METHODS AND APPARATUS FOR BLOCKCHAIN INTEROPERABILITY AND IDENTITY MANAGEMENT

Bishakh Chandra Ghosh

METHODS AND APPARATUS FOR BLOCKCHAIN INTEROPERABILITY AND IDENTITY MANAGEMENT

Thesis submitted to the
Indian Institute of Technology, Kharagpur
For award of the degree

of

Doctor of Philosophy

by

Bishakh Chandra Ghosh

Under the supervision of

Prof. Sandip Chakraborty

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

September 2023

APPROVAL OF THE VIVA-VOCE BOARD

Date: / / 22

Certified that the thesis entitled **"Methods and Apparatus for Blockchain Interoperability and Identity Management"** submitted by Bishakh Chandra Ghosh to the Indian Institute of Technology, Kharagpur, for the award of the degree of Doctor of Philosophy has been accepted by the external examiners and that the student has successfully defended the thesis in the viva-voce examination held today.

(Member of DSC) (Member of DSC) (Member of DSC)

(Supervisor)

(External Examiner) (Chairman)

CERTIFICATE

This is to certify that the thesis entitled ***"Methods and Apparatus for Blockchain Interoperability and Identity Management"****, submitted by Bishakh Chandra Ghosh to the Indian Institute of Technology, Kharagpur, for the partial fulfillment of the award of the degree of Doctor of Philosophy in Computer Science and Engineering, is a record of bona fide research work carried out by him under my supervision and guidance. The thesis in my opinion, is worthy of consideration for the award of the degree of Doctor of Philosophy in accordance with the regulations of the Institute. To the best of my knowledge, the results embodied in this thesis have not been submitted to any other University or Institute for the award of any other Degree or Diploma.*

Sandip Chakraborty
Associate Professor
Department of Computer
Science and Engineering,
IIT Kharagpur

Date:

DECLARATION

I certify that

a. The work contained in this thesis is original and has been done by me under the guidance of my supervisors.

b. The work has not been submitted to any other Institute for any degree or diploma.

c. I have followed the guidelines provided by the Institute in preparing the thesis.

d. I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

e. Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.

f. Whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

Bishakh Chandra Ghosh

Dedicated to my parents

Kanika Ghosh
and
Late Bijan Bihari Ghosh

ACKNOWLEDGMENTS

This thesis is an outcome of a long journey, and in this section, I humbly appreciate all the people without whose guidance and support it would not have been possible. Firstly, I would like to express my deepest gratitude towards my supervisor Dr. Sandip Chakraborty, who motivated me toward this academic goal right from my undergraduate days. Apart from his guidance in research, his unflinching push towards developing practical systems with real-world significance provided me with a flourishing environment that is nothing less than an oasis in today's result-oriented academic landscape. It is only because of his cheerful attitude and contagious positive spirit that I could get through the ever-demanding journey of Ph.D.
I want to thank Venkatraman Ramakrishna for mentoring me from my internship days at IBM Research - India, till the submission of my Ph.D. thesis. Through countless technical discussions and brainstorming sessions, along with insightful suggestions from the industry perspective, he helped me gain the required technical expertise for achieving impactful research outcomes.
I sincerely thank Prof. C Pandu Rangan and Prof. Hans-Georg Fill for generously dedicating their time to review the thesis, providing insightful comments and suggestions that significantly enhanced its quality.

I would also like to sincerely thank my Doctoral Scrutinee Committee (DSC), – Prof. Niloy Ganguly, Prof. Arobinda Gupta, Dr. Somindu Chaya Ramanna, and Dr. Goutam Sen, for their advice and constructive criticisms, which helped me to improve the thesis.
I thank the current HoD, Prof. Arobinda Gupta, the past HoDs, Prof. Dipanwita Roy Chowdhury and Prof. Sudeshna Sarkar, and all the Department of Computer Science and Engineering faculties for extending their guidance in both technical and administrative processes. I express my gratitude to the director, Prof. Virendra Kumar Tewari, all the Deans, and other officials of IIT Kharagpur, whose collective efforts have provided us with an environment for world-class research and studies.
I also thank all the anonymous reviewers of all my publications who have helped me improve the quality of this thesis.
I want to thank Dr. Rajeev Shorey, UQIDAR – IIT Delhi, for giving me the opportunity to serve as a web chair in the COMSNETS conference organizing committee.

This thesis is a result of many people's collective efforts, especially my collaborators, to whom I convey my deepest gratitude. Among them, Dhinakaran Vinayagamurthy and Sikhar Patranabis were more like mentors who nurtured my expertise in cryptography and security. I thank Krishnasuri Narayanam, Chander Govindarajan,

Dushyant Behl, Dileban Karunamoorthy, and Ermyas Abebe for their support. I want to convey my sincere gratitude to Prof. Soumya K. Ghosh, and Sourav Kanti Addya, who provided me with much-needed encouragement during my initial days at IIT Kharagpur. I have been fortunate enough to work with fantastic peers, student collaborators, and interns. Thank you, Shubha Brata Nath, Tanay Bhartia, Nishant Baranwal Somy, and Hrishabh Sharma. I have learned a lot from all of you.

Dr. Sujoy Saha's mentorship and guidance throughout my undergraduate days and beyond provided me the courage to pursue Ph.D. I am grateful to him and Prof. Subrata Nandi for providing me with the best possible opportunities during my time at NIT Durgapur and building the foundation within me to pursue higher studies. I sincerely thank my peers at NIT Durgapur, Hridoy Datta, Partha Sarathi Paul, Kingshuk De, Prithviraj Pramanik, Arka Prava Basu, and Sunny Saurav for our fruitful technical discussions over countless cups of tea.

I have been fortunate to learn from some of the best faculties at IIT Kharagpur, as well as NIT Durgapur. I was privileged to learn in-depth about cryptography and network security from Dr. Jaydeep Howlader, and Prof. Debdeep Mukhopadhyay. Dr. Debasis Mitra's art of teaching made me realize that an instructor can make learning any subject/discipline equally exciting. I learned the most about computer science as a whole while attending his classes on data structures and algorithms, computer architecture, and VLSI.

My sincere gratitude goes to Sadhan-da, Bappa-da, Durga-da, and other staff from the CSE department for their timely assistance in handling the official procedures. I must mention Moni masi, whose homely meals helped me survive; thank you.

Peers, seniors, and juniors of my lab and other labs played a key role in shaping me toward where I stand. I am thankful to Bidisha di, Ayan da, Basabdatta di, Rohit da, Surjya da, Satadal da, Madhumita di, Paramita di, for your support and guidance as wonderful seniors. Thanks to Bishal da and Anurag da for helping me in the server team. I thank Punyajoy, Rajdeep da, Abhishek da, Soumi di, Binny da, Rima di, Kalyani di, Soumyadeep da, Mainul da, Gourab da, Salma di, Paramita Das di, Harsh, Soumya, for being fantastic peers and making my time at IIT Kharagpur more enjoyable.
I am fortunate to have made great friends during this journey, and I value these friendships more than the degree. I lived, learned, and grew with them for four years. I was inspired by Sumitro da's meticulous work management and attention to detail, though I could not absorb enough to get close to his skills. Snigdha di often provided me (and others in the lab) with her philosophical take on subjects which was in contrast to our usual bluntness. Souvic gave us the much-needed energy after the COVID lockdown period, which revived us into the usual flow of life. Abhijit da is an ocean of knowledge from whom I have learned the most in these four years as a CS engineer. His down-to-earth attitude is something I look up to, and Soumyajit da is another man with a similar trait. The only thing that amazes

me more than Soumyajit da's knowledge as a researcher is his enthralling personality. Our late-night music sessions with Soumyajit da and Arnab da are some of the best memories of my time at IIT Kharagpur. Thanks to Arnab da's guitar tabs and research papers, I now get to learn cryptography and music from the same author! The morning football and cricket matches gave the much-needed break from the usual lab routine. Thanks to Soumyadyuti da, Akashdeep, Siddhartha da, Anirban da, and Manaar da for arranging them and including me despite my eye-watering skills. My performance in playing CS was perhaps much better. Thanks to Durba di, SD, SK, Kungfu too for joining the fights. I want to thank Avirup for tagging along with me through the initial days at Kgp.

I want to thank and convey my best wishes to my current peers, Lalan, Utkalika, Anirban (Add), Sugandh, Debasree di, Aritra, Prasenjit (foss), and Argha. I wish I could spend some more days shouting 'fossils', going to PD, and in general, doing what 'paagal log' do. You all are insanely talented, and I will keep on congratulating you in the future on the countless successes that you are about to get.

Outside the boundaries of IIT Kharagpur, I was fortunate to have friends including Rajendra, Sayan (Kuila), Sumit, Subhadip, Arkadipta, Anirban, Kajal, Bumba da, Deepjyoti, Sayan (sei sei), who have been by my side for many, many years now.

This thesis is dedicated to the constants of my life. Arpita has always been there for me through good times and bad times. Despite the pressure of her own Ph.D. and her own issues, her constant presence, undwindling support, and subtle scoldings steered me clear of some of the toughest times of my life. Thank you for being the friend I can rely on and the pillar I can lean on. If I am asked to name my role models, my brother Binayak is always going to be on that list. On the one hand, he is like a shadow who is always there if I need to look back and reach out; on the other hand, he is like a milestone that lures me to greater heights. It is his financial as well as emotional backing that allowed me to leave a job and pursue higher education. I dedicate this thesis to the loving memory of my father Bijan Bihari Ghosh. The most important person in my life is my mother, Kanika Ghosh. Whatever big or small achievements I have is because of her impeccable parenting, unconditional support, and unlimited blessings. No words can ever express how indebted I am to her. Thank you.

Bishakh Chandra Ghosh
IIT Kharagpur, India

Author's Biography

Bishakh Chandra Ghosh received his B.Tech. degree in Information Technology from National Institute of Technology Durgapur, India in 2018. His areas of interest includes distributed systems, cloud computing, and web technology. He spent three months with IBM Research, India as an intern in 2020, where he worked on blockchain interoperability.

Publications from the Thesis

1. **Bishakh Chandra Ghosh**, Sikhar Patranabis, Dhinakaran Vinayagamurthy, Venkatraman Ramakrishna, Krishnasuri Narayanam, Sandip Chakraborty, *"Private Certifier Intersection"*, 2023 Network and Distributed System Security (NDSS) Symposium, San Diego, California, 27 February − 3 March 2023.

2. **Bishakh Chandra Ghosh**, Tanay Bhartia, Sourav Kanti Addya, Sandip Chakraborty, *"Leveraging Public-Private Blockchain Interoperability for Closed Consortium Interfacing"*, 2021 IEEE Conference on Computer Communications (IEEE INFOCOM), Virtual Conference, 10 May − 13 May 2021.

3. **Bishakh Chandra Ghosh**, Venkatraman Ramakrishna, Chander Govindarajan, Dushyant Behl, Dileban Karunamoorthy, Ermyas Abebe, Sandip Chakraborty, *"Decentralized Cross-Network Identity Management for Blockchain Interoperation"*, 2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), Virtual Conference, 3 May − 6 May 2021.

4. **Bishakh Chandra Ghosh**, Dhinakaran Vinayagamurthy, Venkataraman Ramakrishna, Krishnasuri Narayanam, Sandip Chakraborty, *"Privacy-Preserving Negotiation of Common Trust Anchors Across Blockchain Networks"*, 2022 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), Virtual Conference, 2 May − 5 May 2022 [Short Paper].

5. **Bishakh Chandra Ghosh**, Sourav Kanti Addya, Anurag Satpathy, Soumya K. Ghosh and Sandip Chakraborty, *"Towards a Democratic Federation for Infrastructure Service Provisioning"*, 2019 IEEE International Conference on Services Computing (SCC), Milan, Italy, 8 July − 13 July 2019. [Short Paper]

6. **Bishakh Chandra Ghosh**, and Sandip Chakraborty, "Trustless Collaborative Cloud Federation", in IEEE Transactions on Cloud Computing. **[Under Revision]**

Other Publications

The following is a list of publications during my tenure as a student at IIT Kharagpur, which are not a part of this thesis.

1. Sourav Kanti Addya, Anurag Satpathy, **Bishakh Chandra Ghosh**, Sandip Chakraborty, Soumya K Ghosh, Sajal K Das. "Geo-distributed Multi-tier Workload Migration over Multi-timescale Electricity Markets" IEEE Transactions on Services Computing 2023.

2. Dhaval Thummar, Yerramaddu Jahnavi, Mudavath Prathyusha, Sayad Shahanaz, **Bishakh Chandra Ghosh**, Sourav Kanti Addya. "DeSAT: Towards Transparent and Decentralized University Counselling Process" IEEE International Conference on Blockchain (Blockchain) 2022.

3. Yerramaddu Jahnavi, Mudavath Prathyusha, Sayad Shahanaz, Dhaval Thummar, **Bishakh Chandra Ghosh**, Sourav Kanti Addya. "Democratizing University Seat Allocation using Blockchain" 14th International Conference on COMmunication Systems & NETworkS (COMSNETS) [Demo] 2022.

4. Chander Govindarajan, **Bishakh Chandra Ghosh**, Nitin Gaur, Venkatraman Ramakrishna, Dushyant K. Behl, Petr Novotny. "Blockchain Declarative Descriptor for Cross-network Communication" US Patent Application 2021.

5. Petr Novotny, Venkatraman Ramakrishna, Chander Govindarajan, Dushyant K Behl, **Bishakh Chandra Ghosh**, Nitin Gaur. "Blockchain network identity management using ssi" US Patent Application 2021.

6. Sourav Kanti Addya, Anurag Satpathy, **Bishakh Chandra Ghosh**, Sandip Chakraborty, Soumya K Ghosh, Sajal K Das. "CoMCLOUD: Virtual Machine Coalition for Multi-Tier Applications over Multi-Cloud Environments." IEEE Transactions on Cloud Computing. 2021.

7. Partha Sarathi Paul, **Bishakh Chandra Ghosh**, Ankan Ghosh, Sujoy Saha, Subrata Nandi, Sandip Chakraborty. "Disaster Strikes! Internet Blackout! What's the Fate of Crisis Mapping?" 22nd International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI) 2020.

8. Nishant Baranwal Somy, Abhijit Mondal, **Bishakh Chandra Ghosh**, Sandip Chakraborty. "System call interception for serverless isolation." SIGCOMM [Poster] 2020.

9. **Bishakh Chandra Ghosh**, Sourav Kanti Addya, Nishant Baranwal Somy, Shubha Brata Nath, Sandip Chakraborty, Soumya K. Ghosh. "Caching Techniques to Improve Latency in Serverless Architectures." International Conference on COMmunication Systems & NETworkS (COMSNETS) [Poster] 2020.

10. Partha Sarathi Paul, **Bishakh Chandra Ghosh**, Ankan Ghosh, Sujoy Saha, Subrata Nandi, Sandip Chakraborty. "Aco-Wi : Acoustic Initiated Wi-Fi Peer-group Communication for Opportunistic Messaging" 22nd International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI) [Late Breaking Results] 2020.

11. Sourav Kanti Addya, Anurag Satpathy, **Bishakh Chandra Ghosh**, Sandip Chakraborty and Soumya K. Ghosh "Power and Time aware VM Migration for Multi-tier Applications over Geo-distributed Clouds" IEEE 12th International Conference on Cloud Computing (CLOUD) 2019.

12. Partha Sarathi Paul, **Bishakh Chandra Ghosh**, Hridoy Sankar Dutta, Kingshuk De, Arka Prava Basu, Prithviraj Pramanik, Sujoy Saha, Sandip Chakraborty, Niloy Ganguly, and Subrata Nandi. "CRIMP: Here crisis mapping goes offline." Journal of Network and Computer Applications 2019.

13. Partha Sarathi Paul, Chandrika Mukherjee, **Bishakh Chandra Ghosh**, Sudipta Pandit, Sujoy Saha, Subrata Nandi. "On designing a fast-deployable'localized'GIS platform for using'offline'during post-disaster situation" 20th International Conference on Distributed Computing and Networking (ICDCN) [EmeRTeS Workshop] 2019.

ABSTRACT

Development of permissioned distributed ledger technology has introduced blockchains as a viable solution for decentralizing business-to-business interactions in closed consortium networks. Consequently, in recent years, apart from the usual applications around cryptocurrencies, blockchain has seen application in enterprise scenarios such as supply chain, trade finance, logistics, energy trading, etc. However, such rapid adoption and deployment of permissioned networks have introduced technical fragmentation in terms of protocols, algorithms, formats, architecture, and policies (e.g. governance models). This heterogeneity stands as a barrier between different permissioned consortium blockchain networks that wish to interoperate and communicate for broader business goals. Moreover, permissioned distributed ledgers by design were made private so that entities that are not consortium members and, thus outside the network boundary, have no visibility over the network's data, assets, or functions. As a result, there are no means for the consortium to communicate with external entities such as end-users or consumers to which the businesses that form the consortium might need to dispense services. This thesis studies the challenges, existing works, and gaps in both permissionless-permissioned and permissioned-permissioned blockchain interoperability and introduces methods and apparatus for enabling interoperability.

Towards enabling permissioned consortium blockchains to communicate with their end-consumers in the open network, this thesis proposes the first framework and set of protocols for permissionless-permissioned blockchain interoperability. This includes a mechanism allowing the transfer of data such as end-consumer requests, from the open network to the consortium, while ensuring consensus of the consortium participants on the data and the order in which the data arrive. Another protocol facilitates verifiable transfer of data from the consortium to the public blockchain. A proof-of-concept implementation for the use case of cloud federations and scalability evaluations demonstrates the system's viability.

Existing protocols for inter-consortium verifiable transfer of data require the pre-configuration of identities such as public keys / certificates of the blockchain participants. For robust permissioned-permissioned interoperability while eliminating manual identity configuration, in this thesis, we propose an architecture and

protocols for exchanging identities across permissioned blockchain networks. We introduce a decentralized identity infrastructure for trust basis that utilizes the decentralized identifier and verifiable credential concepts. Our solution requires minimal changes to the existing deployed networks, and we demonstrate its usability by applying it in a trade-finance and trade-logistics interoperability scenario.

Determining a common trust basis is an essential requirement for cross-blockchain identity exchange. However, revealing all trust anchors results in loss of privacy since the trust anchors are often well-known organizations such as governments, NGOs, large companies, political organizations, etc. Revealing a trust anchor reveals the entity's association with the same. We develop protocols for privacy preserving negotiation of common trust anchors across blockchain networks to facilitate cross-chain identity exchange. We propose two variants of solutions, one with the active participation of the trust anchors themselves and the other without involving the trust anchors using secure multiparty computation. Through experiments, we evaluate the efficiency of our protocol, and we also prove its security against malicious adversaries.

Finally, we extend the privacy-preserving trust anchor negotiation for blockchains to a more general use case of any verifiable credential presentation flow that requires a common credential certifier. We formally define this problem as 'private certifier intersection' (PCI). Through a novel extension of secret sharing based secure multiparty computation protocols for elliptic curve pairings, we introduce efficient solutions for two different variants of PCI. We perform a detailed evaluation of the protocols on consumer hardware in a real-world setting by placing parties at two different continents.

To summarize, we introduce methods and apparatus for blockchain interoperability and identity management for facilitating cross-chain interactions between permissioned networks and permissionless networks, as well as between different permissioned networks. Bridging the gaps between different decentralized systems, our conrtibutions pave the way for end-to-end connected decentralized networks starting from closed groups of entities such as businesses, and ending at the open network of end-users.

Keywords: blockchain; identity; distributed ledger technology; interoperability

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

PCI Private Certifier Intersection

PCI-All Private Certifier Intersection Validate-All

PCI-Any Private Certifier Intersection Validate-Any

PSI Private Set Intersection

ABS American Bureau of Shipping

B/L Bill of Lading

B2B Business-to-Business

B2C Business-to-Consumer

BFT Byzantine Fault Tolerant

CA Certificate Authority

CMDAC Configuration Management and Data Acceptance Chaincode

CoSi Collective Signing

CSP Cloud Service Providers

DID Decentralized Identifiers

DLT Distributed Ledger Technology

EC Elliptic-curve

IIN Interoperation Identity Network

L/C Letter of Credit

MPC Multi-party Computation

MSP Membership Service Provider

OIN Organization Identity Validator

PMV Participant Membership Validator

PoS Proof of Stake

PoW Proof of Work

PTAN Privacy-Preserving Trust Anchor Negotiation

SPV Simplified Payment Verification

SSI Self-sovreign Identity

STL Simplified TradeLens

SWT Simplified We.Trade

TA Trust Anchor

TFN Trade Finance Network

TLN Trade Logistics Network

VC Verifiable Credentials

VDR Verifiable Data Registry / DID Registry

Symbols

$< m >_{\mathcal{Y}}$ Message m encrypted with key $\mathcal{Y}$

$\mathcal{A}$ Adversary

σ Signature / certificate

$\mathcal{C}$ Certificate space

m Claim

$\mathcal{M}$ Claim space

$\mathcal{O}$ The point at infinity (the identity element) of an EC

$\mathcal{G}$ Generic group with prime order p

$H(.)$ Cryptographic hash function

$\hat{\mathcal{K}}$ Infrastructure contribution proportion

id	Identity corresponding to a certifier
$\mathcal{ID}$	Set of identities corresponding to the certifiers
λ	Security parameter
$\mathbb{C}$	Catalog of a cloud federation
$\mathbb{F}$	Federation
$\mathbb{V}$	VM configuration set
$\mathcal{F}$	Functionality
$\mathcal{K}$	Infrastructure contribution
$\mathcal{N}_i$	DLT network i
$\mathcal{T}$	Trust anchor
CR	Consumer request
C	Consortium member
O	VM offering set
o	VM offering
vm	VM configuration
$[.]_{\mathcal{G}}$	A secret shared value over $\mathcal{G}$
$[.]$	A secret shared value over F_p
$\mathcal{S}$	Simulator
τ	Trusted third party
$\mathbb{T}$	Set of trust anchors
F_p	Field with prime order p
P_i	Party i

Chapter 1

Introduction

Recent advances in blockchain technology have liberated it from the boundary of cryptocurrencies [2] and opened up a whole new domain of decentralized multi-stakeholder applications [3]. On the one hand, robust smart contracts [4], and faster transaction processing capabilities [5] in permissionless blockchains have fostered the development of publicly accessible decentralized applications such as NFT marketplaces [6], lending platforms [7], multi-player games [8], etc. While on the other hand, the development of permissioned blockchain platforms [9] has made distributed ledger technology (DLT) a plug-and-play solution for a bulk of application scenarios where multiple authoritative entities (which do not necessarily trust each other) need to collaborate toward some common goal. For example, in the case of trade and logistics, different businesses form a consortium for operating a business-to-business (B2B) supply chain, where transparency, accountability, and provenance tracking of every transaction is vital. However, the participating businesses do not wish to blindly trust any central authority to maintain this system for obvious uncertainty regarding its trustworthiness and reliability. In such scenarios, permissioned DLT platforms have emerged as the perfect candidate for providing the necessary infrastructure that eliminates the dependency on a central authority with a transparent, tamper-proof system governed collectively by its independent stakeholders. As a result, permissioned DLTs have seen applications in a wide range of use cases such as supply chains [10, 11], energy trading [12], healthcare [13, 14] etc. However, these instances of early adoption of blockchain technology have largely been through the deployment of DLT networks which

are geared towards narrow short-term objectives, and they are unable to scale to a broader multi-blockchain setup with cross-chain communication [15].

