast exponentiation methods. J. Algorithms, 27(1):129–146, 1998.

[77] Ran Canetti. Security and composition of multiparty cryptographic protocols. Journal of CRYPTOLOGY, 13(1):143–202, 2000.

[78] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, STOC '87, page 218–229, New York, NY, USA, 1987. Association for Computing Machinery.

[79] curve25519-dalek, 2020. https://github.com/dalek-cryptography/curve25519-dalek.

[80] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ecdsa). International journal of information security, 1(1):36–63, 2001.

[81] secp256k1, 2020. https://github.com/rust-bitcoin/rust-secp256k1/.