The use cases of blockchain technology are not necessarily always limited to a single group of stakeholders. Instead, communication in the form of data and asset transfer between different networks is also required. Consider an example of international trade finance and trade logistics. DLT based Trade finance platforms such as *We.Trade*[1] is a collection of financial institutions such as banks, global credit bureau, and businesses that require access to financial services for international trade. These organizations do benefit from the transparency and auditability features of using a DLT platform. However, situations often arise when such a finance network requires information from the logistics providers in order to enforce a contract, such as a *letter of credit*. A *letter of credit* ensures payment to a seller of some goods, provided that the seller has handed the said goods to the logistics provider and produced a *bill of lading* [16] as a proof. This information about goods being dispatched is usually a part of the logistics network, which is often instantiated as a separate blockchain network (e.g. Tradelens [2]). Therefore, cross-blockchain transfer of data is a required to achieve practical goals which are otherwise not possible in the current fragmented landscape of DLT platforms [15]. The necessity of such DLT interoperability is however not limited to only enterprise scenarios of isolated business networks.

Blockchain applications have largely been of two specific categories, - (i) applications which are meant to be publicly accessible by their users without the requirement of any joining procedure, and (ii) private applications handling business logic that are executed by a group of participants who know each other and explicitly approve any new participants who wish to join. Accordingly, permissionless and permissioned blockchain technologies respectively are applied for these two different types of use cases. In the rest of this thesis, we use the terms permissionless blockchains and public blockchains interchangeably. Similarly, we also denote permissioned blockchains as private blockchains. Notably, in many real-world scenarios, these two categories cannot act independently, and in such cases, both permissioned and permissionless blockchains must work together in harmony. This fact can be realized by simply extending the aforementioned trade-logistics use case exam-

[1] `https://www.ibm.com/case-studies/wetrade-blockchain-fintech-trade-finance/`

[2] `https://www.tradelens.com/` (Discontinued at the beginning of the year 2023)

ple. Any supply chain is typically a multifaceted network with one or more closed business networks collaborating through B2B (business-to-business) transactions, facilitating different steps starting from the procurement of raw materials to manufacturing. Finally, an open consumer network having B2C (business-to-consumer) operations [17] allows the distribution and delivery of goods and services to the end-consumers. While a permissioned DLT platform is suitable for B2B interactions, it cannot accommodate end-consumers because of its closed design. As a result, there is clearly a requirement for the development of an interface between such closed networks and open permissionless networks in order to realize any application that involves B2B as well as B2C interactions.

In order to realize end-to-end DLT based systems that can accommodate closed groups of participants, as well as provide an open interface that is publicly accessible, we need to develop frameworks and protocols for cross-blockchain interoperability. In this direction, we next explore the existing solutions and outline the gaps and challenges that motivated us to conduct our research on blockchain interoperability and identity management.

1.1 Motivation

Blockchain interoperability has been an area of research interest in recent times [18]. The major share of efforts have been towards developing protocols for cross-chain cryptocurrency exchange [19–21]. As a result, public-public blockchain interoperability frameworks have seen a fair amount of adoption in the context of cryptocurrency trading [20], cross-chain smart contract execution [22], payment channel networks [23], etc. Some private-private blockchain interoperability solutions have also been proposed [15] to facilitate inter-consortium B2B collaboration for use cases such as federated model training [24]. Notably, there is still no means of constructing an end-to-end decentralized workflow involving multiple B2B links and ending in one or more B2C networks since such a system will require an interface between the public end-consumers and the private consortium of businesses. Concretely, a public-private blockchain interoperability mechanism needs to be designed that will act as a bridge from an open and public network to a closed permissioned network. Such a public-private interoperability protocol is an essential requirement in any setup involving a closed consortium of entities such as businesses / organizations, which

wants to communicate to entities such as their end-users that are outside the purview of the permissioned network.

Having a B2C bridge through a public-private blockchain interoperability protocol is the first step towards realizing DLT networks of consortiums which need to dispense goods / services to the masses. But at the same time, one consortium might need to collaborate with other consortiums to achieve their business goals. The aforementioned example of international trade finance and logistics is one notable use case scenario. In this regard, there are a few private-private blockchain interoperability frameworks that have been developed [18]. One of the most notable works among them is by Abebe et al. [15], which proposed a mechanism for cross-chain verifiable data transfer across permissioned DLTs. The crust of their contribution lies in the general framework of verifying the validity of data originating in one permissioned DLT from another DLT. In their proposed architecture, data originating from one DLT is attached with a proof and sent to the other DLT network. The proof must reflect the consensus view of the source DLT, and the same should be verifiable by the destination DLT. This mechanism is agnostic of the underlying DLT platforms of the two permissioned networks, and their instantiations used proofs through digital signature-based attestations for Hyperledger Fabric [9] based DLTs. Any private-private blockchain interoperability protocol in some way has to achieve such consensus proof generation and verification mechanism. However, one key requirement to enable such cross-chain proof verification is the availability of identity information (e.g. public keys) of the participants of the counter-party networks for validating attestations (e.g. digital signatures). The existing works take it as an assumption that the identity information of the other network's participants is somehow pre-configured before the start of the interoperation protocol [15]. Notably, configuring member identities across two permissioned DLTs without the help of any central trusted mediator is a challenging problem and is essential to enable private-private blockchain interoperability.

Recently introduced *W3C Recommendations* namely, Decentralized Identifiers (DID) [25], and Verifiable Credentials (VC) [1] provide a foundation for distributed identity management. Using DIDs and VCs as basic building blocks, protocols for cross-blockchain exchange of identities can be constructed. Concretely, any trusted credential exchange is dependent on the availability of a common trust basis. This common trust basis or trust anchor attests to the claim(s) of a credential prover, and the same is trusted by the credential

verifier. As a result, even before any kind of credential (e.g. identity) can be exchanged (e.g. through the VC specification), the two parties must somehow negotiate a common trust anchor. Notably, a naive approach to determine a common trust anchor by revealing the entire list of such trust anchors of a party can have major privacy implications. There are strong reasons why revealing one's complete list of trust anchors might not be in one's interest. Often these trust anchors are well known and influential organizations, and hence are accepted by different parties. These well known trust anchors could include government agencies, political organizations, NGOs, etc., and such affiliations might be sensitive information that could potentially be misused. As a result, there are strong reasons for designing protocols for privacy-preserving negotiation of such trust anchors for facilitating cross-blockchain identity configuration. Further deeper in this direction, even outside the use-case of blockchains, in general any application involving credential exchange through DID and VC specification would benefit from such a privacy-preserving common trust anchor determination mechanism. Such protocols must be compliant with the standard digital signature schemes used to issue VCs such as ECDSA [26], and also be efficient enough to be of practical use.

Based on the aforementioned gaps in the existing blockchain technology landscape, specifically around public-private blockchain interoperability and decentralized identity exchange, we next define a set of objectives that we aim to address in this thesis.

1.2 Objectives of the Thesis

From a bird's-eye view, this thesis intends to design and develop architectures and protocols for enabling an end-to-end decentralized ecosystem of connected permissioned and permissionless distributed ledgers that are capable of interoperating with each other. As outlined in the previous section, there have been prior works that have contributed toward this goal, but nonetheless, there are gaps that need to be addressed. In this section, we detail the specific objectives of the thesis.

1.2.1 Enabling Public-Private Blockchain Interoperability

Emerging blockchain networks such as *IBM Food Trust* [27], *TradeLens* [10], *Marcopolo* [28], etc., use private DLT platforms like *Hyperledger Fabric*, *Corda* [29] etc., to form closed consortiums of businesses. However, a key limitation of such existing private blockchain platforms is the restriction of their applicability within only closed consortiums where data and assets are not required to be communicated outside the network boundary. Thus, *Fabric*, *Corda*, or other existing private blockchains do not support any interface or protocols for interacting with the open network outside, which is crucial for building consortiums of service providers acting together to deliver services to the consumer network. In this objective, we aim to develop a decentralized interface between the private blockchain networks and the open network of end-users (consumers). While doing so, we address the following sub-objectives. (i) A protocol designed for transferring data, such as end-user requests from the open network, into the private blockchain in a secure and verifiable manner. (ii) Collaboration framework through which consortium participants reach consensus on the end-user data and schedule end-user requests for collective service delivery. (iii) Development of a mechanism through which a permissioned network can communicate its data and assets to its end-users, such that the end-users can validate the authenticity of that data and verify its source. (iv) The final sub-objective is to implement the public-private blockchain interoperability layer achieving these three functionalities, and validate its usability.

1.2.2 Identity Exchange across Permissioned Blockchains for Enabling Interoperation

Inter-consortium collaboration depends on private-private blockchain interoperability. Existing private-private blockchain interoperability frameworks are based on exporting the *consensus views* of a source network's participants and making them verifiable through attestations. But proof-by-attestation relies on a network's ability to gain visibility into its counterparty network to let its participants know the identities and certificate chains of the latter's participants. These identities are essential for validating the signatures in proofs. Our objective is to design a pluggable decentralized identity exchange protocol through

which two interoperating networks can set up their counterparty network's identities. Towards this direction, we address the following sub-objectives. (i) Separation of the interoperation protocol dealing with data and asset transfers from the identity exchange mechanism into two different planes. (ii) Designing DLT agnostic identities in the form of SSI for the identity plane and exposing them outside the permissioned network boundary. Additionally, mapping the SSI to the DLT specific identities of the data plane. (iii) Using trust anchors for trusted exchange of identity information across blockchain networks. (iv) Ensuring consensus of foreign network identity information within a DLT. (v) Development of a DLT agnostic implementation of this identity exchange protocol for a real-world use case.

1.2.3 Cross-chain Negotiation of Common Trust Anchors

Trustworthy exchange of credentials, such as identity information, between two parties depends on the availability of a common trust basis. In the context of identity exchange between two permissioned blockchain networks, the credentials used by organizations to prove their real-world identities to foreign networks may be issued by several different trust anchors. But such a credential is useful and applicable only if the trust anchor that issued it is also trusted by organizations in a counterparty network. But revealing one entity's entire trust anchor list poses a serious threat to its privacy, as discussed in the last section. As a result, one of our objectives in this thesis is cross-chain negotiation of common trust anchors. This involves the following sub-objectives. (i) Defining the trust negotiation problem and the required security guarantees through a formal framework. (ii) Exploring the possible solution approaches that may or may not involve the trust anchors themselves. (iii) Designing a protocol for privacy-preserving trust anchor negotiation that is compatible with Verifiable Credentials specification. (iv) Evaluating the performance and security guarantees of the implementations of these solutions.

1.2.4 Determining Common Trusted Credential Issuers

A key goal of Web 3.0 is to remove dependencies on centralized service providers, including identity providers such as Certificate Authorities (CAs) [30]. Emerging decentralized

identity systems are being developed on the foundation laid by DID and VC specifications. But any VC flow involving a credential prover and a credential verifier is dependent on the availability of a common certifier that acts as a trust anchor between them. Therefore, extending from the last objective, we aim to generalize privacy-preserving negotiation of certifiers between a credential holder and a credential validator for any VC presentation scenario. Through this protocol, two parties will be able to determine their common valid trust anchors without revealing any information about the trust anchors which are not common between them. The following sub-objectives must be met towards achieving this goal. (i) Introducing a formal security definition for private certifier intersection. (ii) Designing an efficient secure multi-party computation framework capable of validating digital signatures. (iii) Instantiating a private certifier intersection protocol for practical signature schemes such as ECDSA. (iv) Handling use cases involving multiple claims. Moreover, any such system has to be efficient enough for practical usage while meeting the security goals.

1.3 Contributions of the Thesis

Following the aforementioned objectives, we next summarize the contributions made in this thesis.

1.3.1 Public-Private Blockchain Interoperability for Service Decentralization

We designed and developed a decentralized collaborative architecture called *CollabFed*, which provides a novel decentralized interface between permissioned blockchain networks and permissionless blockchain networks. This enables closed consortiums of businesses to interact with their end-users, which form the B2C side of a supply chain. The primary contributions of *CollabFed* are summarized as follows.

Overview

Through the unique combination of the public DLT and private DLT networks by enabling interoperability between them, *CollabFed* provides the mechanism for trusted and secure data transfer (a) from the end-users to the business consortiums and (b) from the businesses to the end-users. We define the threat model for public-private blockchain interoperability while taking into consideration the byzantine behavior of end-users in the public network, as well as consortium participants in the private network. Furthermore, *CollabFed* handles sybil attacks of end-users and risks of leakage of sensitive consortium information.

The cornerstone of *CollabFed* is its "consensus on consensus" mechanism for propagating the data, such as end-user requests committed on the public blockchain reliably to the private blockchain. This protocol ensures that the consortium network has consensus on (i) end-user requests and data, as well as (ii) the order in which they are received by the consortium. While doing so *CollabFed* ensures both safety and liveness of the interface.

Once an end-user request is processed and the consortium generates a response, that response must be securely transferred back to the requesting user in the public network. We employ the concept of Collective Signing (CoSi) [31] where a set of consortium members collectively sign a response to make the consensus view verifiable outside the private network boundary. Furthermore, *CollabFed* encrypts each response with the public key of the end-user for ensuring privacy.

We implemented *CollabFed* for the use case of cloud federations, where a consortium of cloud service providers provides cloud infrastructure in the form of virtual machines to its end-users. The implementation used a combination of Ethereum [4] for the public network , and Fabric [9] and Burrow [32] for the private network.

Summary of Results

We evaluated *CollabFed* for a decentralized cloud federation use case using three cloud providers. We also performed scalability tests for up to 32 emulated consortium participants. Following are some of the key observations from the evaluation.

1. The characterization of the public-facing interface using different Ethereum test networks, namely, proof-of-work based Ropsten and proof-of-authority based Rinkeby network, reveal that the Rinkeby network has significantly less transaction commitment latency.

2. The transaction processing latency of the private blockchain is however significantly less compared to that of the public blockchain. Both Fabric and Burrow based implementations are capable of processing 64 concurrent end-user requests in around 5 seconds in the "consensus on consensus" protocol.

3. Scalability experiments using 32 consortium participants in a *Mininet* [33] emulation shows that the CoSi inspired signature collection takes around 3.5 seconds even for a high inter-participant latency of 400ms.

4. Overall, *CollabFed* has acceptable latency overhead in use cases such as cloud federations, where the end-user requests are relatively infrequent and inherently take a long time to serve compared to other applications such as e-commerce. This is often an acceptable trade-off for the decentralization guarantees that *CollabFed* offers.

1.3.2 Decentralized Cross-Network Identity Interoperation

We proposed architecture and protocols for the exchange of identities across permissioned blockchain networks based on the following design principles: (a) The architecture must be DLT platform agnostic and should be applicable to any permissioned blockchain technology. (b) The networks and their participants should be free to choose identity registries and providers. (c) Networks must retain their autonomy in exposing or not exposing their identities. (d) The solution implementation should not require changes to the existing DLT platforms.

Overview

Our proposed decentralized identity management architecture separates the identity interoperation functionality into a separate plane from the data interoperation plane. However,

since permissioned blockchain network identities (e.g. root CA certificates in Fabric) have no manifestation outside the network boundary, the first step is to design network agnostic credentials. For this purpose, our architecture uses *Decentralized Identifiers* (DID) and *Verifiable Credentials* (VC) to provide SSI to the participants that can be exposed outside the private DLT.

For facilitating verifiable exchange of identity and credentials, we introduced "Interoperation Identity Network" (IIN), which consists of a DID registry [25], and a pool of credential issuers (trust anchors). The DID registry is built on a public ledger to avoid centralization. The IIN trust anchors issue credentials attesting to the identity of consortium participants, as well as the validity of their membership in a particular DLT network. A pool of such IINs together forms the *distributed identity infrastructure* that provides the trust basis for cross-network verifiable identity exchange.

Based on this identity plane architecture, we designed a protocol for cross-network identity interoperation which involves two broad phases: (a) network independent SSI configuration, and (b) identity exchange and validation. In the first phase, each private blockchain network participant creates its SSI in the form of a DID, and registers the same in some IIN's DID registry of its choice. Then it gets its identity and consortium membership claims attested by some trust anchor which issues necessary verifiable credentials. Once network participants have their respective SSI and VCs configured, the protocol can proceed to the identity exchange phase. Through the identity exchange phase of the protocol, a permissioned network configures the identity, membership, as well as DLT-specific data plane credentials (e.g. Fabric certificates) of the participants of a counterparty network with which it wishes to interoperate. This is achieved by fetching the DID of each participant, validating their identity and membership VCs, and fetching their DLT-specific credentials, which are self-signed using their DID. Finally, to ensure consensus of the network on a counterparty network's identity, we design a flow through which the participants' endorsements are collected on it. The identity configuration of a foreign network is acceptable only if it has sufficient endorsements.

We implemented the identity plane architecture using Hyperledger Indy [34] for the DID registry, and Hyperledger Aries [35] for VCs. We implemented the identity exchange protocols for two Hyperledger Fabric networks, enabling them to interoperate in the data

plane without manual identity pre-configuration.

Summary of Results

We demonstrated the proof-of-concept implementation of our protocol by extending the two-network use case in Abebe et al. [15]. We started with scaled-down versions of the trade logistics network, TradeLens, and the trade finance network, We.Trade. Our protocol replaces the naive implementation in Abebe et al., where organizational identities and root and intermediate certificates were fetched out-of-band manually. The key results and analysis are as follows.

1. Through our protocol, the simplified versions of TradeLens and We.Trade. networks could successfully verify and configure the counterparty network's identities without manual intervention. Furthermore, the data plane operations are kept unaltered and work seamlessly.
2. Our proof-of-concept implementation required minimal change, namely, only one additional smart contract in the existing deployed blockchain networks. This demonstrates the extensibility of our solution for any permissioned blockchain platform.
3. We analyzed the security guarantees of our architecture and protocol in terms of confidentiality, integrity, and availability.

1.3.3 Cross-chain Negotiation of Common Trust Anchors

The identity exchange protocol introduced above provides a mechanism for permissioned DLTs to validate and configure foreign network identities. However, the credentials used by organizations to prove their real-world identities and network memberships may be issued by several different trust anchors (TAs) associated with different registries. But such a credential is applicable only if the TA that issued it is also trusted by organizations in a counterparty network. Determining such common TAs without revealing the other TAs is

important to protect the privacy of the parties. We study this as "Privacy-Preserving Trust Anchor Negotiation" (PTAN) problem and propose two genres of solutions for the same.

Overview

We introduce the definition of the PTAN problem in the universal composability (UC) framework, and provide solutions for it while considering malicious adversary model. Notably, trust anchors in practice are often well-known entities such as government organizations, large businesses, etc. Hence, in our threat model, we consider that the adversary has access to the universal set of all possible TAs. Consequently, we point out that black box usage of existing cryptographic constructions such as private set intersection (PSI) are not enough to achieve PTAN.

We first provide a simple solution for common TA negotiation with the help of the trust anchors' active participation. At a high level, in this protocol, the parties contact their respective TAs initiating a negotiation. Only when some TA receives negotiation requests simultaneously from both the parties, it responds to them positively, and thus reveals itself as a common anchor. This naive approach however contradicts one key goal of verifiable credentials which promises that possessing a VC excuses a TA from being involved in the VC presentation process.

Avoiding the involvement of the TAs themselves requires the parties to compute their common TAs without the help of any other mediator. We present a secure multi-party computation (MPC) based protocol that achieves this for credentials with ElGamal signatures. Through an optimization where the r component of an ElGamal signature (r, s) is made public, our protocol can efficiently verify the VC signatures within the MPC, while revealing neither the signature nor the signer. After validating the VCs, the protocol computes the intersection of the valid input trust anchors and outputs one or more common TAs as per the requirement.

Summary of Results

We implemented our proposed MPC based PTAN protocol using the MP-SPDZ [36] framework and evaluated its performance and data communication overheads. The noteworthy results are summarized as follows.

1. With the varying size of input sets from both parties, our proposed MPC based PTAN protocol shows a linear increase in the execution time. Using 64-bit primes for the MPC protocol, even for large input sizes of 120 from each party, it takes less than 25 seconds.

2. The complexity of the protocol is however dependent on the size of the smaller input set size. This is made evident by keeping one party's input set size constant at 20, and varying the other from 20 to 120. In this scenario, the execution time of MPC based PTAN remains relatively unchanged at less than 20 seconds (using 128 bit primes).

3. Just like execution latency, the protocol shows similar trends in the data communication overhead. There is a linear increase in the volume of data communicated with the increase in the number of input TAs from each party. This data communication overhead is also proportional to the size of the smaller input set.

4. In addition to the performance and overhead analysis of the PTAN protocol, we also conduct a detailed security analysis and formally prove that it is secure against malicious adversaries.

1.3.4 Private Certifier Intersection

Evolving from the aforementioned problem of common trust anchor negotiation across permissioned blockchain networks, we try to generalize the problem of finding the common certifiers applicable in any VC presentation use case. We try to answer the question "*Can parties owning certificates efficiently identify a common set of certifiers without leaking anything else?*". We define this problem of finding common certifiers without revealing any information about the other certifiers as "Private Certifier Intersection" (PCI).

Overview

Although seemingly similar to private set intersection (PSI), PCI is fundamentally different since it involves validating the certifiers by verifying the credentials issued by them before determining the common certifiers. It turns out that in the setting of semi-honest corruptions (i.e., when the participating parties behave honestly as prescribed in the protocol), one can easily achieve a secure PCI protocol by using any secure PSI protocol in a black-box way. However, this does not hold for malicious corruption, and we develop a PCI protocol for this type of adversary.

We formalize the security guarantees expected of a (multi-party) PCI protocol using the simplified universal composability (SUC) framework, and define two variants of PCI: (a) PCI-Any-DC where a certifier is valid if the party has a valid certificate from it attesting to *any* one of its public claims, and (b) PCI-All where a certifier is valid if the party has a valid certificate from it attesting to *all* its public claims.

The centerpiece of our contribution is a novel secret-sharing based MPC framework that is tuned for elliptic curve pairings. We build upon the SPDZ secret-sharing based MPC protocol [37] to achieve the first secret-sharing based MPC framework that seamlessly supports elliptic curve pairing operations as fundamental gate-level building-blocks with malicious security against a dishonest majority of adversarial parties.

We then use this framework to build two efficient and provably secure PCI protocols, one for PCI-Any-DC that uses ECDSA signatures for VCs, and the other for PCI-All using BLS signatures. We make several optimizations to both ECDSA PCI-Any-DC and BLS PCI-All protocols to make them practically efficient. Through BLS-based signature-aggregation techniques, the efficiency of PCI-All is brought at par with PCI-Any-DC.

Summary of Results

We extended MP-SPDZ [36] to implement our proposed secret-sharing framework supporting elliptic curve operations, including bilinear pairings. Using this as a foundation, we implemented both ECDSA PCI-Any-DC and BLS PCI-All protocols. We present a formal

security analysis of the protocols. After a detailed analysis of the performance of individual components of our MPC framework, we conduct an end-to-end performance evaluation of the protocols in realistic setups by placing parties in three geographic regions across two continents. The key observations from this evaluation are as follows:

1. In a WAN setup with inter-party RTT latency of $\sim 62ms$, both ECDSA and BLS implementations of PCI-Any-DC take less than 150 seconds for 100 inputs from each party. Notably, we use consumer hardware with only 8 core CPU and 8 GB RAM for these experiments. Using more powerful hardware, up to $\sim 71\%$ reduction in end-to-end execution time is observed.

2. Both memory and data communication overheads of PCI-Any-DC increase with the increasing size of the input sets. For 100 inputs from each party, both ECDSA and BLS PCI-Any-DC require around 1.6 GB of data to be communicated. The memory requirement of the BLS variant is more compared to that of the ECDSA variant of PCI-Any-DC.

3. The BLS PCI-All protocol is more efficient than an ECDSA PCI-All protocol that iteratively validates all the claims for a certifier. An increase in end-to-end latency of ECDSA PCI-All is observed with an increasing number of claims and a fixed number of certifiers. On the other hand, the latency of BLS PCI-All stays almost unchanged with the increasing number of claims.

4. ECDSA PCI-All requires more data to be communicated for more number of claims from the parties, whereas for a fixed number of certifiers, BLS PCI-All has identical data communication overhead for any number of claims. The memory consumption of ECDSA PCI-All is considerably less compared to BLS PCI-All, still, it shows an increasing trend with increasing number of claims.

1.4 Organization of the Thesis

In this section, we briefly describe the organization of the thesis.

- **Chapter 2** provides a background on the recent advancements around blockchains, smart contracts, and decentralized identity management standards. This is followed by a detailed literature survey of blockchain interoperability encompassing public-public, private-private, as well as public-private interoperability. Finally we examine the existing tools and techniques which are related to negotiation of common trust anchors across blockchain networks.
- **Chapter 3** presents *CollabFed*, a public-private blockchain interoperability framework for service providing permissioned consortium networks.
- **Chapter 4** presents a decentralized architecture and protocol for cross-network identity configuration across permissioned DLTs, which is an essential requirement for enabling private-private blockchain interoperability.
- **Chapter 5** deals with the design and development of protocols for privacy-preserving negotiation of common trust anchors across blockchain networks.
- **Chapter 6** introduces the problem of private certifier intersection (PCI), and provides solutions for two variants of PCI. The solutions are based on a novel MPC framework that supports elliptic curve pairings.
- **Chapter 7** concludes the thesis by summarizing the primary contributions and suggesting potential future directions.

Chapter 2

Background and Related Work

In this chapter, we first briefly introduce blockchains and smart contracts, followed by some other building blocks related to decentralized identity management. Looking ahead, these concepts will be extensively used in the later chapters for building interoperability architectures across different types of blockchains. Following this, we discuss the existing literature on blockchain interoperability and trust anchor negotiation in detail. While doing so, we outline the key challenges of interoperability and the gaps in the existing solutions for decentralized trust negotiation.

2.1 Blockchain

Before delving into the aspects of blockchain interoperability which is the crust of this thesis, we first introduce the concept of blockchain which is also generally termed as distributed ledger technology (DLT) [3, 38].

The blockchain technology has been made popular by cryptocurrencies after the rise of Bitcoin [2]. It may be described as "the decentralized transparent ledger with the transaction records – the database that is shared by all network nodes, updated by miners, monitored by everyone, and owned and controlled by no one." [39]. At its core, a blockchain is a replicated append only ledger whose transactions are governed by byzantine fault tolerant

consensus protocols [40]. With the introduction of smart contracts, blockchains were no longer limited to applications in cryptocurrencies, but open to any decentralized computation [3]. This made them comparable to the classical *state-machine replication* [41], with applications in a wide range of domains ranging from supply chains [11], IoT [42–44], data-sharing systems [24, 45–47], healthcare [13, 14], transportation [48], energy trading [12], etc. As a rule of thumb, we can say that the blockchain technology is applicable in any scenario where there are multiple stakeholders who do not completely trust each other, yet they together want to run a shared system without relying on any central trusted authority. In fact, the adoption of blockchain has thus far been limited to groups of stakeholders working towards some common goal, resulting in the instantiation of specialized isolated DLT systems with no means of communicating with one another [15]; - a problem which we try to address in this thesis.

Based on the consensus architecture and the membership policy deciding who can participate, the blockchain networks can be categorized into (a) permissioned (private) blockchains (b) permissionless (public) blockchains. The general characteristics and specific instantiations of each these categories is discussed in the following subsections.

2.1.1 Permissionless blockchain

In a permissionless or public blockchain network, the participation is open to everyone. Anyone can join the network without any identity verification process. As a result, a participant is not aware of the identities of every other participant in the network. The most popular consensus protocol for this kind of DLT is "proof of work" (PoW) introduced for Bitcoin [2]. However, due to the poor transaction throughput and high resource consumption by PoW [49], many alternate consensus protocols and blockchain platforms have since been proposed, such as Proof of Stake [50], Bitcoin-ng [51], Byzcoin [52], Algorand [5] etc. These have slightly different system models and safety guarantees, but they have one property in common i.e. *open-membership*. As a by-product of the open-membership and the consensus protocols, permissionless networks gain a very important property – *public verifiability*, allowing anyone to verify the correctness of the state of the system [3, 53].

Proof of Work (PoW) consensus based blockchains like Bitcoin, Litecoin, Ethereum

(till September 15, 2022) etc., account for the majority of market capitalization in digital cryptocurrencies [1].

If the block mining process depended on the number of votes to determine the next block, then having "one-account-one-vote" would be a broken system since anyone could allocate as many accounts as one wanted in a permissionless blockchain. This problem is known as the Sybil attack [54], where a single participant can have multiple identities.

To counter this, in essence, PoW tries to achieve "one-CPU-one-vote". The PoW consensus protocol requires miners to expend computing resources in the form of CPU time to mine new blocks [2].

Concretely, the blocks in the blockchain have a *nonce* field which can contain arbitrary data. The miners are required to set the nonce to some value such that the cryptographic hash of the block (such as SHA-256) have a certain number of leading zeros. The CPU time, and hence energy expended to mine a block grows exponentially with the number of leading zeros required to mine a block. This parameter is known as the proof-of-work difficulty, and it is determined by a moving average targeting an average block mining rate. Once a block is mined, it cannot be changed without redoing the work. Moreover, as more blocks are chained after it, the work to change a block includes 're-mining' all the blocks after it. An important part of the PoW protocol is that honest miners would always work on the longest chain. Therefore, any attacker wanting to change a block must first change the target block, then mine new blocks after it till this new chain overtakes the longest existing chain.

Proof of Work's "one-CPU-one-vote" strategy is certainly a practical approach to countering Sybil attack. However, it has certain economic implications. The miners who process the transactions in a block to keep the blockchain operating need to expend CPU resources, and hence energy - which has monetary value. Therefore, there has to be some economic incentive for the miners to keep the blockchain operational. In addition, there must be sufficient economic deterrent for any attacker that will potentially prevent it from deploying enough CPU resources to alter the chain. Miners are incentivized in two ways - (i) mining rewards, and (ii) transaction fees. Mining reward or block reward is the constant amount of

[1] `https://coinmarketcap.com/`

coins (cryptocurrency) given to a miner for successfully mining one block. This provides a steady flow of new coins in the system. The process is thus analogous to gold miners spending resources to add new gold into circulation. The mining reward is revised and reduced every constant number of blocks to emulate the scarcity of supply of coins. Eventually, the mining reward runs out and the miners are left only with the second type of reward, that is transaction fees. The miners are free to choose any transactions to be included in a block, and thus they choose the transactions that offer the most transaction fees. Once mined, the transaction fees of a transaction goes to the miner. Miners gain a monetary profit when the monetary value of the cryptocurrency coins earned through mining exceeds the cost of energy expended in the mining process. This in turn acts as a deterrent to attackers because if an attacker somehow manages to get more than 50% of the CPU resources in the network, then it can make more economic gains by honest mining than by attacking a particular block.

The safety property of PoW holds with the assumption that no entity (single or colluded) has more than 50% of the processing power in the network [49]. However, there have been other attacks such as "selfish mining" [55], "eclipse attacks" [56], etc. which can further compromise the safety of the network. In terms of liveness, PoW mines all the transactions eventually as long as the network is partially synchronous and under the control of honest miners (safety condition). The transaction fees also govern how fast a transaction will be committed [57]. The rate of producing new blocks has a trade-off with the safety of the blockchain, with stale block rate of 0.41% in Bitcoin compared to 6.8% in Ethereum due to the faster confirmation time in the later [49]. As a result, the transaction rates of these blockchains are slower ($<$ 100 transactions per second) compared to other alternatives [52].

Proof of Stake (PoS) consensus protocols have been developed to reduce the resource consumption required for the consensus process in case of PoW, and they also offer better transaction confirmation latency [5, 50]. In essence, the "one-CPU-one-vote" idea of Proof of Work is replaced with "one-coin-one-vote" in Proof of Stake, where a participant's investments in the blockchain (cryptocurrency) determine its probability of successfully mining a block. More precisely, in PoS, a *leader* is selected among the stakeholders of the blockchain, and that leader is responsible for proposing and validating (mining) new blocks. The probability of a participant becoming a leader is proportional to its stake, that

is the amount of cryptocurrency coins of the concerned blockchain. In some blockchain networks, a minimum stake is required to be eligible to participate as a leader. For example, in Ethereum PoS, a user must deposit 32 ETH into a deposit smart contract in order to participate as a leader. The leader for a time slot is usually selected through a random function such as the Verifiable Random Functions (VRF) in Algorand [5] while giving weights proportional to the stake of the participants. Once selected, a leader must participate in the block proposal and block validation process.

Just like in Proof of Work, in Proof of Stake protocols, there are economic incentives that drive the participation of miners and deter malicious behaviour. Firstly, the stake of the miners acts as collateral that can be destroyed if the leader (validator) tries to attack the system or refuses to participate in the mining process, implying denial of service. The two most common attacks which a leader can try are (i) proposing different blocks to different subsets of the blockchain network in an attempt to create a fork, and (ii) attesting two or more different proposed blocks, thus announcing contradictory attestations. Such attacks can potentially create forks with two competing chains, in which case the "richest chain" wins [3]. Here the "richest chain" indicates the chain that is attested by the highest amount of stake. Once any such malicious behaviour of a leader or validator is detected, a part of the stake of that participant is destroyed as a penalty - thus enforcing honest behaviour. Similar penalty is also imposed for lazy participants who do not participate in validation process in a timely manner, thus deterring denial of service attacks. Similar to PoW, the honest miners also receive transaction fees and mining rewards for their service in proposing and validating blocks. Although this incentive keeps the blockchain operational, there are concerns about the "rich-get-richer" with Proof of Stake systems [3].

The condition under which PoS blockchains maintain the safety property is that more than two thirds ($> \frac{2}{3}$) of the stake (for example cryptocurrency) is owned by the honest users. For ensuring the liveness property, these blockchains depend on certain practical assumptions like network reachability, strong network synchrony [5], elected committee members participate in the consensus process, and stakeholders do not remain offline for a long period of time [50]. The transaction throughput and latency of PoS blockchains are much improved when compared to the PoW ones, with Algorand claiming $125\times$ throughput of Bitcoin [5].

BFT Based protocols have also recently gained much attention with Bitcoin-NG [51] and Byzcoin [52]. These blockchains use traditional BFT protocols such as PBFT [58] in order to achieve consensus in a close committee which is elected and changed over time through PoW. Thus, PoW is used only for electing committees, while the transaction blocks can be generated quickly by the committees using BFT protocols. The safety assumption of such consensus protocols is that the byzantine nodes should have less than $\frac{1}{4}$ of the system's total hash power at any time. For liveness, they rely on the weak synchrony property of the network [52]. Because of the decoupling of committee/leader election using PoW and mining blocks using BFT protocols, these protocols achieve a very high throughput (around 1000 transactions per second in Byzcoin).

Apart from PoW, PoS and BFT based protocols there are other consensus protocols such as Proof-of-Useful-Work [59], Proof-of-Elapsed-Time [60] etc. are also discussed in the literature.

Irrespective of the performance gains through efficient consensus protocols [5], permissionless DLTs still have the issue of open membership allowing anyone to participate, and public access allowing anyone to read the ledger that might be unwanted in cases such as in enterprise scenarios [9]. As a result, a separate class of DLTs were introduced in the form of permissioned blockchains which focus specifically on privacy of the ledger and restricted membership.

2.1.2 Permissioned blockchain

In contrast to the open-membership of permissionless blockchains, permissioned blockchains require the participants to go through a joining protocol, and as a result are closed systems where each participant knows the identity of every other participant. Permissioned DLT "provides a way to secure the interactions among a group of entities that have a common goal but which do not fully trust each other" [9]. Thus, although the participants may be faulty or malicious exhibiting byzantine behavior, they cannot create or introduce new members into the network arbitrarily (sybil attack [54]) to gain an advantage. Many permissioned blockchain platforms use traditional byzantine fault tolerant (BFT) consensus like PBFT [58, 61], BFT-SMaRt [62], etc.; while these lack the *public verifiability* property

due to closed membership, they gain in terms of privacy and performance. The choice of the permissioned DLT primarily on the consensus protocol used by it. Different consensus protocols affect the conditions under which the blockchain ensures safety and liveness in the presence of Byzantine faults. It also impacts the latency of processing each transaction affecting the performance of the overall system. We discuss some of the popular permissioned blockchain platforms and their safety and liveliness assumptions next.

Hyperledger Fabric supports multiple consensus protocols as a pluggable component of the ordering service [9]. This includes crash fault tolerant orderer or a Byzantine fault tolerant order based on BFT-SMaRt protocol [63]. The safety property of BFT-SMaRt holds under the assumption that the number of faulty participants is less than one third ($< \frac{1}{3}$) of the total number of participants in the blockchain. For liveness, BFT-SMaRt requires that the network satisfies eventually synchronous property [62]. Hyperledger fabric follows execute-order flow for transactions and reports transaction throughput of more than 2000 per second [63].

Hyperledger Burrow is a private blockchain platform based on the Tendermint consensus protocol [64]. Similar to PBFT and BFT-SMaRt, Tendermint protocol's safety property requires that less than one third of the total participants in the system are Byzantine faulty. If the safety condition is satisfied, then with the assumption that the network has partial synchrony property, Tendermint also satisfies liveness.

Hyperledger Iroha is a distributed ledger for decentralized identity, and it uses YAC consensus protocol [65] which ensure safety and liveness under assumpations similar to PBFT [58].

Hyperledger Indy uses the Plenum consensus protocol[2], which is an implementation of RBFT [66]. RBFT is based on the PBFT protocol, with parallel PBFT flows for more robustness. It has the same assumption for safety property that less than one third of total nodes in the system are Byzantine faulty. For liveness it relies on the assumption of an asynchronous network with synchronous intervals, during which messages are delivered within an unknown bounded delay. With a lab network setup of 8 nodes, RBFT claims throughput of upto 5000 requests per second of size 4KB each. We note here that Indy is a public per-

[2] `https://github.com/hyperledger/indy-plenum/wiki`

missioned ledger implying the ledger data can be read by anyone, but the ability to invoke transactions in order to update the ledger is restricted to the members. Indy is primarily used for decentralized identity management which we will discuss in detail in Section 2.3.

Apart from the aforementioned DLT platforms, new BFT protocols are being developed which are pushing the frontiers of performance of the existing ones that can be incorporated to build better permissioned DLTs. *Honey badger of BFT protocols* [67], *BEAT* [68], and *HotStuff* [69] are some of the recent BFT protocols that are capable of achieving throughput of more than 15000 transactions per second [68] in LAN setting. A more comprehensive study of different consensus protocols for DLTs can be found at the survey by Xiao et al. [70]. We now shift our focus to smart contracts which actually makes DLTs useful applicable to a wide range of applications.

2.2 Smart Contracts

The idea of smart contracts goes back to 1997, which at its core is about enforcing and executing contracts with the help of software and hardware based protocols [71]. Interestingly, the combination of smart contracts and distributed ledgers allows us to eliminate the requirement of any trusted enforcing entity and build a decentralized enforcement system for the contracts.

The fundamental functionality that blockchains provide is *agreement* or *consensus* among its participants. This consensus can be on a value, or on a set of instructions. Concretely, smart contracts are programs on which the participants of a blockchain network have consensus. Additionally, each execution of the smart contract must also be agreed upon by the participants [72]. Thus, smart contracts have to be executed by all (or majority depending on the consensus requirement) the participants in the network taking part in the consensus process. The consensus criteria of smart contract execution brings two new considerations –

(a) The exectution of the contract must be deterministic (so that consensus can be reached).

(b) If the contract acts on a state or value, then the blockchain must have consensus on that state or value too.

There are many smart contract execution platforms including public Ethereum's quasi-Turing complete *Ethereum Virtual Machine* (EVM) [4], and private Fabric's containerized smart contract engine which is decoupled from the consensus protocol [9]. Although fundamentally the execution of smart contracts are based on consensus on each execution result, there are two major different flavors of the flow of smart contract execution.

(i) **Order-Execute**: This is the traditional flow used in most blockchains like Ethereum, Hyperledger Burrow, etc., which involves three steps – (i) reach to the consensus on the set of contract execution transactions and also their ordering, (ii) execute each of those transactions sequentially and deterministically, and (iii) update the state of each contract on the blockchain ledger according to their execution result.

(ii) **Execute-Order**: Here, the transactions are first simulated (executed) to find their results and corresponding change in the current blockchain state. Then these transaction results are ordered and the consensus is reached on them. Finally, the results are applied on the current state while rejecting conflicting transactions [9]. We discuss about conflicting transactions with an example later in this section.

The execute-order method of processing smart contracts can have several advantages including parallel execution and ability to process non-deterministic transactions with no safety violation [9]. However, it can have performance limitations due to failed conflicting transactions which are executed in parallel and then ordered.

We now explain the flow of execution of a smart contract in a DLT using an example. Consider the simple counting contract as shown in Algorithm 1, which maintains a counter state from 0 to 99. The count can be changed by incrementing it through *CallIncrement*, or doubling it through *CallDouble* procedures.

The first step is to deploy or install and initialize the contract in the blockchain. This step involves agreement on (a) the contract program (the instructions), as well as (b) on the initial state of the contract, that is the *count* variable in this case. This initial state is decided

Algorithm 1: Counting Contract

```
1: State Variable: count
2: procedure INIT                                   ▷ Contract deployment
3:     count ← 0
4: procedure CALLINCREMENT                          ▷ Contract execution
5:     count ← count + 1  mod 100
6: procedure CALLDOUBLE                             ▷ Contract execution
7:     count ← count × 2  mod 100
```

through the *Init* procedure. Thus the blockchain participants agree on the initial value of *count* variable to 0.

After deployment, any participant can execute the contract. Such executions are done through *transactions*, each of which denote that a participant is executing the contract once. Transactions include which procedure of the contract is being called, by which participant or contract, and also the arguments passed by the caller. Later we discuss what happens if two participants fire two transactions simultaneously. Let the two transactions be of different types, one *CallIncrement* ($\mathcal{I}$) and another *CallDouble* ($\mathcal{D}$).

Using order-execute flow: First the transactions and their order are agreed upon through the consensus process. The order can depend on various parameters such as transaction fees in Bitcoin and Ethereum. Thus, the order can be $\mathcal{I}, \mathcal{D}$ or $\mathcal{D}, \mathcal{I}$. After consensus, the transactions are executed by each participant sequentially, resulting in the new *count* value to be 2 or 1 respectively in the two orderings. Finally the state of the blockchain, which is the current value of the *count* variable, is updated with this new value. The subsequent transaction executions are applied on this updated state.

Using execute-order flow: The transactions are first executed by the participant on the current state (say version v_{t-1}) to find the result (updated state v_t). This updated state, along with the version number of the last state is sent out as the transaction for consensus. Thus, if $\mathcal{I}$ is executed first, then the value of $count$ is calculated as 1, and this is sent for consensus and ordering. After the consensus is reached the state is updated with $count = 1$ $(version\ v_t)$. After this process is complete, if $\mathcal{D}$ is executed, then it acts on the updated state ($count = 1$ in v_t) and after the execution finishes, the new state becomes $count = 2$ in version v_{t+1}. Similarly, if first $\mathcal{D}$ and then $\mathcal{I}$ were executed, then the final

state would have been $count = 1$.

The problem with *execute-order flow* appears when both the transactions are simulated in parallel by the two participants to find the updated state before sending it for consensus. On the initial state of $count = 0$ in v_{t-1}, simulating $\mathcal{I}$ results in $count = 1$, while $\mathcal{D}$ results in $count = 0$. Then both of these updated states are sent for consensus. Simply applying these transactions one after another can clearly result in incorrect state. For example, if the order is agreed as $\mathcal{I}, \mathcal{D}$, the correct resulting state should be $count = 2$. Whereas, by applying $count = 1$ first and then $count = 0$ next from the updated states of the transactions, the final state becomes $count = 0$, which is incorrect. Therefore, in order to prevent such conflicting transactions acting on the same state, the version of the state on which the transaction acts is checked. This is called *Multiversion concurrency control* [73], used in databases. Thus the two (or more) parallel transactions both simulated on v_t are conflicting, and thus only one of them can be committed and the rest will fail. These failed transactions will need to be executed again on the updated state.

Later in Chapter 3 we will observe how the difference in execution methods of smart contracts results in different system performance. Having an overview of blockchain and smart contracts, next we delve into the concepts of decentralized identifiers and verifiable credentials that play a key role towards our goal of blockchain interoperability.

2.3 Decentralized Identifiers and Credentials

The traditional internet relies on centralized authorities (CAs) for identity management which forms the basis of trust and security [74]. Web browsers and operating systems come with a list of trusted CAs [30], which are able to issue certificates attesting the identity of online service providers. It is often difficult for a new organization to introduce itself into that privileged list [75]. Some of these organizations reduce the centralization by bringing in a federated structure whereby they permit their subsidiaries or sub-entities to issue certificates [76]. However, at the root, centralization of trust still exists. In addition, the method of distribution of the trusted CA list often delays the introduction of new certifiers [75, 77]. Although user identification and authentication do not rely on this CA

infrastructure, centralization of user's identity on the internet still exists with the service providers issuing and controlling the identities on their users' behalf [78].

Recent developments towards decentralization in Web 3.0 aim to depart from the traditional web's dependencies on centralized service providers. A prominent feature of Web 3.0 is the trend of using self-sovereign identities (SSI) [79]. Decentralized identifiers (DIDs) [25] and Verifiable Credentials (VCs) [1] are *W3C Recommendations* that lay down a set of specifications which enable parties to own and control their identities.

1. **Decentralized Identifiers (DIDs)** is a *W3C Recommendation* [25] for self-sovereign identity (SSI). A DID is a URI that resolves to a DID document that contains information about its subject, such as aliases and pseudonyms. It can contain public keys to authenticate the subject's signature and service endpoints for communication with the subject to obtain verifiable credentials (described in detail next). A DID subject can be any entity ranging from individuals, organizations, consortiums, to objects and things (e.g. IoT devices). We note here that a DID or its corresponding DID document on its own is not associated with any real-world identity. A DID only allows its controller (which can be the subject itself) to authenticate itself, implying that it has control of the DID through *authentication verification methods* [25]. Only a controller is able to modify a DID document. But, in order for a DID controller to prove some claim about itself (such as its identity) to another party, that claim has to be attested by a certifier which is trusted by the verifying party. We call such a certifier a "trust anchor" that provides a trust basis between the prover and the verifier. A mechanism through which such claim certificaton and verification can be carried out is verifiable credentials.

2. **Verifiable Credentials (VC)** is a *W3C Recommendation* [1] which provides a digital representation for credentials (just like physical credentials) where the issuer is different from the subject to which the credential is issued, and the credential document is cryptographically verifiable. VCs ensure that merely possessing a credential document does not allow it to be presented and verified. Instead, only the VC's subject to which the credential is issued is able to present it through Verifiable Presentations (VPs). In a typical VC and VP workflow (Figure 2.1), an issuer attests to some claim about a subject by issuing a VC to it. The subject, in this instance, becomes the

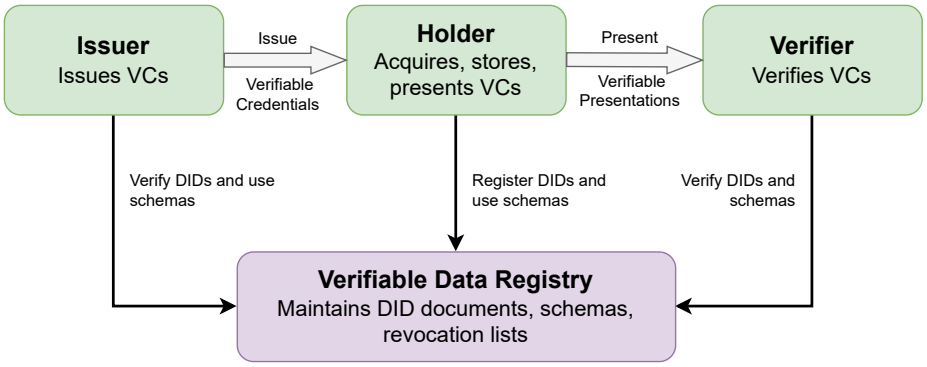


Figure 2.1 A typical Verifiable Credential and Verifiable Presentation workflow that utilizes DID documents and schema maintained in a Verifiable Data Registry [1]

holder of the VC and is able to present the same to verifiers. We note here that no other entity (including the original issuer) can present the VC on behalf of the holder since the VP has to be signed by the holder, and the credential itself contains the public key (or DID) of the holder against which the VP can be verified. A VP can have multiple claims attested by multiple same or different issuers, including the subject itself (self certified credential). A verifier of a VP is convinced about the validity of a subject's claim only if it also trusts the issuer that issued the VC. We elaborate on this fact through an example:

Suppose a person wants to prove his/her COVID-19 vaccination certificate to a verifier. There are multiple COVID-19 vaccines, and say the person is vaccinated using two separate vaccines. As a result, the person has two different VCs attesting to his/her claim of "vaccinated for COVID-19" status, say one from issuer V1, and another from V2. Depending on which vaccine the verifier *also* trusts, the person has to present either the VC from V1 or V2. Looking ahead, in Chapter 5 and Chapter 6, we will develop a mechanism to find such common trusted certifiers between the VC holder and the verifier in a privacy-preserving way.

3. **Distributed Verifiable Data Registry (VDR)** provides the necessary infrastructure that supports the DID and VC ecosystem (Figure 2.1). Concretely, the primary functionality of VDR is to store and resolve DID documents against the DID URIs. Further, for resilience and trust, VDRs are usually built on distributed ledger principles. VDRs authenticate the controller(s) of a DID and only allow a controller to modify a DID document. In addition to maintaining and providing access to DID documents,

VDRs store and help in the distribution of several artifacts that are required in a typical VC workflow. One such artifact is a *verifiable credential schema*, which specifies the structure of a certain type of VC. For example, a university degree certificate schema will usually have details of students, such as their grades. Any issuing entity can register its own schema, and a pointer to the schema is embedded within the VCs. During the VP verification process, a verifier fetches the schema in order to get the metadata and structure that a VC follows.

Hyperledger Indy [34] is a prime example of such a decentralized identity management system built on a shared ledger maintained by a pool of members. An *Indy* network is a public permissioned ledger allowing open queries but restricting data publishing capability to designated *stewards* and *trustees*, who can create and modify *verinyms*. Verinyms refer to DIDs that are associated with legal (or real-world) identities of DID owners. Thus, in addition to providing the usual functionalities of a VDR, Indy also allows privileged participants (*stewards*, *trustees*) to attest to the real-world identity claims of DIDs within the registry. Indy also maintains schemas, public keys for authentication, and revocation information, on its ledger through consensus [34, 66], enabling trustworthy validation of verifiable credentials. In Chapter 4 we use Indy as a building block for identity exchange across different permissioned blockchain networks.

4. **Identity and Credential Messaging.** To complement a verifiable data registry, we need a platform-neutral and interoperable peer-to-peer messaging protocol to communicate verifiable credentials and presentations among issuers, subjects, and verifiers. Hyperledger *Aries* [35] provides such a protocol, called *DIDComm*, to facilitate interaction between different DID subjects using DID public keys and services endpoints, with support for encryption. *Indy* and *Aries* together provides a complete set of tools to issue and revoke VCs, and to present and verify VPs. While *Indy* provides a decentralized registry to resolve DID documents and also schema and definitions for VCs and VPs, the actual exchange of credentials through the issuance of VCs, and the presentation of VPs is a peer to peer process where the issuer, holder, and presenter interact directly. The Aries *DIDComm* protocol also supports encryption which enables privacy between two communicating subjects. Looking ahead, these tools would be useful in developing the identity exchange protocols for permissioned

DLTs in Chapter 4.

2.4 Interoperability in Blockchains

Since its initial boost in popularity for the use in cryptocurrencies such as Bitcoin [2], blockchain as a technology has seen rapid growth. This fast paced development of newer alternatives to the Bitcoin platform in order to support advanced smart contracts (e.g. Ethereum [4]), faster transactions (e.g. Algorand [5]), and enterprise use cases (e.g. Hyperledger Fabric [9]) has led to the explosion of different types of DLTs [3]. This has resulted to the introduction of several independent blockchain networks in practice, often based on different DLT platforms, operating as isolated silos with no defined way to interoperate [15]. The term *interoperability* in the context of DLTs can refer to different properties which allow two separate blockchain networks, or one blockchain network and another traditional centralized system to faithfully exchange data. Interoperability often refers to the ability to counter fragmentation between different systems. The blockchain ecosystem as a whole is fragmented in different layers, and accordingly the fragmentation can be categorized into the following types [15]:

(i) Technical fragmentation. Technical fragmentation refers to the differences in the data communication protocols over the physical layer upto the transport layer.

(ii) Syntactic fragmentation. While technical fragmentation is at the lowest level, at a higher level syntactic fragmentation denotes the different formats and specifications according to which data is stored and communicated within the different DLT networks. This also includes the difference in representation of transactions, identity, membership, smart contracts, and other states of the DLT. For example, some DLT platforms might save data purely in the form of key value stores (e.g. Fabric), while others may save transactions as a DAG (directed acyclic graph) [80].

(iii) Semantic fragmentation. This refers to the trust model and consensus protocols according to which data is recorded on the shared ledger. Different DLTs often have different consensus protocols, as well as different membership criteria (such as permissioned and

permissionless) which govern how transactions are agreed upon and committed. The validity of any data originating from a source DLT network thus has to be verified by the destination network as per the consensus criteria of the source network for any faithful cross-network communication. Moreover, in order for the data to be accepted on the destination network, there must be consensus on the data itself, as well as consensus on the validity of the data as per the aforementioned criteria.

(iv) Fragmentation in governance. Governance in a DLT system is completely different from that in centralized systems due to the involvement of multiple stakeholders who do not necessarily trust each other. Different blockchain networks have different governance models and policies. As a result, the ways in which a blockchain network's structure can be changed (such as new members joining or old members leaving), or the properties such as transaction fees, etc., varies from DLT to DLT.

The purpose of any blockchain interoperability enabling mechanism is to bridge the above fragmentations, allowing different blockchain systems to communicate with each other for transfer of data and assets [18]. We note that technical and syntactic fragmentation are not new to the blockchain ecosystem, and they also existed in the traditional Web 2.0 landscape. Instead, the research towards blockchain interoperability is more focused towards bridging semantic fragmentation and the governance aspects [15]. Moreover, interoperability is required not only between DLT instances using different blockchain platforms, but also between different DLT instances using the same blockchain technology (say Hyperledger Fabric). The blockchain platforms in practice, such as Bitcoin, Ethereum, Fabric, or Burrow [32] are not designed to make their network instances interoperable with each other. As a result, different network instantiations using the same technology also suffer from the inability to communicate with each other that severely limits the practicality of blockchains in general [18, 81].

We categorize the existing literature on blockchain interoperability based on the type of the two collaborating blockchain systems - permissioned (private) or permissionless (public). The basis of consensus in private blockchains is byzantine fault tolerance [40, 58, 67], in contrast to the consensus protocols in public blockchains which has to deal with open membership and sybil attacks [5, 54]. Consequently, the mechanisms to prove and verify the authenticity of data originating from these two different families of DLTs are entirely

different [18]. In the following subsections we discuss the existing works on public-public, private-private, and public-private blockchain interoperability.

2.4.1 Public-public Blockchain Interoperability

Most public-public blockchain interoperability solutions are designed for cross-blockchain cryptocurrency transfer and exchange [20, 21, 82]. The primary problem addressed with respect to cryptocurrency exchange across two or more different parties within the same ledger or different ledgers is *guaranteeing atomicity*. Informally, a two party *atomic swap* protocol allows parties P_1 and P_2, holding assets $asset_1$ and $asset_2$ to exchange their assets, such that the exchange has either of the two outcomes: (i) P_1 holds $asset_2$ and P_2 holds $asset_1$ (ii) the exchange fails and P_1 holds $asset_1$ and P_2 holds $asset_2$. When the concerned assets are in two different ledgers then such atomic swaps are called *atomic cross-chain swaps* [19].

Hashed time-locked smart contract [19, 83] is one of the most prominent constructs for achieving atomic cross-chain swaps. A hashlock smart contract initialized using a hash h, from a cryptographic hash function $\mathcal{H}(.)$, ensures transfer of some asset from one party to the other upon receiving a secret s, such that $h = \mathcal{H}(s)$. Therefore, if two hashlocks are created with the same hash h, then revealing the secret s ensures unlocking both of them simultaneously, ensuring atomicity. Till the secret is not revealed, the asset is locked within the hashlock. To prevent assets from getting locked forever, a combination of timeouts with hashlocks resulted in hashed time-locked smart contracts. However, there are many corner cases where such mechanism can fail or give unfair advantage to certain parties. Herlihy et al. [19] is the first systematic analysis of the hashed time-locked smart contract based atomic cross-chain swaps. They modeled n-party cross-chain swap as a directed graph, and showed that such atomic cross-chain swaps are not possible if the graph is not strongly connected. Xu et al. [84] studied the success rate of transactions using hashed time-locked smart contracts, and showed that interestingly, the success rate of transactions depend a lot on the exchange rates off the cryptocurrencies.

Practical usability of any atomic cross-chain swap protocol depends on not only its security guarantees, but also performance, and other features such as the ability to compare

values of different assets for trading. Hashed time-locked contract based protocols are slow and require parties to wait for many minutes for settlements. To counter these disadvantages, Tesseract [21] aims to provide an ideal cryptocurrency exchange which is real-time, similar to a centralized exchange, as well as trustless like a decentralized exchange. Internally Tesseract uses a trusted execution environment [85], which is a hardware backed secure enclave protecting the integrity and confidentiality of software execution. A separate work XCLAIM [20] introduces the notion of cryptocurrency-backed assets, and uses it for cross-chain cryptocurrency exchange. The primary advantage of XCLAIM over hashed time-locked contracts is the requirement of significantly less transaction fees for exchanging assets, while not relying on any trusted party / trusted hardware.

The different cross-chain swap protocols discussed above are limited to their own different sets of blockchains, and applicable in different scenarios. For example, hashed time-locked smart contracts require the availability of an identical cryptographic hash function in the different blockchain smart contract systems. On the other hand, in case of cryptocurrency-backed assets or side-chains, everyone needs to trust a third ledger which is an unrealistic expectation. In a very recent work, Thyagarajan et al. [86] introduced a protocol for universal atomic swaps which is applicable for all permissionless blockchains, even those which do not support custom scripting as smart contracts. The centerpiece of their contribution is leveraging *verifiable timed signatures* [87] for designing a timelock without smart contracts. This protocol supports not only 2-party swaps, but also complex swaps extending to cyclic swaps.

Other notable public-public blockchain interoperability systems include payment channel networks [23, 88], sidechains [89], and anonymous multi-hop locks [82], *SmartSync* [22]. Besides cross-chain asset exchange, other interoperability protocols such as OmniLedger's Atomix [90] allow shards of the same blockchain to communicate through atomic transactions. Overall, there exists robust as well as provably secure protocols in the literature, for cross blockchain communication between two permissionless ledgers. Most of these techniques fundamentally rely on the open nature of permissionless ledgers which allow public verification of the validity of transactions. However, the different mechanisms need to be developed for private blockchains as discussed next.

2.4.2 Private-private Blockchain Interoperability

Compared to public-public blockchain interoperability protocols, private-private interoperability mechanisms are much less abundant [18]. The first network-neutral interoperability protocol for verifiable transfer of data and assets between two different permissioned networks was introduced by Abebe et al. [15]. From an infrastructure perspective they developed a *relay service* for connecting the networks, where the relay could be maintained by one or more participants of the networks. Here, the verifiability of cross network data transfer is assured through the use of proofs by attestations. The consensus view on some data is captured by attaching attestations (in the form of digital signatures) of the requisite number of participants of the originating network. The data is then bundled with the attestations and sent to the destination network, where these proofs are validated. Depending on the particular blockchain platform such as Fabric or Corda, the specific digital signature scheme for attestations may vary, however the overall protocol remains agnostic of the network implementations. The trust basis of this protocol lies with the identity of the permissioned network participants which is used to project the consensus view outside the network boundary.

Later *Hermes* [91] middleware introduced crash recovery capabilities making the gateways fault tolerant for robust blockchain interoperability. *Hermes* uses a variant of two-phase commit protocol (2PC) called ODAP-2PC. In [92], the authors proposed a formal model for state sharing problem across permissioned ledgers. Based on this model, they designed a protocol which leverages a public blockchain as a secure bulletin board for state commitments. Publishing views to an immutable public ledger prevents equivocation and allows the protocol to maintain correctness even in an adversarial model where all members of a committee can be malicious. Furthermore, intervals at which state commitments are published on the public ledger indicate a bound on the age of data exposed by the permissioned ledger.

Notably, in each of the existing private-private interoperability protocols, the trust basis is the identity of the permissioned network participants. The identity management protocol is however not deliberated on, and it is assumed that identity management is handled off-chain and coordinated in an ad-hoc fashion between the stakeholders. As a result, there is a

clear gap, and development of decentralized identity management protocols for cross-chain identity exchange is an essential requirement towards end-to-end private-private interoperability systems.

2.4.3 Public-private Blockchain Interoperability

Public-private blockchain interoperability involves transfer of data from a permissioned network to a permissionless network and vice versa. Depending on the type of the ledger from where the data originates, mechanisms to validate its authenticity differs. In [93], the authors devised a cryptographically secure protocol to enforce fair exchange of data between a private blockchain and a public blockchain. Concretely, the protocol allows a private blockchain to expose some data along with proofs validating its authenticity to an external party, only if the external party pays some monetary reward to a participant of the private blockchain. In this scheme, the permissioned blockchain presents a zero knowledge proof to the external party to guarantee fairness in the exchange of private data with the payment from the external party.

Departing from the usual chain relays that target PoW consensus based blockchains, Verilay [94] presents a verifiable blockchain relay for proof of stake DLTs, specifically the ones that are based on Practical Byzantine Fault Tolerance (PBFT) [58]. While Verilay is designed for interoperability across public proof of stake ledgers, it is capable of validating a ledger's state that uses 'opened up' PBFT consensus protocol [64], and hence can be tweaked to validate permissioned ledger states also.

Notably, in spite of some existing works towards public-private blockchain interoperability, no existing protocol provides concrete guarantees of safety and liveness of an interface between a permissioned ledger and a permissionless ledger. Informally, a public-private interoperability protocol must ensure that (i) the permissioned network participants agree on the data (or assets), as well as their order in which they arrive from outside the permissioned ledger boundary, and (ii) any data originating from a permissioned ledger must be verifiable by external entities with respect to the consensus state of the ledger. As a result, there exists a gap in terms of public-private blockchain interoperability that need to be addressed.

2.5 Cross-Blockchain Identity Management

Cross-blockchain identity management is an essential requirement especially for data sharing by private blockchains where the verification of proofs outside the boundary of a permissioned ledger requires participant identities such as public keys. In the permissionless domain there are a range of efforts attempting to address identity issues which include naming services such as Ethereum Name Service (ENS) [95] and Polkadot's naming system [96], as well as a number of solutions based on the Decentralized Identity Framework such as uPort [97], and Ontology [98]. However, these systems are either designed to simplify user experience in public networks, such as addressing an entity with a user-friendly name instead of an arbitrary byte string, or to provide a user-facing identity solution for creating and sharing credentials. These systems don't address the general problem of resolving and verifying identity issued by different ledgers for enabling cross-ledger communication.

Permissioned networks built on Corda [29] can transact states representing data and assets with each other via the *Corda Network* [99], a global publicly-available network that uses a common root of trust for identity. Though a consortium of nodes may optionally choose to deploy a segregated network with its own trust root for privacy and confidentiality, it will be unable to communicate directly with the rest of the global *Corda Network* unless it merges with it. The use of *Corda Network* however has a key limitation: it is restricted to the Corda protocol and doesn't allow integration with other DLT protocols like Fabric [9] and Besu [100].

Hyperledger Cactus [101] project for cross-chain interoperability leaves networks autonomous and in control of their interactions with other networks, but currently relies on manually sharing network identity information. Recently proposed decentralized identity management models such as Decentralized identifiers (DIDs) [25] and Verifiable Credentials (VCs) [1], which are *W3C Recommendations*, are being adopted for identifying entities and proving claims in decentralized settings. However, there is still a requirement of a decentralized identity management infrastructure, and protocols, capable of exchanging identities across different blockchain networks, especially permissioned networks. We work towards these goals in Chapter 4 of this thesis.

2.6 Trust Negotiation

Trust negotiation is the process of disclosing credentials and other information between two or more parties, such that sufficient trust can be established for them to transact in future. This notion of trust negotiation is not new, and the problem of trust negotiation have been faced by service providers in the Semantic Web [102]. The idea of privacy-preserving trust negotiation is to minimize the exposure of sensitive information while determining a common trust basis between the parties. In Web 2.0, trust basis is limited to a predefined set of certificate authorities (CAs) [30] which often come preinstalled with web browsers and operating systems. However, in a completely decentralized setting, in order to initiate a trusted interaction across two or more parties without relying on any canonical trusted set of CAs, negotiation of trust basis is essential. From a high level, privacy-preserving trust negotiation involves determining a common certifier that acts a trust anchor between two or more parties, without revealing any certifier that is not common between them. In Chapter 6, we define the most generalized form of symmetric trust negotiation as *Private Certifier Intersection*.

This problem is conceptually similar to Private set intersection (PSI) [103]. PSI has been extensively studied, with a wide range of solutions based on garbled circuits [104], homomorphic encryption [105], oblivious transfer [103], and other techniques [106–112]. However, there is no straightforward way of using PSI as a black-box to achieve privacy-preserving trust negotiation, particularly in the face of malicious adversarial corruptions. The key reason being the trust anchors in practice are often well known entities such as governments, large companies, etc. As a result, a malicious party can input a list of all plausible trust anchors (universal set) as input to the PSI in order to reveal the honest party's trust anchors. Intuitively, some form of validation of the inputs from the parties is required for privacy-preserving trust negotiation.

Private intersection of "certified sets", introduced in [113], is an augmentation of PSI with the additional requirement that the input claim-sets are certified by some certification authority (CA). However, this primitive has fundamentally different privacy goals as compared to privacy-preserving trust negotiation as it assumes that the information of the CAs is public and that the two parties agree apriori on which CAs they mutually trust. Conversely,

in the case of privacy preserving trust negotiation, the CAs (certifiers) are, in fact, the input to the protocol (and thus cannot be made public apriori) while the claims are public. We could also have a variant of this problem where the claims are additionally private.

Hidden-issuer anonymous credentials (HIAC), introduced very recently in [114], is a related cryptographic primitive that allows a credential holder to prove its claim(s) to a verifier without disclosing the identity of the credential issuer (i.e., the certifier). However, HIAC inherently requires the set of certifiers trusted by the verifier to be published as an "aggregator", thereby revealing the identity of each such certifier. Hence, while one could use for trust negotiation, such an adaptation would only achieve *one-sided privacy* since of the parties would have to make its list of certifiers publicly available.

Issuer-Hidden Attribute-Based Credential [115] is another related system in which a user can prove a credential issued to it without revealing which issuer among a set of issuers acceptable to the verifier issued that credential. Similar to HIAC, this system also provides one-sided privacy while revealing the certifier set of the verifier. Moreover, the concrete solution presented in [115] uses a trusted setup, which is costly in practice.

The "secret handshake" family of protocols [116, 117] enable (role-based) authenticated key exchange between parties without revealing any information beyond the common group memberships shared by the parties. These protocols, however, differ fundamentally from privacy-preserving trust negotiation in the sense that: (a) they do not capture the notion of validating certificates and claims, and (b) the process of issuing membership credentials is part of the protocol itself (in practice issuing credentials/certificates is a process independent of trust negotiation).

In Chapter 5, we present two variants of solution for trust negotiation across different permissioned blockchain networks. Extending in Chapter 6 we generalize the problem of symmetric trust negotiation as *Private Certifier Intersection*.

Chapter 3

Public-Private Blockchain Interoperability

Business-to-Business (B2B) and Business-to-Consumer (B2C) online marketplaces have gained much attention nowadays within various sectors, including e-commerce, ride-hailing, cloud service provisioning (e.g., cloud federations), supply-chain management, etc. However, there has been a continuing debate about the market-monopoly and unfairness they created in the digital economy [118, 119]. Such platforms typically work as the central agent or broker to interconnect various businesses and consumers. In such a firm-controlled marketplace, supporting trustworthiness and unbiased business transactions is always a concern. Blockchain is a natural extension to help trustworthy and bias-free business by allowing the stakeholders to interact over a decentralized marketplace. As a result, various recent works advocate for developing blockchain-based electronic marketplaces [120–125]. However, there is a fundamental limitation of the current blockchain technologies to support this, as discussed next.

An electronic marketplace is typically a multifaceted network with one or more closed business networks collaborating through B2B transactions and, finally, an open consumer network having B2C operations [17]. For example, in a typical supply chain, manufacturers, wholesalers, and retailers form different closed business networks, and finally, the end-customers create an open consumer network. Another example is cloud federation

platforms like OnApp [126], where small cloud service providers (CSPs) construct a closed consortium to provide cloud resources to customers. Depending on customer requests, the transactions flow from the open consumer network to various closed business networks, and the service is finally delivered back to the open consumer network.

Emerging blockchain networks such as *IBM Food Trust* [27], *TradeLens* [10], *Marcopolo* [28], etc., use private (permissioned) distributed ledger-based systems like *Hyperledger Fabric* [9] and *Corda* [29] to form closed consortiums of businesses. However, a key limitation of the existing private blockchain platforms is the restriction of their applicability within only closed consortiums where data and assets are not required to be communicated outside the network boundary. Thus, *Fabric*, *Corda*, or other existing private blockchains do not support any interface or protocols for interacting with the open network outside, which is crucial for building consortiums of service providers acting together to deliver services to the consumer network.

However, there are challenges in designing such interfacing.

- **First**, the businesses, as well as the consumers, can exhibit byzantine behavior in the absence of a firm-controlled marketplace. Therefore they can collude to deceive and take control over the consortium decisions.
- **Second**, the consumers' service requests need to be agreed upon by the businesses within the closed consortium along with their ordering, before they can be processed. Otherwise, any malicious business can take priority over a profitable service request, thus affecting the fairness of the system. Although private blockchain can ensure transaction execution order within the closed network, they do not support transactions from outside the closed network pertaining to Sybil attacks from the open network participants [54].
- **Third**, the service responses from the closed consortium also need to be transferred back to the consumer who requested the service. Such information must be verifiable by the consumers against the valid consensus at the business network. Further, the privacy of the information must be ensured.

Towards developing a decentralized collaborative architecture for service providing

consortiums, in this chapter we introduce ***CollabFed***, which addresses the above challenges by building a novel decentralized interface between the private blockchain networks and the open network of consumers. *CollabFed* ensures multi-party consensus validation and considers threats such as Sybil attacks and byzantine behaviors of the participants. The decentralized interface is engineered through a unique combination of the public blockchain and private blockchain networks by enabling interoperability between them to support trusted and secure data transfer in both the directions, that is (a) from the consumers to the businesses and (b) from the businesses to the consumers (**Contribution-1**). Our *Consensus on Consensus* mechanism handles the transfer of data from the open network into the private blockchain in a secured and verifiable manner (**Contribution-2**). We employ a novel mechanism based on *collective signing* (CoSi) technology [31] to generate verifiable results from the consortium, which is accessed securely by the consumers (**Contribution-3**). Moreover, *CollabFed* facilitates the collaboration among the participating businesses and enables fair scheduling of requests through a distributed consensus. Performance in terms of latency is of utmost importance here, so we analyze the effect of *order-execute* and *execute-order* transaction execution workflows on the performance of request scheduling.

Considering a use case of a decentralized brokerless cloud federation, we have done a proof-of-concept (PoC) implementation of *CollabFed* using *Ethereum* as the public blockchain platform and *Hyperledger Fabric*, and *Burrow* as the two different candidates for the private blockchain platform, and tested it with three emulated CSPs (**Contribution-4**). The experiments prove the viability of *CollabFed* as a platform for service provisioning consortiums, which supports interaction between a private blockchain network and the end-consumers. Evaluation of the performance shows acceptable overhead on the federation, and a Mininet-based emulation with 32 CSPs also validates its scalability over a large geo-distributed setup.

3.1 System Model and Design Challenges

We consider the interconnecting network between the consumers and the closed consortium to be partially synchronous where there is an upper bound Δ on the time of message delivery [127, 128]. If a message is not received within the time-bound Δ, then it is con-

sidered as a message fault. The intuition is that in a realistic communication, the messages must have arbitrary but bounded delay. This results in challenges such as unordered message delivery and message drops. Additionally, we consider different types of attacks that might affect the above operations, as follows.

3.1.1 Threat Model

A decentralized consortium is prone to the following types of attacks, which we take care of in the design of *CollabFed*.

Byzantine participants: We consider that at most $\frac{1}{3}$ of the participants, both for businesses and consumers, may exhibit byzantine behavior [40, 58, 62, 66]. A consumer can try to deceive the consortium by sending different requests to different businesses, while the businesses can collude themselves to alter the decision protocols' results to take control of the consortium.

Sybil attacks: BFT consensus protocols assume that each participant has only one distinct identity [58, 66, 128]. If somehow one participant can generate multiple identities, then using such redundancy, it can launch a "Sybil Attack" [54]. The consumers thus can launch a Sybil attack to the closed consortiums by using multiple identities.

Impersonation attacks: As a decentralized architecture, the consortium does not have a single spokesperson responsible for communicating with the open network consumers. Exploiting this, a malicious business from the closed consortium might try to deceive a consumer by posing as the consortium's spokesperson and providing false information.

Leakage of sensitive information: The business and the consumers communicate over an open, unsecured channel through message passing. Therefore, sensitive information like credentials, contact information, etc., might get leaked.

3.1.2 Design Philosophy and Challenges

CollabFed's primary objective is to develop a mechanism through which any closed consortium designed using a private blockchain platform can interface with open consumer networks. Considering the threat model as discussed above and the possibility of unordered message delivery along with message drops, in *CollabFed*, the following two guarantees need to be ensured at the consortium interface.

Definition 1. Consortium Interface Safety - The interface should ensure that all the correct consortium members agree on the same set of incoming consumer requests in same order.

Definition 2. Consortium Interface Liveness - The interface must ensure that all the correct consumer requests are eventually be processed and committed by the closed consortium.

Thus, a mechanism is needed such that the interface meets the safety and liveness guarantees, and the consortium members are in a consensus on each request. To achieve consensus over the ordering of consumer requests from the open network, we propose to use public blockchain platforms [4, 5, 50] for interfacing the closed consortium to the consumers of the open network. The consensus algorithms over a public blockchain setting are designed to be resistant to Sybil attacks. Therefore, using a public blockchain platform, the consumers' requests from an open network can be ordered. However, merely clubbing together any public and private blockchain is not enough to enable the targeted consortium interface; there are open challenges that need to be solved.

(i) **Passing consensus of one network to another:** The public and the private blockchain networks run their own consensus protocols independently. The interface should pass the consensus information from one network to another by ensuring (i) security, and (ii) accountability. The interface should guarantee that the consensus information of one network is verifiable at the other network.

(ii) **Transferring sensitive information from the closed network to the consumers:** Once a consumer request is scheduled and processed by the closed consortiums, the

associated service information such as access credentials, invoice, shipping information, etc., need to be passed to only the targeted consumer who has requested for the service. Therefore, merely putting the information to the public blockchain will not help, as anyone will access it. Protocols need to be designed to share such sensitive information with the targeted consumer only.

(iii) **Verifiability of the consortium decision:** The consortium's decision of scheduling, service provisioning, etc. comes through a consensus over the private network. However, once this information is forwarded to the public network, the consumers should be able to verify such decisions to avoid any byzantine behavior from the colluded consortium members.

3.2 Decentralized Consortium Interface

The functionality of *CollabFed Consortium Interface* is broadly two-fold: (a) transferring consumer requests from the open network to the closed consortium members (Figure 3.1), (b) transferring consortium responses to the open network consumers in a secure and verifiable way (Figure 3.2). The *Consortium Interface Safety* is achieved using two rounds of consensus over the consumer requests – (1) *regular consensus (mining) of the public blockchain*, and (2) A *Consensus on Consensus* mechanism. The details follow.

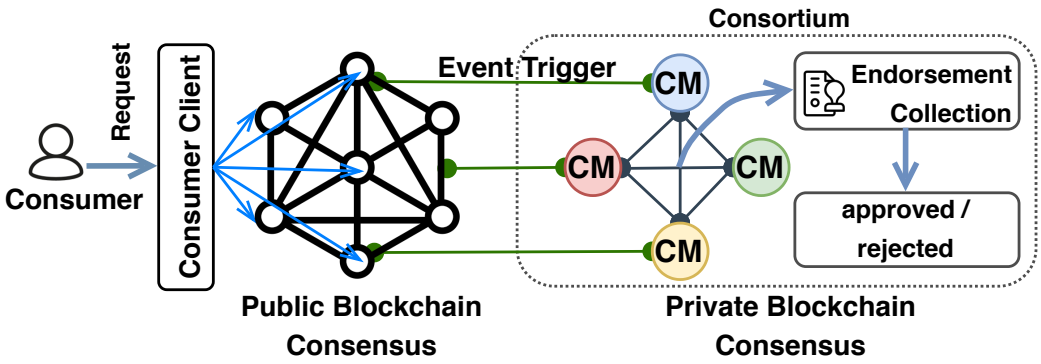


Figure 3.1 Transferring Consumer Requests from Public Blockchain to the Consortium Members (CMs)

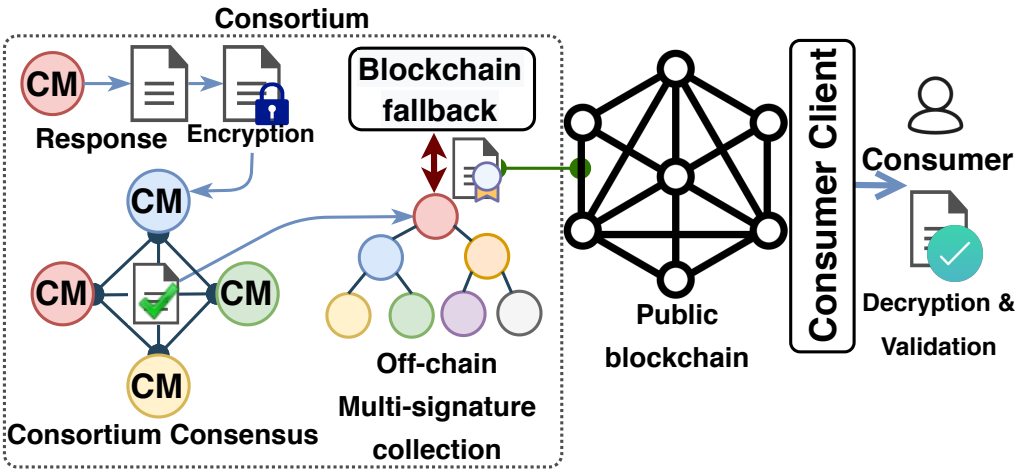


Figure 3.2 Secure and Verifiable Data Transfer from CMs to Consumers

3.2.1 Regular Consensus (Mining) over Public Blockchain

Before scheduling and processing any consumer request, the consortium members must reach a consensus on the same. Moreover, there needs to be a consensus on the order in which the requests are to be considered to ensure *Consortium Interface Safety*. This ensures that a malicious member of the consortium cannot collude the network by triggering the scheduling of an invalid consumer request or take priority on a specific consumer request. *CollabFed* uses public blockchain in conjunction with the private consortium to support this. However, for supporting interoperability between the two networks, the consensus has to be propagated between them.

To interact with the consortium, consumers send their requests through the public blockchain. These requests are formed as transactions to a smart contract - *User Request Contract*, deployed in the public blockchain. Just like a web interface of a central firm-controlled platform, this smart contract acts as the communicating point for the consumers to reach the consortium, albeit in a decentralized way. The "consumer request" transactions are then committed to a block in the ledger through the public blockchain platform's mining/consensus process. For example, *Ethereum* used a modification of one of the most popular consensus protocols: "proof of work" (PoW) [2], and recently it switched to an alternate consensus protocol - Proof of Stake [50] [129] on September 15, 2022. There are several other consensus protocols used in different blockchain systems such as Bitcoin-

NG [51], Byzcoin [52], Algorand [5] etc.

These consensus protocols have different safety and liveliness assumptions of their own; however, their common objective is to reach consensus on a block of transactions. Moreover, since these are permissionless blockchain protocols, they are designed to resist Sybil attacks.

Once a block is mined and committed in the public blockchain, this ensures that there is a consensus on the particular block and their order in which they are committed, since each block is linked to the previous one through its cryptographic hash. Moreover, the set of transactions in each block also has a fixed packing order for the smart contracts' deterministic serial execution. Despite these properties, public blockchain consensus itself is not enough to satisfy *Consortium Interface Safety*, and consortium members cannot simply pick user requests from the public blockchain and start processing them. The reasons are as follows. (1) Due to the partially synchronous network, some consortium members might not get the mined block in time and thus cannot participate in its scheduling. (2) Malicious consortium members may introduce and schedule invalid consumer requests that are not mined at all. (3) Public blockchain consensus protocol like PoW, often goes through temporary forks [130], resulting in conflicting consumer requests or conflicting ordering in different members. Thus, *CollabFed* has to carry out a second round of consensus, which we call *Consensus on Consensus*.

3.2.2 Consensus on Consensus

In [81], the authors have shown an interesting result that states that cross-chain communication is impossible without a trusted third party. To circumvent this impossibility result, *CollabFed* uses a novel idea where the private consortium members also participate in the public blockchain to represent themselves as their own trusted agent. Whenever a new block is committed in the public blockchain, the trusted agents corresponding to the private consortium members get an event-trigger, which in turn invokes a *Propagation Contract* in the private blockchain network. Before invoking the *Propagation Contract*, the transactions of the public blockchain can be verified individually by the consortium members by existing methods such as *Simplified Payment Verification (SPV)* as used in standard public

blockchain like Bitcoin [2].

The task of the *Propagation Contract* is to collect **verification endorsements** from consortium members for each consumer request. The *verification endorsements* are the digitally signed certificates from the consortium members, indicating that the corresponding members agree on the processing of a consumer request committed over the public blockchain. As per the standard BFT protocols [52, 58], a consumer request can be committed for scheduling in the private consortium if the majority ($\frac{2}{3}$rd) of the consortium members endorse the request transaction. The endorsement protocol used in the *Propagation Contract* is shown in Figure 3.3. The details follow.

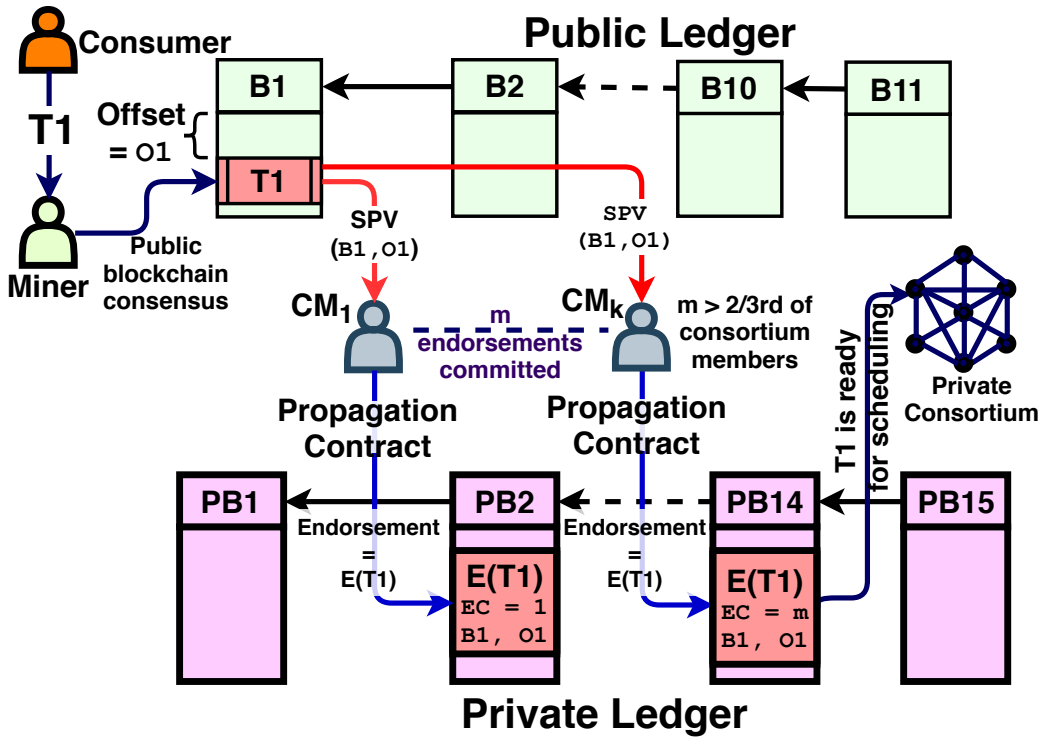


Figure 3.3 Propagation Contract: *Consensus on Consensus*

Endorsement Initialization: Whenever a consortium member receives a "consumer request" transaction through the event listener of the public blockchain, it checks whether there is already an endorsement available in the private ledger corresponds to that transaction. If no endorsement is available, it initiates the endorsement collection process for that particular request by initiating the `endorsement-count (EC)` variable set to 1, and committing the signed endorsement in the private ledger. The request is also accom-

panied by a sequence number for representing its order. This sequence number is formed as {`blocknumber, offset`}, indicating the block in which the request transaction is committed in the public blockchain, and its packing order inside the block.

Endorsement Propagation: As other consortium members also get the same consumer request and with the same {`blocknumber, offset`} through the event listener of the public blockchain, they also execute the *Propagation Contract* for it, which adds their signed endorsements while incrementing the `EC`. Each execution of the *Propagation Contract* is also a transaction. Therefore, each endorsement also goes through the consensus process of the private blockchain.

Commitment: Thus, the number of endorsements for a request goes up until it reaches greater than two-third of the number of consortium members ($\texttt{EC} > \frac{2}{3}|consortium|$). At this point, the majority of the consortium participants have consensus on the request through endorsements, and each such endorsement has a consensus of the network. Thus, the consumer request is marked as approved and ready to be scheduled.

Theorem 1. The *Consensus on Consensus* mechanism ensures consortium interface safety and consortium interface liveness.

Proof. Whenever a transaction is committed in a block in the public blockchain, it implies all its correct participants including consortium members agree on it, along with the (`blocknumber`, `offset`). The *Consensus on Consensus* mechanism endorses the transactions from the public blockchain and then commits the endorsements in the private blockchain. A transaction is scheduled only when more than $\frac{2}{3}$ of the consortium members endorse the transaction. Given that each endorsement transaction also undergoes consensus in the private blockchain, and the given verifiability property of the private ledger, a transaction from the public blockchain is executed only when the majority of the consortium members endorse it. Further, the transactions are executed in the order of (`blocknumber`, `offset`) parameters of the public blockchain ensuring agreement on the order. Thus the *Consensus on Consensus* mechanism ensures interface safety.

Consortium interface liveness depends on the liveness of the public blockchain. The event-listeners for correct consortium members eventually trigger the propagation contract

when a transaction is committed in the public ledger. Even if there is a temporary fork, the propagation contract is executed when the transaction is finally committed in the public ledger. □

The *Propagation Contract* triggers *Scheduling Contract* that schedules the requests based on a predefined business logic. After a request is scheduled and processed over the closed consortium, the service results have to be transferred back to the consumers. The details follow.

3.2.3 Secure and Verifiable Response Transfer

A consortium is operated collectively by its participant businesses. Hence, any data/information provided by it has to be the result of the collective consensus process. Thus, in the absence of a central coordinating platform, this consensus has to be collected and verified by the consumers, without depending on any trusted agent. There can be two variations of information originating from the consortium. (1) ***Consortium information*** such as information about the participating businesses, service catalogs, etc., and (2) ***Request responses*** that are the results of scheduling and processing consumer requests such as a digital document.

Both of these kinds of data are generated collectively by the consortium members through the private blockchain's consensus process. However, this consensus information has no manifestation outside this closed network. Thus, consumers being outside the consortium and not participating in the consensus protocol cannot verify the correctness of the data that is committed through transactions in the private blockchain. A separate protocol has to be designed through which information transfer from the consortium to the consumers can be validated outside the private network concerning the consensus of the participating businesses. Moreover, although *consortium information* can be considered publicly available, the *Request responses* to the consumers may contain sensitive information that should remain confidential while being transferred across the open network of the public blockchain.

In *CollabFed*, we use the concept of *Collective Signing* (CoSi) [31] where a set of consortium members collectively sign a valid information to make it verifiable. We utilize

Boneh-Lynn-Shacham (BLS) cryptosystem [131] for collecting and aggregating signatures from the individual participating businesses. Similar to Byzcoin [52], which uses CoSi to reach to a BFT consensus, a piece of information posted by the consortium through the public blockchain is considered to be valid, if and only if it has been signed by at least $\frac{2}{3}$rd of the consortium members. The details follow.

BLS Signatures

A BLS signature is computed as $\sigma_i(m) = H\left(m\right)^{x_{\mathsf{C}_i}}$, where m is the message that is to be signed, $H\left(.\right)$ is a cryptographic hash function, and x_{C_i} is the secret key of the consortium member C_i. The property that makes BLS signatures special is that they can readily be extended to multi-signatures. Therefore, for n members participating in the consortium, $\mathsf{C}_1, \mathsf{C}_2, \cdot, \mathsf{C}_n$, the aggregated multi-signature can be calculated as follows.

$$\begin{aligned} \sigma_{1..n}(m) = H\left(m\right)^{x_{\mathsf{C}_1}+x_{\mathsf{C}_2}+..+x_{\mathsf{C}_n}} = \prod_{i=1}^{n} H\left(m\right)^{x_{\mathsf{C}_i}} \\ = \sigma_1(m) \times \sigma_2(m) \times .. \times \sigma_n(m) = \prod_{i=1}^{n} \sigma_i(m) \end{aligned} \tag{3.1}$$

This aggregated multi-signature $\sigma_{1..n}(m)$ can be verified with the help of the public keys of the individual consortium members. This verification is done by comparing the pairing operation between the aggregated signatures and the aggregated public keys. The aggregated public key for n members is calculated as $\prod_{i=1}^{n} \mathcal{Y}_{\mathsf{C}_i}$, where $\mathcal{Y}_{\mathsf{C}_i}$ is the public key of C_i.

Posting information using BLS

Any information about the consortium is communicated to the consumers by posting the same in the public blockchain. Such information originates from the result of the *Collaboration Contract* in the private blockchain, which is responsible for reaching consensus on them. This resultant data like updated information or updated catalog, etc. must be collectively signed by at least $\frac{2}{3}$rd of the participating consortium members. This again has two different levels of security requirements for *Consortium information* and *Request responses*.

Posting Consortium Information to the Public Blockchain: Let $\mathcal{I}$ be a piece of public consortium information that is meant to be seen by all consumers. $\mathcal{I}$ is proposed by a consortium member in the private blockchain where consensus is reached over it. To post this information over the public blockchain, the consortium members over the closed network construct a *Signing-Request message* as $\texttt{sign}\{H(\mathcal{I}), \mathbb{B}, \sigma_{\mathbb{B}}(H(\mathcal{I}))\}$ and forward it to all other consortium members. Here $\mathbb{B}$ is a bitmap indicating which members have signed the message and $\sigma_{\mathbb{B}}(H(\mathcal{I}))$ is the aggregated collective signature on the hash of the message $\mathcal{I}$. Every consortium member, upon receiving this message, adds its own signature through multiplication, as shown in Eq. (3.1), updates $\mathbb{B}$ and sends back the response. Once signatures from majority of the members have been aggregated, the final response message $\{\mathcal{I}, H(\mathcal{I}), \mathbb{B}, \sigma_{\mathbb{B}}(H(\mathcal{I}))\}$ is posted in the public blockchain. The authenticity of this message can be easily verified using the public keys of the members who have signed the message, and the integrity can be checked by computing and comparing the hash of $\mathcal{I}$. This verification process is carried out by the *Consumer Client* and is transparent to all the consumers. The *Consumer Client* only accepts those messages which have the required number of signatures ($> \frac{2}{3}|consortium|$) along with the proper hash.

Posting Private Information for a Consumer: Posting private information to a consumer through the public blockchain requires some mechanism to preserve confidentiality. This is done by encrypting the message using the public key $\mathcal{U}$ of the consumer. The message is also similarly authenticated using the aggregated multi-signature of the consortium members. Thus the final message which is posted in the public blockchain is $\{< m >_{\mathcal{U}}, H(< m >_{\mathcal{U}}), \mathbb{B}, \sigma_{\mathbb{B}}(H(< m >_{\mathcal{U}}))\}$, where $< m >_{\mathcal{U}}$ denotes a message m encrypted using the key $\mathcal{U}$. Thus, only the consumer can decrypt the message using its secret key $x_{\mathcal{U}}$. The *Consumer Client* handles the decryption and verification of authenticity.

3.2.4 Optimizing the Latency for Signature Collection

Since the messages to be transferred from the consortium to the consumers already have to be committed in the private blockchain, the multi-signature collection process is decoupled and carried out off-chain to improve the latency. Thus the consortium members communicate through peer-to-peer messages to form the verifiable signed message. This

multi-signature mechanism's latency depends on the way the members forward the messages and collect back the signatures to generate the final payload by aggregating them. Thus a communication tree is formed along which the singing request and the signatures are exchanged. One extreme case of this is when one of the members acts as the leader, and the other members sign their messages and forward them back to it. The leader constructs the collective signature by including its own signature and validates other members' signatures against their public keys.

This strategy is likely to have low latency because of its star topology with a path length of at most one but will have high signature combination computation overhead for the leader. Another extreme is to consider a linear chain of consortium members through which the above round of messages propagate; this will have less computation overhead for each member but will have high network latency. *CollabFed* uses an M-ary tree structure to propagate multi-signature collection messages through which individual signatures are collected, and the multi-signature is constructed following Eq. (3.1). Interestingly, the latency for multi-signature generation changes with the value of M, which we analyze in Section 3.4.

Handling denial of service: Off-chain multi-signature collection improves the latency of the process. However, it introduces the risk of denial of service. Although the message to be signed is first committed in the private blockchain through the consensus process, some malicious consortium participants may try to halt the consortium through denial of service attack by not responding to signature collection requests. As a result, to prevent that and detect the faulty members to hold them responsible, *CollabFed* resorts to a blockchain contract-based signature collection after the off-chain protocol fails (possibly with a timeout). For a message, the *Signature Collection* contract is initialized in a similar way as *Propagation Contract*, and gathers BLS signatures of the members. Thus any non-cooperating member is detected through this transparent process, who can be held responsible.

3.3 Use Case Implementation: Cloud Federation

To evaluate the potential of *CollabFed*, we have implemented a use-case of cloud federations like *OnApp*, where a group of CSPs participate in a single marketplace to offer cloud infrastructure such as virtual machines (VMs) as a service (IaaS) to the consumers. Traditionally cloud brokers [132] or centralized marketplaces like *OnApp* coordinate all interactions between the CSPs and the consumers. To design a fully trustless decentralized architecture for cloud federations, we use *CollabFed* to implement a private network of CSPs and a public network of consumers, called *CollabCloud*. Apart from the basic functionalities of *CollabFed*, *CollabCloud* implements a Fair Scheduling Contract within the CSP consortium to schedule the VM requests among the participating CSPs while ensuring fairness in terms of profitability of the CSPs and quality of service (QoS) for the consumers.

The *Fair Request Scheduling Contract* takes into account the contribution of the individual CSPs in the federation and schedules consumer requests in proportion to it. We define the federation $\mathbb{F} = \{\mathsf{C}_1, \mathsf{C}_2, \ldots, \mathsf{C}_n\}$ as a collection of CSPs C_i. A CSP C_i can support certain VM configurations which are represented by $\mathbb{V}^{\mathsf{C}_i} = \{\mathsf{vm}_1, \mathsf{vm}_2, \ldots, \mathsf{vm}_m\}$. Thus the catalog of the federation is the union of all such VM configurations being offered by the individual CSPs, represented as $\mathbb{C} = \bigcup_{\mathsf{C}_i \in \mathbb{F}} \mathbb{V}^{\mathsf{C}_i}$. Similar to the catalog, the contribution of each CSP C_i is a set of *VM offerings*, denoted by $\mathsf{O}^{\mathsf{C}_i} = \{\mathsf{o}_1, \mathsf{o}_2, \ldots, \mathsf{o}_m\}$. A *VM offering* is defined as a three-tuple: $\mathsf{o} = \{\mathsf{vm}, k, c\}$, where vm denotes a VM configuration, k denotes the quantity of the VMs of the particular configuration the CSP can offer, and c denotes the expected pricing of that VM type. A consumer request for a VM is defined as a four-tuple: $\mathsf{CR} = \{\mathsf{CR}_{id}, \mathcal{U}, \mathsf{vm}_j, T\}$, where CR_{id} is the unique identifier of the consumer request, $\mathcal{U}$ is the public key of the consumer making the request, $\mathsf{vm}_j \in \mathbb{C}$ is the VM configuration selected from the catalog $\mathbb{C}$, and T is the duration for which the VM is requested.

The fair scheduling smart contract is shown in Algorithm 2. The input to the algorithm is a consumer request CR_i, the proportions of contribution of all CSPs in the federation $\mathbb{K} = \{\hat{\mathcal{K}_{\mathsf{C}_i}} |\ \mathsf{C}_i \in \mathbb{F}\}$, and an array $\mathbb{W}$ consisting of the results of this algorithm for last $|\mathbb{W}|$ scheduled requests. We define infrastructure contribution $\mathcal{K}_{\mathsf{C}_i}$ of each CSP C_i as $\mathcal{K}_{\mathsf{C}_i} = \sum_{\mathsf{o} \in \mathsf{O}^{\mathsf{C}_i}} \mathsf{o.vm}.\mathcal{CPU} \times \mathsf{o}.k$, that is the weighted sum of the quantities of its *VM offerings* indicating the amount of IaaS capacity (hardware resources) contributed. Thus,

Algorithm 2: Fair Request Scheduling Contract

Input: CR_i, $\mathbb{K}$, $\mathbb{W}$
Result: Scheduled CSP: C_s

for $\mathsf{C}_j \in \mathbb{F}$ **do**
  * Initialize current proportion of scheduled requests of C_j to 0 *\
  $\mathcal{J}_{\mathsf{C}_j} \leftarrow 0$
  for $l \leftarrow 1$ *to* $|\mathbb{W}|$ **do**
    if $\mathbb{W}[l] = \mathsf{C}_j$ **then**
      $\mathcal{J}_{\mathsf{C}_j} \leftarrow \mathcal{J}_{\mathsf{C}_j} + 1$
    end
  end
  $\mathcal{J}_{\mathsf{C}_j} \leftarrow \frac{\mathcal{J}_{\mathsf{C}_j}}{|\mathbb{W}|}$
  $\mathcal{D}_{\mathsf{C}_j} \leftarrow \mathcal{J}_{\mathsf{C}_j} - \hat{\mathcal{K}_{\mathsf{C}_i}}$
end
$\mathsf{C}_s \leftarrow \text{argmax}_{\mathsf{C}_i \in \mathbb{F}}(\mathcal{D}_{\mathsf{C}_j})$
enqueue($\mathbb{W}$, C_s)
if $|\mathbb{W}| > \mathfrak{K}$ **then**
  dequeue($\mathbb{W}$)
end
return C_s

each time the catalog is updated, the proportion of contributions are also changed. The contribution proportion is thus $\hat{\mathcal{K}_{\mathsf{C}_i}} = \frac{\mathcal{K}_{\mathsf{C}_i}}{\sum_{\mathsf{C}_j \in \mathbb{F}} \mathcal{K}_{\mathsf{C}_j}}$, for each CSP C_i.

In essence, the scheduler works similarly to a *weighted fair queue* [133] which ensures that the rate of consumer requests received by each CSP C_i is proportional to $\hat{\mathcal{K}_{\mathsf{C}_i}}$. For this purpose, the scheduling contract keeps track of a window ($\mathbb{W}$) of the past scheduled results. We implement $\mathbb{W}$ as a queue containing results for past requests that is $\mathsf{CR}_{i-1}, \mathsf{CR}_{i-2}, \ldots, \mathsf{CR}_{i-|\mathbb{W}|}$. Here each result corresponds to some CSP to which the past request was scheduled. The algorithm first computes the proportion of requests scheduled to a particular CSP as $\mathcal{J}_{\mathsf{C}_j}$ and then computes the proportion deficit as $\mathcal{D}_{\mathsf{C}_j}$. The request CR_i is then scheduled to the CSP, having the maximum deficit in its share of past scheduled requests. Then the window $\mathbb{W}$ is updated by inserting the new result, and also removing the oldest result if $|\mathbb{W}| > \textit{some threshold}\ \mathfrak{K}$.

Verifiability of the Scheduling Algorithm: CR_i is obtained from the public blockchain, and $\mathbb{K}$ is available in the private ledger. Finally, the past scheduled requests are obtained from the previous results of the *Fair Resource Scheduling* contract in the private blockchain. Thus, each CSP has access to all the information from the two blockchains. For verifia-

bility, it must be ensured that all the CSPs act on the same version of information. With each execution of *Fair Resource Scheduling* contract, the value of $\mathbb{W}$ is altered. Therefore, the CSPs must know which version of $\mathbb{W}$ is applicable for which transaction. This is ensured in two different ways – **order-execute** and **execute-order** based executions of the contracts [9].

Case (i): order-execute – The transactions are ordered first, and the consensus is achieved on this ordering. The transactions are then executed sequentially based on the agreed order, and $\mathbb{W}$ is updated. Thus every CSP applies the transactions in the same sequence on $\mathbb{W}$, starting from the initial version.

Case (ii): execute-order – Each transaction is first simulated on a particular version of $\mathbb{W}$, and this version number is also included in the transaction. Then the simulation result (an updated version of $\mathbb{W}$) is sent for consensus. In the case of multiple such parallel transactions acting on the same version of $\mathbb{W}$, only one transaction is agreed upon during the consensus and accepted. The rest of them are rejected.

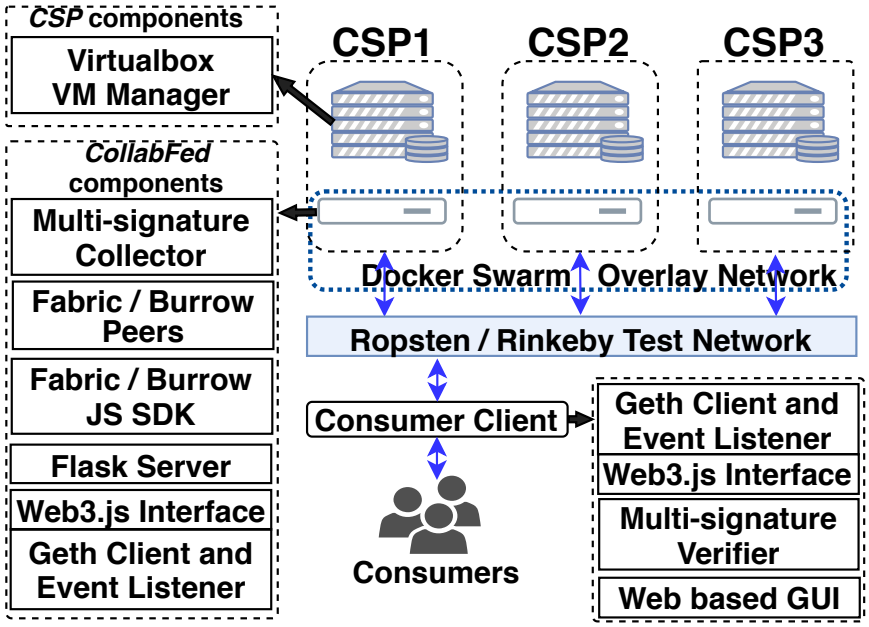


Figure 3.4 *CollabCloud* modules and Testbed setup

3.4 Evaluation

In order to test the feasibility and practicality of *CollabFed*, we have implemented a PoC of *CollabCloud* decentralized cloud federation. Each component of *CollabFed* along with the additional cloud federation specific functionalities are developed, and the end-to-end system is deployed in a testbed (Figure 3.4). Since *CollabFed* needs one public blockchain platform for providing the *Consortium Interface*, we have chosen *Ethereum* [4]. For the private blockchain, we have tested with *Hyperledger Fabric* and *Burrow* platforms. The public blockchain smart contracts are implemented using `Solidity (v0.5.0)` [1] language, and they are executed on the Ethereum Virtual Machine (EVM). We have used `Truffle` (`https://www.trufflesuite.com/`) for the development and testing of the Ethereum contracts. Two test networks [2], `Ropsten` and `Rinkeby` are used to run the consortium interface. `Ropsten` uses Proof of Work (PoW) whereas `Rinkeby` uses Proof of Authority (PoA) [134] for consensus. We evaluate *CollabFed*, as well as the cloud-federation functionalities from two different setups. First, we develop an in-house testbed with three emulated CSPs over six cloud servers (each CSP having two servers). Next, to analyze the scalability of different components of *CollabFed*, we perform an emulation-based evaluation over the Mininet virtual emulation network [33].

3.4.1 Platform Setup

To test the end-to-end functionality and performance of *CollabFed* along with its various components, we set up a PoC testbed of cloud federation emulating 3 CSPs participating in the federation. Figure 3.4 shows the setup where each CSP has two cloud servers – one 4-core Intel Core i5-4590@3.30GHz server with 8GB memory (Ubuntu 18.04, Linux Kernel 4.15) for running *CollabFed* services, and another 88-core Intel Xeon Gold 6152@2.10GHz server with 256GB memory (CentOS 7.7, Linux Kernel 3.10) for running the CSP's usual services including VM placement and hosting the VMs. All the services are run in `Docker` (`https://www.docker.com/`) containers, and the networking is established through a Docker swarm overlay network.

[1] `https://solidity.readthedocs.io/en/v0.5.0/`

[2] `https://docs.ethhub.io/using-ethereum/test-networks/`

For implementing the CPS functionalities, we have used `VirtualBox` (`https://www.virtualbox.org/`) for creating VMs, and a `Flask` (`https://flask.palletsprojects.com/en/1.1.x/`) server for accepting VM placement requests and interfacing with `VirtualBox`. Since each CSP has only one emulated data center, which is the host server itself, the placement algorithm does not affect our system's evaluation. However, each CSP has its own set of supported VM specifications that it offers, resulting in different catalogs. We use the *Fair Request Scheduling* contract that allocates requests based on the proportionality of the virtual CPU (vCPU) contribution in the federation by each CSP.

Apart from the testbed, to evaluate the scalability of different components of *CollabFed*, we also created a Mininet-based network topology for emulation. We created test scenarios with several *CollabCloud* CSP nodes ranging from 2 to 32 and the latency between them ranging from 50ms to 400ms to capture their performance in real-world deployments.

3.4.2 End-to-end Testbed experiments

In these experiments, we used the PoC testbed to evaluate each component's performance while doing end-to-end consumer request processing for VM provisioning. We used emulated consumers with numbers ranging from 4 to 64 and programmed them to send parallel requests at the same instance of time. We have evaluated the latency and overheads of processing these requests in each *CollabFed* module.

Consortium Interface: Each consumer request encounters the public blockchain twice, first when it propagates from the public blockchain to the consortium, and then in the *Resource Response Contract*, when the processed result is transferred back from the private blockchain to the public one. Figure 3.5 shows the distribution of latency for processing the consumer requests over the public Ethereum blockchain. The processing latency over Ethereum test networks varies widely at different times depending upon the usage by other Ethereum users across the globe. We have collected the data for two weeks at different times of the day, and the same has been plotted in Figure 3.5. We observe that the PoW-based consensus process of Ropsten test network has a higher transaction processing time compared to the PoA-based Rinkeby network. From Figure 3.5 we can observe that the

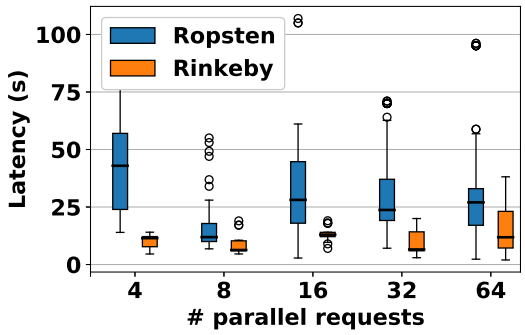


Figure 3.5 Transaction commitment latency in public blockchain indicating PoW-based Ropsten test network has a higher transaction processing time compared to the PoA-based Rinkeby network.

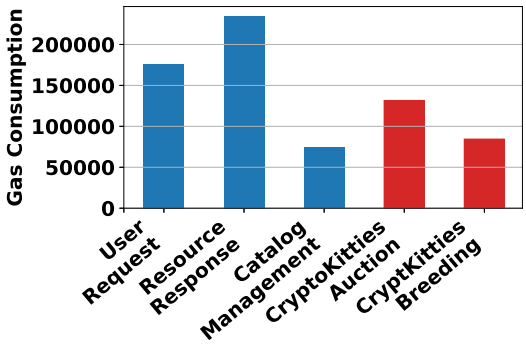


Figure 3.6 Gas consumption of the smart contracts of *CollabFed*, compared to other popular smart contracts. *CollabFed* shows acceptable gas consumption.

median time taken to commit a transaction is between 10 seconds and 40 seconds in case of Ropsten. However, notably there are some outliers where this latency is much higher, sometimes around 2 minutes. The reason for such variation of transaction processing latency are as follows. Firstly, in spite of a steady average block time of around 13 seconds (taken over a day), the time taken to mine a block varies from as low as 5 seconds to more than a minute. As a result, some transactions are mined quickly, while others take a longer time. Secondly, after submitting a transaction, it is not guaranteed to be included in the next block that is going to be mined. Instead, often it is mined in the second block or third block after the time of its submission, thus taking 2x or 3x the block mining time. Finally, because of these experiments being conducted in a test network and not the main network, there the block time was less stable, contributing to the difference in the median transaction latency.

Each contract in the public blockchain requires some transaction fees proportional to its computational complexity or storage requirements. In Ethereum, this is measured as "Gas". Figure 3.6 shows the gas consumption of the smart contracts of *CollabFed*, along with the cloud federation specific contracts. We observe that the *Resource Provisioning* contract is of the highest complexity since it has to store the multi-signatures for each transaction and the encrypted resource access information. To understand whether this Gas requirement is too high or too low, we benchmark these values concerning the Gas consumption by `CryptoKitties` (https://www.cryptokitties.co/) which is a common Ethereum applica-

tion, and found that they are comparable.

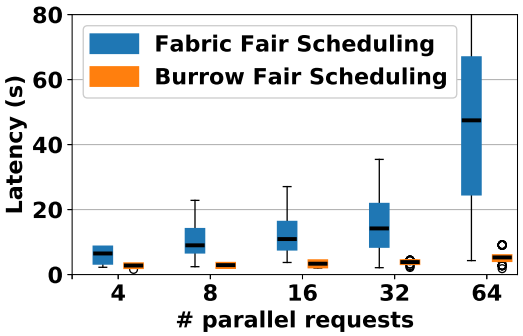


Figure 3.7 *Fair Scheduling* Latency: Fabric vs Burrow

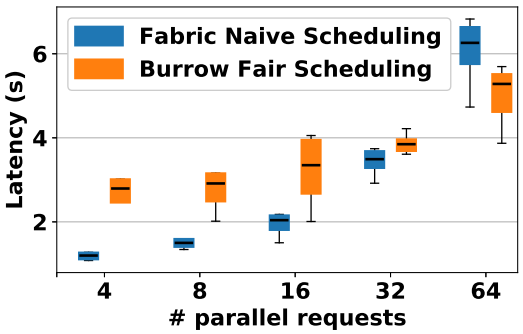


Figure 3.8 Fabric Static Scheduling vs Burrow Fair Scheduling

Request Scheduling over Private Ledgers: Figure 3.7 shows the time required for executing fair scheduling contracts in the private blockchain. We observe that the transaction processing time for Fabric is much higher than that of Burrow. The reason for this result is specific to the type of processing required by the *Fair Request Scheduling*. The key difference between Burrow and Fabric is the transaction execution workflow followed by them. Fabric follows *execute-order* flow, while Burrow follows *order-execute*. Executing first and then committing the results introduces a new problem for the type of contracts that read and change the system's common state, just like the *Fair Request Scheduling* uses a history of the already scheduled requests at different CSPs. The reason is as follows. While executing multiple transactions in parallel, let's assume that they get executed on the same current state S_c, and thus the output is based on S_c. After that, once any one of the transactions is committed, the current state is changed to S'_c. This state change also might change the output of other transactions that would be executed after it. As a result, when the other parallelly executed transactions are processed for committing, they fail in the ordering and validation phase since their execution results do not match with the execution result on $\mathcal{S}'_c$. Fabric does not retry to execute the failed transactions by itself, so *CollabFed* over Fabric reschedules the failed transactions, thus increasing the latency. We refer the reader to Chapter 2.2 for an in-depth description of different flows of transaction execution.

To validate our hypothesis regarding the source of higher overhead caused by Fabric, we also tested with a naive scheduling contract that schedules the requests based on a static rule depending on its ID. This contract does not depend on the current state of the blockchain.

In Figure 3.8, we can see that the scheduling latency of Fabric has dramatically improved. We also noticed that there are no transaction failures due to inconsistent execution results. Moreover, we saw no such latency improvements for Burrow with such a naive scheduling contract. It may be noted that for more parallel requests, Burrow still performs marginally better than Fabric. Consequently, we can conclude that the choice of private blockchain technology depends heavily on the fair scheduling contract's business logic.

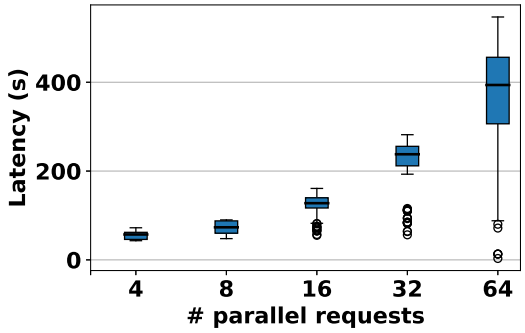


Figure 3.9 VM Provisioning Latency

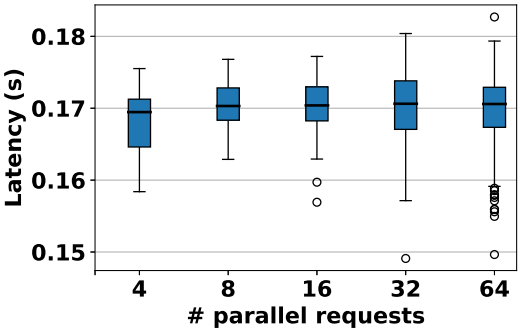


Figure 3.10 Sign. Collection Latency

After a request is scheduled, a VM is provisioned accordingly, and the access information is signed through collections of BLS multi-signatures. Figure 3.9 shows the distribution of the time taken for VM Provisioning. This increases with the increase in number of parallel requests, mainly due to the limited processing capability of the hardware of our setup. This latency is specific to the cloud federation application of *CollabFed*, and thus does not count towards its performance. Figure 3.10 shows the distribution of latency for multi-signature collection. We see that the multi-signature collection latency remains fairly consistent.

Resource Consumption: *CollabFed* consumes CPU, memory, and network bandwidth, which are an additional overhead to normal operations of a consortium. Figure 3.11 shows the box-plot distribution of CPU usage by *CollabFed* server for executing the private blockchain transactions in Fabric and Burrow, and for multi-signature collection. We observe that the CPU consumption is reasonably low, below 10% in most cases for all the services. Similarly, Figure 3.12 depicts the distribution for memory requirements which stays below 200MB.

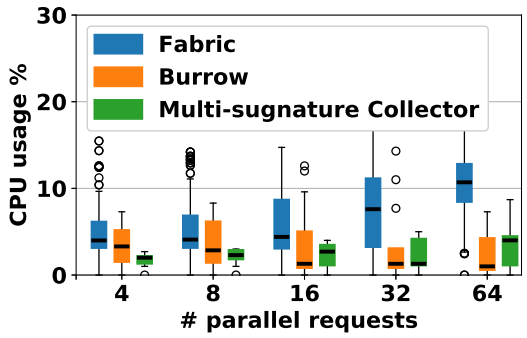


Figure 3.11 CPU Usage

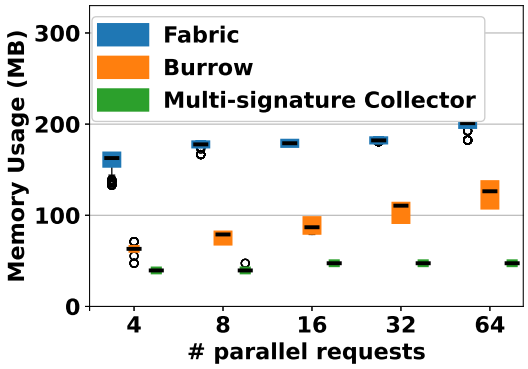


Figure 3.12 Memory Usage

3.4.3 Mininet scalability experiments

The public blockchain platforms being open networks have been designed to be scalable, and extensive research has been done to study their performance [5, 52]. We focus on the scalability of the private network and the multi-signature collection. For this, we set up an experiment with 32 emulated CSPs over a Mininet [33] topology, which forms a *CollabFed* consortium. We also changed the inter-CSP network latency to emulate the CSPs' spread across different geographic regions.

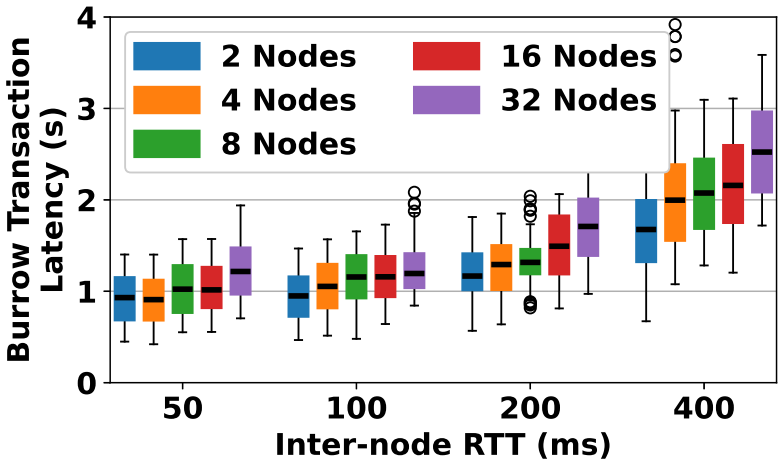


Figure 3.13 Burrow scalability

Figure 3.13 shows the distribution of Burrow propagation contract execution and commitment latency. The experiment has been done with inter-CSP latency varying in each case, from 50ms to 400ms. We observe that the median transaction latency lies around 2.5

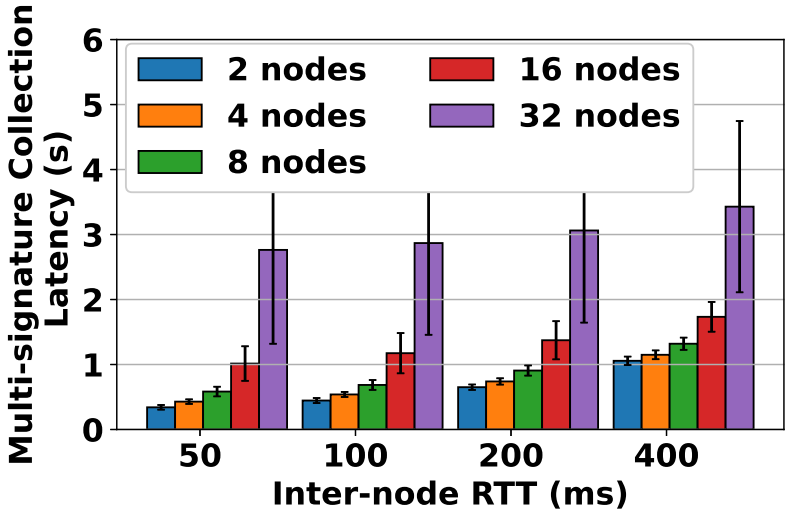


Figure 3.14 BLS scalability

seconds with 32 nodes and 400ms inter-CSP latency. Further, the increment in the transaction latency due to an increase in the number of CSPs or inter-CSP network latency is not very high, which indicates the scalability of the proposed approach.

Figure 3.14 presents the mean and the standard deviation of multi-signature aggregation latency with a varying number of nodes and inter-CSP latency values. We observe that the mean latency is below 2 seconds for 16 SPs and about 3.5 seconds for 32 SPs. This also indicates the scalability of the signature aggregation scheme. However, the multi-signature collection latency can have a big impact due to the collection tree structure. To study it, we constructed a complete M-ary communication tree with 32 SPs. Table 3.1 shows the multi-signature collection latency for different values of M. The inter-CSP latency for this test is kept fixed at 400ms. We can observe a sharp improvement in the latency from linear (M=1) to binary tree (M=2) structure. The latency is more or less stable from $M = 4$. However, the multi-signature combination complexity for individual CSPs increase with the value of M. Therefore, the value of M in a real deployment can be chosen based on this trade-off.

Table 3.1 Effect of communication tree on multisig collection latency

M	1	2	4	6	8	16	31
Mean Latency (s)	29.9	5.0	3.2	2.3	2.4	3.0	2.9
Standard Deviation	1.8	0.3	0.2	0.1	0.2	0.6	1.3

3.5 Summary

Towards a fully trustless decentralized architecture for an electronic business consortium providing services to consumers, in this chapter we introduce a public-private hybrid blockchain architecture with a unified interface between the consortiums and the open network. To the best of our knowledge, this is the first attempt to fill a critical gap in the application of blockchain in the enterprise and business use cases. *CollabFed* is flexible in terms of the choice of public and private blockchain networks; however, the performance and security guarantees depend on the assumptions of those underlying blockchain technologies and consensus protocols. The PoC implementation of *CollabFed* indicates that the system is scalable and performant with Hyperledger and Ethereum – one of the most popular private and public blockchain platforms, respectively.

The permissioned ledger based consortium of *CollabFed* provides a decentralized platform for B2B interactions between the organizations. But the B2B operations might not be limited to a single blockchain network [15]. There is a trend in the industry to create blockchain consortiums as *minimum viable ecosystems*, often comprised of closely related businesses such as trade finance [28, 135], logistics [10], etc. Although the consortium enables B2B interaction within the blockchain networks, these consortiums have compelling reasons to interoperate with other consortiums as well for achieving different business goals. As a result, inter consortium data transfer and other forms of cross-chain interactions are required. Although an existing work [15] has tried to introduce cross-network communication for permissioned ledgers, it leaves the problem of identity exchange unaddressed which is the centerpiece of any verifiable data transfer protocol between permissioned networks. We address this issue of cross network identity management in the next chapter.

Chapter 4

Decentralized Cross-Network Identity Interoperation

The current trend in enterprise blockchain industry is to create *minimum viable ecosystems* for business consortiums, i.e., limited business processes managed by permissioned blockchain networks that are built on diverse distributed ledger technologies (DLTs) [9,29]. This has fragmented the wider blockchain ecosystem, producing isolated networks with data and assets in silos [15]. But these networks have compelling reasons to interoperate while remaining operationally independent for privacy and logistical reasons. For example, an exported good that is financed on a trade finance network [28, 135] may be tracked on a separate provenance network [27] or a trade logistics network [10], while the traders' identities are attested on a shared KYC (Know Your Customer) network [136]. Such cross-network operations should be as trustworthy as transactions within those networks. The problem of transferring or exchanging assets across networks is well-studied, especially in the context of permissionless networks [19–21, 83]. We focus instead on the data sharing problem, that is, the transfer of data recorded in the ledger of one permissioned network to the ledger of another. To prevent fraud, the information must be accompanied by proof of veracity and provenance, as there is no guarantee that parties interested in it will be privy to both the ledgers.

Though several schemes exist for data sharing with proof, they are typically rele-

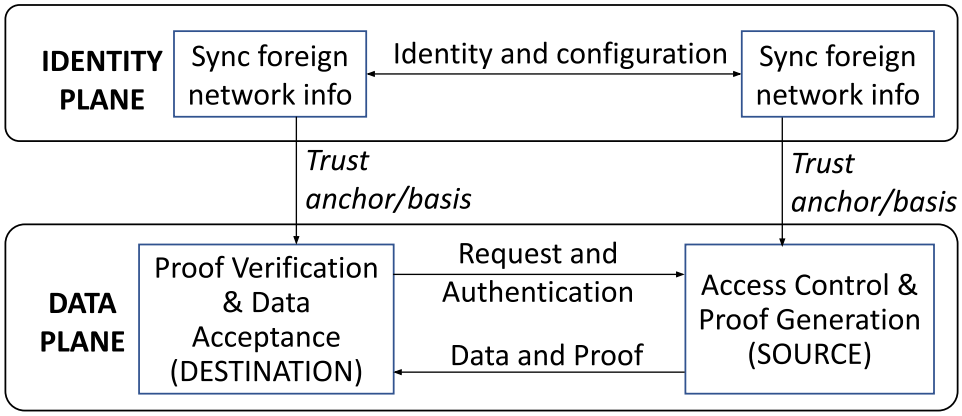


Figure 4.1 Generalized Cross-Network Data Transfer Protocol

vant for permissionless networks [137, 138], rely on intermediary infrastructure (relay chains) [139, 140], or use API-level integration between trusted end-points. Because we want networks to remain autonomous and interoperate directly as per need, we base our work on the foundation laid by Abebe *et al* [15]. According to their model, as illustrated in the lower half (or *data plane*) of Figure 4.1, a source network receiving an information request applies an access control policy through consensus before generating data and proof, and a destination network accepts the information after validating the proof against a verification policy, again through consensus. This proof must reflect the *consensus view* of the source network's peers. Though the protocol is agnostic of the nature of proof and policy, the opacity of a permissioned network necessitates some form of *proof-by-attestation*; i.e., the proof must consist of a quorum of network participants' signatures attesting to the veracity of shared data. But proofs-by-attestation rely on a network's ability to gain some visibility into its counterparty network, to enable its participants to know the identities and certificate chains of the latter's participants so signatures in proofs can be validated. Gaining such visibility is therefore crucial to establish a trust basis for cross-network data sharing. In this work, we separate the data requests and proof validation concerns from identity concerns by placing them in separate planes of activity (see Figure 4.1). Further, we replace the assumption made by Abebe *et al* that the networks' participants have a priori knowledge of each others' identities and certificates, which is impractical to guarantee, with a generic and pluggable *identity plane* protocol that provides a trust basis for data plane interoperation. In a naive implementation, credentials could be directly exchanged via network proxies, but this is both insecure (reliance on intermediaries) and unsustain-

able (discovering and exchanging credentials on the fly). Our primary contribution in this chapter is the design of a secure distributed identity management infrastructure and set of protocols linking permissioned networks and laying the basis for blockchain interoperation.

To build such infrastructure, we rely on the fact that enterprise blockchain networks are created by mutual agreement among existing real-world organizations possessing digital identities, though their network affiliations are not well-attested outside the network consortiums. We handle heterogeneity challenges in sharing and proving identity (credential formats, digital signature algorithms, sharing policies, identity providers) as well as privacy constraints (an organization may wish to reveal its network affiliation for cross-network data sharing while keeping its other attributes and affiliations secret). Our infrastructure enables networks to dynamically discover and sync each others' membership lists and verify the pre-existing identities of participant organizations without mandating a single centralized identity and validation registry (which would undermine the principle of decentralization and constrain the autonomy of interoperating networks). Dynamic changes in network memberships are handled, avoiding situations where non-participants or ex-participants can claim to be present participants of a network (which would result in invalid proofs of data). Finally, our system is agnostic of the DLT on which a network is built (structure, consensus protocol, identity issuance, cryptographic mechanisms).

In Section 4.1, we elaborate on the challenges and requirements while making the case for building our solution on the open frameworks like decentralized identifiers, verifiable credentials, and DLT-based identity registries. Our solution, built on Hyperledger Indy [34] and Aries [35], with its building blocks, architecture, and protocols, are described in Section 5.2, and a proof-of-concept implementation for Hyperledger Fabric [9] networks is presented in Section 5.4. We analyze our system in Section 4.4 before summarizing the findings in Section 6.7.

4.1 Decentralized Group Identity Management

An identity plane protocol involves distributed management of group identities. This is because a permissioned blockchain network is a collective rather than a unitary entity, de-

riving its identity from its participants (typically organizations), who may join or leave the network at any time. For privacy and security, such networks are open to outsiders only through invitation, and they manage identities internally in diverse ways, making cross-network identity management a challenge. For example, in a Hyperledger Fabric network, each participating organization runs one or more *Membership Service Providers (MSPs)* to issue identities and certificates to its peers and clients [9, 141]. Whereas in a Corda [29] network, identity is managed through a hierarchy of certificate authorities (CAs), from a single root CA to one or more doormen CAs down to individual node CAs.

Bridging the identity gap between networks so that they can share data and validate proofs, therefore requires cross-network identity management. This can be done in different ways, but if we wish to let the networks remain autonomous, avoid dependence on centralized identity providers, and preserve blockchain tenets of consensus and distributed trust, we must overcome several technical challenges:

- **Platform heterogeneity:** networks built on different DLTs have different structures and different procedures for transaction commitment and consensus.
- **Identity management heterogeneity:** DLTs have diverse mechanisms for identity management and use diverse cryptographic algorithms.
- **Lack of common identity infrastructure:** external identity providers for network's participants (members), who may serve as a common root of trust, are themselves diverse (i.e, non-standardized), and there is also no guarantee that two networks will have a common set of identity providers to vouch for the members of both.
- **Privacy:** participants must be required to share only the bare minimum information necessary for interoperation.
- **Security:** outsiders and ex-members should not be able to claim that they belong to a given network; i.e., the integrity of network membership should be guaranteed.
- **Consensus on identity:** registering the identity of a foreign network's member organization in one's ledger creates a shared truth for a network and therefore must be governed through its consensus mechanism.

To address these challenges, and knowing both the nature of permissioned networks and what technologies exist for distributed identity management, we can derive certain design requirements for our solution:

Decoupling Participants' Identities from Network: Identities created within permissioned blockchain networks, e.g. Fabric root certificates, are unknown outside network boundaries. Since interoperation requires identities to be shared and validated externally, and to avoid creating a central authority or spokesperson, identities ought to be independently verifiable by external entities while remaining under the participants' control. This necessitates the *decoupling* of participants' identities from the networks they belong to. Fortunately, we can rely on the *self-sovereign identity (SSI)* concept [25, 79], which allows an entity to control/manage its identity and prove properties about itself while retaining independence from any centralized registry/provider.

Decentralized Identity Registries: To complement SSI (as a means of decoupling), there must exist external registries to facilitate the resolution of network participants' decentralized identities [25] and the issuance, validation, update, and deactivation of credentials. To aid in mapping SSI to network-specific identity, such registries must maintain publicly accessible credential structures and validation information on behalf of one or more trusted identity-providing or credentialing authorities. Several *Decentralized Identifier* (DID) registries exist [142], but our use cases ideally require those that are not governed by centralized authorities.

Maintaining Network Membership Integrity: At any given instant, a network's membership list must be unambiguously available to a counterparty network in an interoperation session, and any changes to membership (joins and leaves) must be communicated using a trustworthy process. Incorrect membership information, or worse, malicious outsiders and ex-members pretending to belong to a network, will corrupt a data-sharing procedure that relies on proof-by-attestation, because a proof consumer may mistakenly accept an out-of-date (or fake) set of identities and signatures. Our solution must ensure that the membership information can be validated though consensus in the receiving network at any given instant.

Trust Anchors to Discover and Certify Network Consortiums: Though a permissioned network is a voluntary consortium of independent members, once created, it acquires a life of its own and remains wedded to its identity even if most of the original participants leave. Hence, for a newly-formed network, it is more straightforward to discover its identity and

subsequently discover its membership rather than deal with the chicken-and-egg problem of extrapolating a network identity from a group of participants' identities. For discovery, we must rely on reputed *trust anchors* that may either directly represent the consortium or serve as an authoritative reference for it. Examples: Being well-known stakeholders, IBM or Walmart could represent (anchor) the IBM Food Trust Network [27], whereas Maersk could represent TradeLens [10]. Such anchors can be implemented with different levels of decentralization; they could be single entities or sets of corroborating entities representing different network participants; they could run their own organizations or belong to shared identity registries, maintaining credentials in shared ledgers. Ultimately, the trust anchor(s) should represent the collective rather than a single network participant to ensure decentralization.
Compatibility with Networks' Identity Management Systems: No identity plane protocol we develop ought to require any internal modifications to underlying blockchain platforms. Though these networks may have to implement additional adapters to manage heterogeneity for interoperation purposes, the internal transaction commitment and consensus processes, which rely on existing certification and cryptographic mechanisms, must be allowed to proceed unchanged. Therefore, our solution must be simultaneously compatible with existing permissioned DLT identity management systems and agnostic of their specific implementations.

4.2 Solution

We now present a decentralized identity management solution to fulfill the requirements stated in Section 4.1, and whose development was guided by the following design principles:

- The solution should not be tied to, or only applicable for, a particular DLT.
- Networks and their participants should be free to choose identity registries and providers (or use their existing ones).
- Networks must retain their autonomy while gaining the ability to interoperate universally.

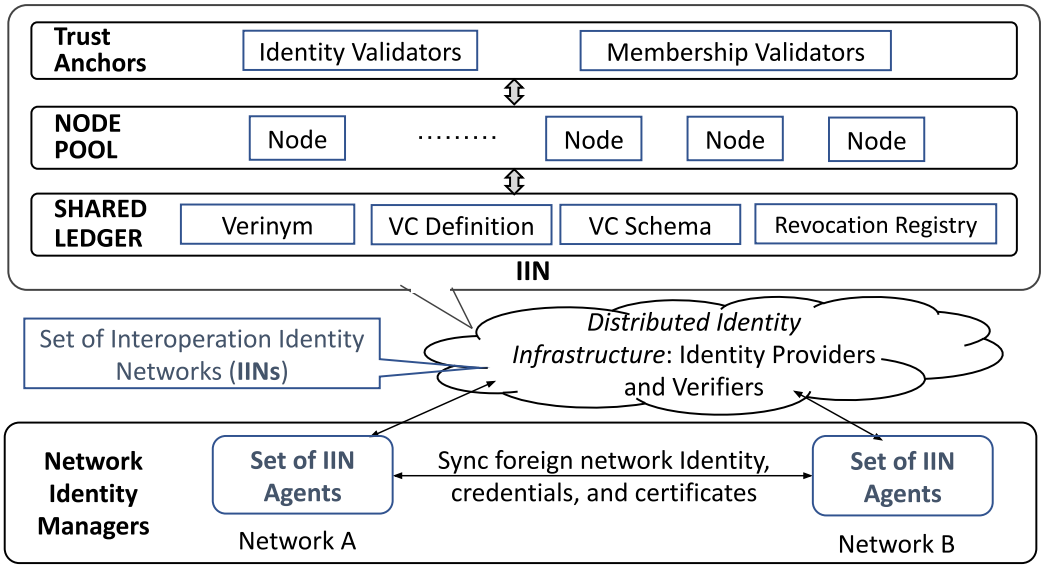


Figure 4.2 Architecture to enable identity plane exchanges

- No change should be required in a network's regular operation, nor should it be burdened with onerous additional configurations.

4.2.1 Building Blocks

We rely on existing decentralized identity management concepts and tools to serve as building blocks for our solution.

- **Decentralized Identifiers (DIDs)** is a W3C draft [25] for self-sovereign identity (SSI). A DID is a URI that resolves to a DID Document that contains information about its subject like aliases and pseudonymns. It can contain public keys to authenticate the subject's signature and service endpoints for communication with the subject to obtain verifiable credentials (see further below). We assume each network participant possesses a DID, decoupling its external identity from its network affiliation.
- **Verifiable Credentials (VC)** is W3C specification on cryptographic digital credentials based on DIDs, used to certify claims about their holders [1], who can then prove their claims to third parties using **Verifiable Presentations (VP)**.

- **Distributed Verifiable Data Registry (VDR)** is a data registry that maintains identity records in a distributed ledger. Examples are Hyperledger Indy [34] and Sidetree [143]. Indy, built on a blockchain network, maintains DID records through consensus. (*Note*: our design will accommodate centralized DID registries too but we recommend Indy-like networks for optimal levels of decentralization.)
- **Identity and Credential Messaging** in a platform-neutral and interoperable protocol is required for peer-to-peer exchange of DID records, VCs, and VPs among issuers, subjects and verifiers. An example (though our design can accommodate any equivalent) is the *DIDComm protocol* in *Hyperledger Aries* [35], which facilitates such exchanges with support for encryption using keys and service endpoints stored in registry DID records.

We refer the reader to Chapter 2.3 for a more detailed discussion on DID, VC, VDR, and DIDComm. In the following subsection we describe the decentralized identity plane architecture.

4.2.2 Architecture

Two sets of modules are required to realize an identity plane protocol (Figure 4.2) - (i) a set of networks separate from the interoperating networks, collectively called *Distributed Identity Infrastructure*, in which identity and credential records are maintained, and (ii) a set of agents within an interoperating network, collectively called *Network Identity Managers*, each acting on behalf of a participant, syncing and validating identities across network boundaries, and recording foreign identities on the local ledger.

Distributed Identity Infrastructure

This is a *cloud* of what we term **Interoperation Identity Networks (IINs)**, which collectively provide a common root of trust for networks to sync identities and certificates. Each IIN consists of a distributed VDR, which, without loss of generality, is built on a public permissioned ledger allowing open queries but restricting writes to designated *trust anchors*

(*see below*). The VDR ledger, shared and maintained by the IIN's pool of nodes through consensus [34, 66], also records verifiable credentials' schemas, authentication keys for credentials' attributes, and revocation lists. Each IIN in our infrastructure cloud enables trustworthy validation of certain networks' memberships by maintaining DID records for their participants and schemas and keys for issuing and verifying credentials.

Trust Anchors: An *IIN* does not have a centralized source of trust. Instead, a trust basis is collectively created by *trust anchors (TAs)*, entities that possess ledger write privileges, allowing them to issue DIDs to entities and maintain DID records on the IIN ledger, and further, issue VCs to DID owners. Some TAs are bootstrapped into an IIN, but any other DID owner can be assigned anchor privileges by an existing anchor too, thereby creating a trust hierarchy. Each TA in an IIN vouches for certain network members' real-world identities and attests to their participation (membership) in interoperating networks. A TA, in effect, is like a conventional identity provider but in our architecture, maintains identity records in a shared ledger with other TAs.

IIN TAs come in two varieties. ***Organization identity validators (OINs)***, who already possess well-known real-world identities, are responsible for associating a network participant's DID with its real-world identity on an IIN ledger (through consensus). (*Note*: The W3C draft proposal does not mandate a DID's automatic association with a real-world or legal identity upon creation [25, 144], so we need OINs to make such associations explicit.) The presence of a DID record for a network participant with such an association implies that its identity is vouched for by one or more TAs of that IIN. ***Participant membership validators (PMVs)*** are responsible for validating the membership of a DID owner in a given permissioned network. Enterprise blockchain networks formed by mutual agreement among its participants have no single central authority that can certify the network's structure or its members. Therefore, proving participation of an organization requires attestation by one or more PMVs, which, like OINs, are reputed in industry or well-known as consortia representatives. For example, either IBM or Walmart, both reputed entities, could act as validators for the membership of the IBM Food Trust [27] network, in whose launch and governance they both played key parts. These trust anchors issue verifiable credentials to participants attesting their network memberships and also revoke them when organizations leave networks. An organization may hold multiple VCs attesting its memberships in multiple networks. In a cross-network data sharing session, it can construct a VP proving

its membership in either counterparty network without revealing its memberships in other networks, thus ensuring privacy [1].

IIN Artifacts: the following are maintained on the ledger:

- *Real-world DIDs* attesting the real-world identity of an organization (network participant).
- *Membership VC schema and authentication key* - A membership VC for an organization proves its participation in a given network. The VC contains its DID and the name/identity of a network it is a member of as attributes (claims). This *Membership VC schema* is recorded on the IIN ledger, and every organization's VC must adhere to it, though different VCs may use different encodings and cryptographic algorithms. The public key used for authentication of this VC is also recorded on the ledger and used to validate membership claims made by the organization using a VP.
- *Memberlist VC schema and authentication key* - A memberlist VC consists of a name/identifier for a given network and a list of the network's participants' DIDs as schema attributes. Memberlist VCs are issued by PMVs and used by interoperating networks to fetch each others' list of participants, after which each participant's membership VC can be fetched and validated.
- *Revocation Registries* - When a blockchain network's member leaves, the *Membership VC* indicating its affiliation with the network must be revoked. We use cryptographic accumulators [145] as revocation registries, allowing membership presence checks without revealing the entire list of members. Each PMV registers a separate revocation registry on the IIN ledger. This registry is updated when a VC is revoked, and is also looked up by entities validating a verifiable presentation made by a Membership VC holder.

Network Identity Managers

These components lie within the trust boundary of a permissioned network, one or more acting on behalf of each participating organization. A network identity manager is respon-

sible for identity-syncing, i.e., (i) presenting its own identity and membership credentials to a foreign network, and (ii) correspondingly validating the membership credentials of a foreign network's members, and fetching and storing their certificates in the local ledger for data plane interoperation. We will henceforth refer to these managers as IIN Agents as they rely on IINs and their trust anchors for discovery and connections with foreign networks.

IIN Agent: This is responsible for registering the real-world DID of a network participant on an IIN and obtaining a Membership VCs from a TA of that IIN. It communicates with IIN Agents of foreign network participants using a confidential web-based channel to prove its own identity and validate their identity and membership claims using VPs. Furthermore, *IIN Agents* exchange their network-issued identity and certificates using self-signed VPs that can be verified against their real-world DID. Once verified, they configure these certificates in their network's shared ledger through consensus. This maps an organization's decoupled SSI (real-world DID) to its network-issued identity, which ultimately makes proof verification possible in the data plane for interoperation.

Ledger Artifacts: Each permissioned network maintains the following policy configuration for identity-sharing, trust, and interoperation:

- *Interoperation network list:* list of foreign networks with which the local network is willing to interoperate.
- *Trust list:* IINs and specific TAs within that are trusted for identity and membership validation of foreign networks.
- *Foreign network identities:* identities of organizations participating in foreign networks and their network-issued credentials (typically certificate chains).

4.2.3 Identity Exchange Protocol

Figure 4.7 illustrates the end-to-end flow in our canonical identity plane protocol to discover and sync identity and credentials across an example pair of networks (Network A and Network B) and a single IIN. In this example, Org1 and Org2 in Network A are learning about the identity and membership of Org3 in Network B so that the networks can

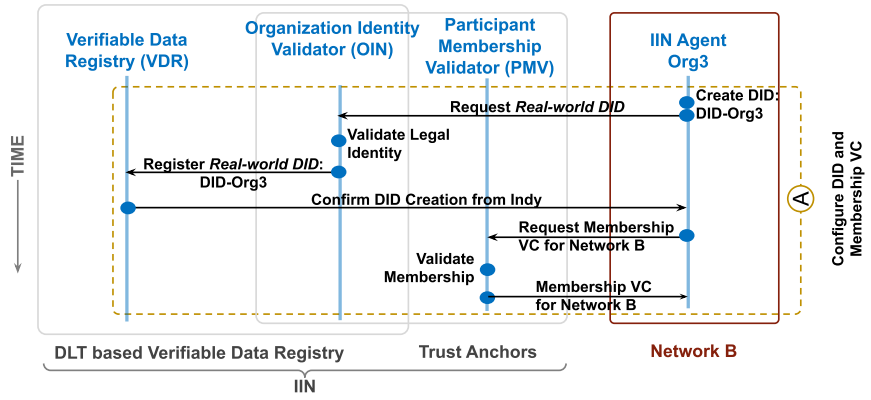


Figure 4.3 Phase (A) of Identity Exchange Protocol - Configure DID and Membership VC

share ledger data with each other. Note that by repeating these steps, identity information of any of the other organizations in Network B can be discovered, fetched, validated and configured in Network A. We describe the protocol in four phases (A) Configure DID and Membership VC of the organization whose identity would be configured in the foreign network - Figure 4.3, (B) Validate DID and Membership of foreign network organization - Figure 4.4, (C) Fetch blockchain network specific identity information - Figure 4.5 (D) Update identity information in ledger - Figure 4.6.

Phase *(A). Configure DID and Membership VC:* Org3, that is the organization whose identity and membership needs to be configured by the foreign network first creates a DID and requests an Organization Identity Validator (OIN) to attest to its identity and register it as a real-world DID (Figure 4.3). Once this DID is registered in the IIN, Org3 must get a Membership VC issued to it by a Participant Membership Validator (PMV) of that IIN. The PMV validates the request using some out-of-band validation procedure, issues the VC, and updates the revocation registry accordingly.

Phase *(B). Validate DID and Membership:* Before starting the validation process, Org1 and Org2 need to know who the participants of Network B are. Org1 requests a Memberlist VC from the PMV associated with Network B, which returns a self-signed VP. After Network B's participants' real-world DIDs are known, Org1 resolves Org3's DID Document from the IIN ledger. This DID Document contains the service endpoint which is then used by Org1 to request Membership VP from Org3. Org1 then validates the VP received using the Membership VC schema, authentication key, and the revocation list, all fetched from

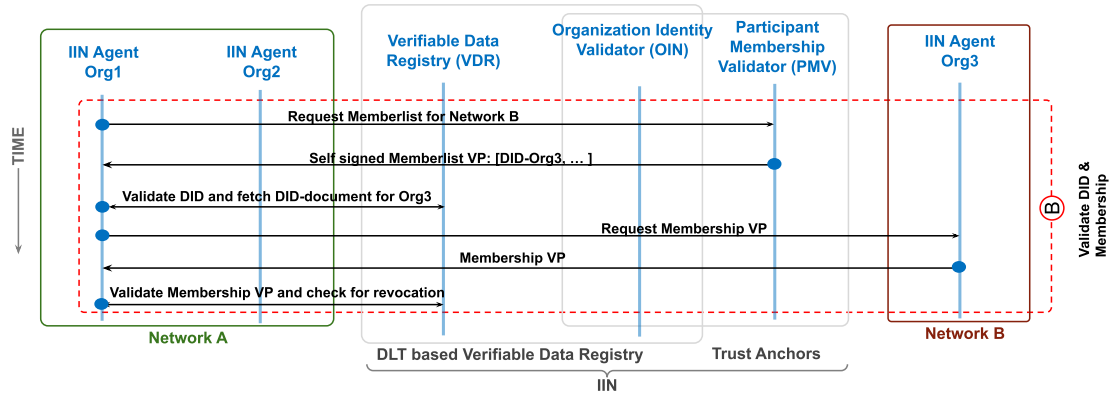


Figure 4.4 Phase (B) of Identity Exchange Protocol - Validate DID and membership of foreign network organization

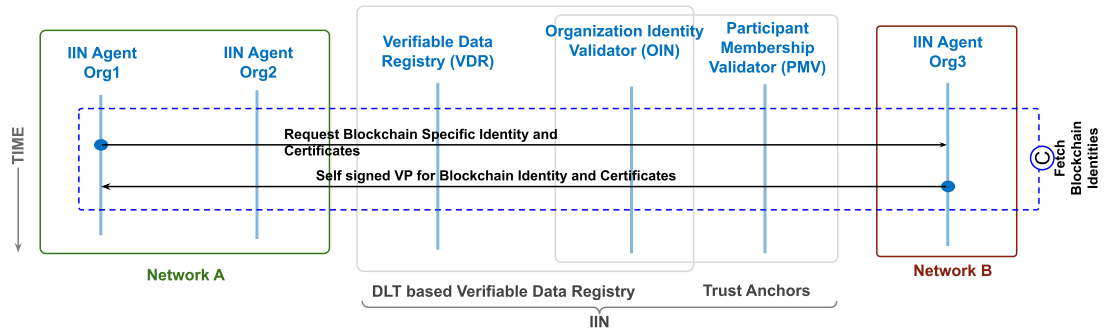


Figure 4.5 Phase (C) of Identity Exchange Protocol - Fetch blockchain network-issued identity information

the IIN ledger (see Figure 4.4).

Phase *(C). Fetch Blockchain Identity Information:* At the end of Phase (B), Org1 has already validated the DID and Membership credentials of Org3. But the corresponding blockchain network specific identity and certificates of Org3 are required for interoperation in the data plane. So Org1 requests Org3 for its network-issued identity and certificates, which Org3 returns in the form of a self-signed VP that is validated against Org3's real-world DID's authentication key. (Figure 4.5)

Phase *(D). Update Identity in the Ledger:* Though Org1 now has verified the identities of Org3, it cannot record it on Network A's ledger without a consensus among the network's participants. Therefore, Org2 independently carries out steps (B) and (C) above and endorses Org1's request to commit Org3's identity to the blockchain (using a smart contract

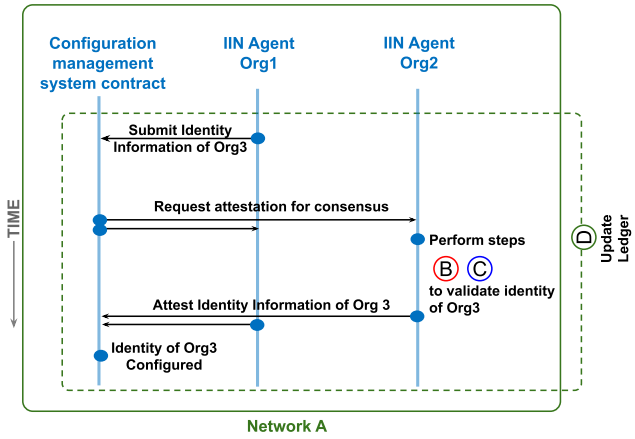


Figure 4.6 Phase (D) of Identity Exchange Protocol - Update identity in the ledger

transaction). Thus, no single participant of a network can unilaterally manipulate the local record of the identity of a foreign network's participant (Figure 4.6).

The end-to-end flow of the Identity Exchange Protocol, from phases A to D, are depicted in Figure 4.7.

4.3 Use Case for Hyperledger Fabric

We demonstrate a proof-of-concept implementation of our protocol by augmenting the two-network use case in Abebe et al [15]. We started with the already developed scaled-down versions of the trade logistics network, TradeLens[1] [10], and the trade finance network, We.Trade [135], namely *Simplified TradeLens (STL)* and *Simplified We.Trade (SWT)* respectively. Here, each network runs Hyperledger Fabric peers, membership service providers (MSP), an ordering service, and application (client-layer) components, in Docker containers. Each network was equipped with a relay and two system contracts: Configuration Management and Data Acceptance Chaincode (CMDAC) and Exposure Control Chaincode (ECC).

STL consists of a Seller and a Carrier organization, as illustrated in Figure 4.8 each running a peer and a CA (serving as MSP). Consignments are created and dispatched in

[1](Discontinued at the beginning of the year 2023)

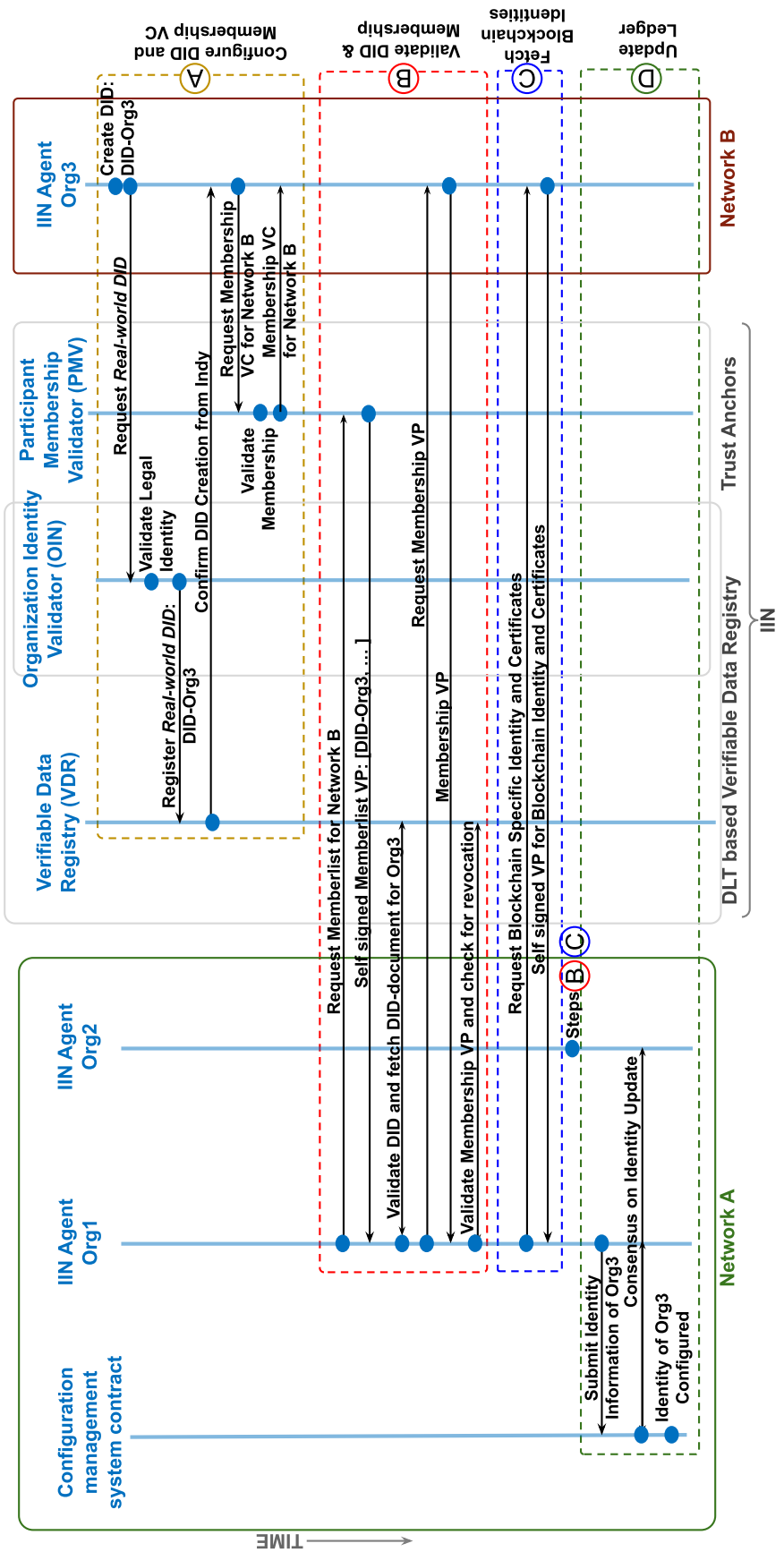


Figure 4.7 End-to-end Identity Exchange Protocol

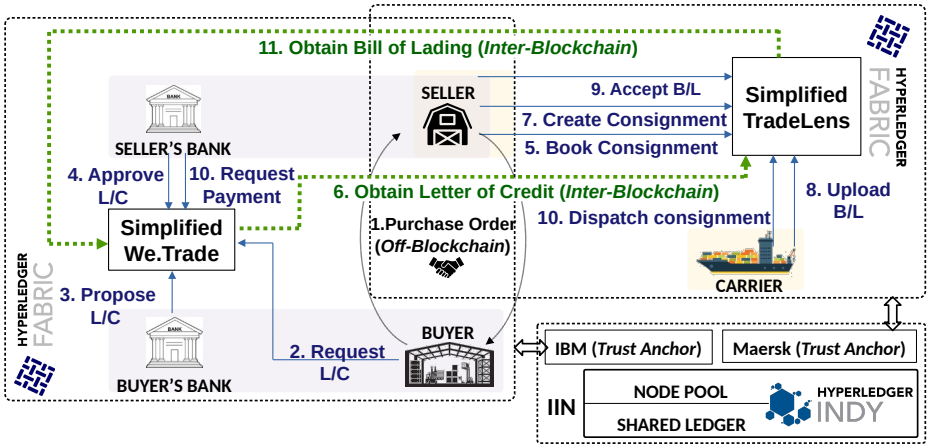


Figure 4.8 Simplified Cross-Network Trade Use Case

the workflow, with a bill of lading [16] (B/L) recorded on ledger. SWT consists of a Seller and a Buyer organization, each with 2 peers and CAs, running a letter of credit [146] (L/C) management workflow. SWT application clients include the same Seller that is a member of STL, a Buyer, and the Seller's and Buyer's banks. The interoperation steps are: (1) transfer of L/C from SWT to STL as a prerequisite for consignment creation and (2) transfer of B/L from STL to SWT for payment obligation enforcement.

4.3.1 Distributed Identity Infrastructure

We used Hyperledger Indy to implement an IIN with trust anchors. Though other DID registries and DID providers exist, Indy is the most mature and offers all the features described in Section 5.2. Indy maintains DID records on a public permissioned ledger shared by a pool of nodes running a consensus protocol [34,66]. Trust anchors called *stewards* are bootstrapped into an Indy network within the genesis block, and these stewards can assign trust anchor privileges to other DID owners too: Indy supports TAs of the OIN and PMV categories out-of-the-box. DID records contain service endpoints, credential schemas, and authentication public keys (called *credential definitions*). Real-world DIDs are called *verinynms* (anonymous *pseudonyms* are also supported), and a TA (OIN) can register a verinym on the Indy ledger using a special *NYM transaction*. TAs (including stewards) and IIN Agents (*see below*) are implemented using the companion Hyperledger Aries [35] framework, which enables confidential peer-to-peer communications among Agents and

TAs.

For proof-of-concept, we deployed a single IIN: an Indy network bootstrapped with 4 independent Sovrin stewards [147]. Two trust anchors were enrolled in the IIN by the stewards: one in the name of IBM to represent the SWT consortium and another in the name of Maersk to represent STL (see Figure 4.8). (Note that IBM and Maersk are initiators and major players in the real We.Trade and TradeLens networks respectively, and are therefore realistic sources of trust.) A single IIN with two trust anchors is sufficient to demonstrate operational mechanics in a proof-of-concept; in a production implementation, we will likely have more diversity but the mechanisms used will be identical to what we demonstrate here. The SWT Seller and Buyer register verinyms with, and obtain Membership VCs from, the IBM anchor in the IIN; likewise, the STL Seller and Carrier register and obtain theirs from the Maersk anchor.

4.3.2 Fabric Network Organizations and Identity Providers

In a Fabric network, identity is independently managed within an organization by one or more *membership service providers* (MSPs) [141], implemented as a set of Fabric CA Servers [148] (CA: *Certificate Authority*). Multiple root and intermediate CAs can exist within an organization, creating trust chains. Each peer or transaction-submitting client *enrolls* with an MSP (one of the Fabric CA servers) in their organization to obtain a unique identity and X.509 certificates for transaction signing. Organizations are then linked together on a channel when a configuration block containing their respective MSPs' root and intermediate CA certificates is appended to that channel's blockchain.

Every valid transaction in a block must carry a set of peer signatures that satisfies an *endorsement policy*. Likewise, in a data-sharing instance, any data shared with an external network can only be deemed valid if it carries a set of peer signatures that satisfies a *verification policy*. But proof verification (i.e., signature validation) requires the destination network to possess the source network's organization list and the certificates of its MSPs. Below we show how IIN Agents embedded within a Fabric network enable proof verification by fetching certificates and creating identity records on the ledger using the CMDAC contract.

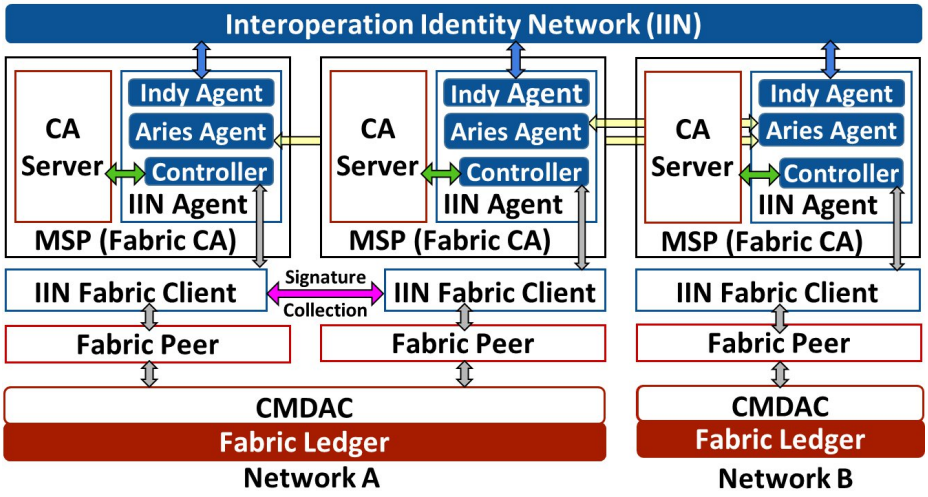


Figure 4.9 IIN Components and Connections for Fabric

4.3.3 IIN Agents within a Fabric Network

An IIN Agent represents an organization outside its network, and hence is designed to be an extension of that organization's MSP. An organization typically uses a single MSP in production, but if multiple MSPs are used, representing organizational units, we can use a different Agents for each. Logically, the Agent functionality ought to be performed by a Fabric CA Server. Rather than modifying the Fabric CA code, for implementation and deployment convenience, our IIN Agent is built as a decoupled service exposing an API for communication with IINs (Indy networks), IIN Agents of other local network organizations, and IIN Agents of foreign networks' organizations (see Figure 4.9). The IIN Agent also has client privileges and can submit transactions to the CMDAC.

An IIN Agent is composed of three modules. An *Indy Agent* built using with `indy-sdk` [149] is used to connect to an IIN's Indy pool and query DID/credential information. An *Aries Agent*, implemented using `ACA-Py` [150], is used to communicate with IIN trust anchors for verinym, VC, and VP requests. It uses DID service endpoints and credential definitions (authentication keys) for encrypted communication. While handling VCs and VPs, the *Aries Agent* calls the *Indy Agent* to fetch/update credential schema and definitions, and revocation lists. The *Controller*, implemented using `Node.js` orchestrates the identity initialization, exchange and validation flow as described in Section 4.2.3. It is responsible

for fetching its associated MSP's latest root and intermediate certificates for sharing with foreign networks. These three modules run within a single Docker container. Lastly, an *IIN Fabric Client* application built on the `Fabric SDK` [151] updates foreign networks' identities and certificates on the channel ledger using CMDAC transactions. This app, running within its own container, takes transaction requests from the *Controller* and executes an application level signature collection flow (see Section 4.3.4) before invoking the CMDAC.

4.3.4 Protocol: Syncing Foreign Identities through Consensus

Our protocol implementation follows the steps described in Section 4.2.3 verbatim except for the final step in which a foreign network's identities are recorded onto the local ledger. This step requires DLT-specific mechanisms, and we will show how they were implemented in Fabric. As an example, we consider the scenario where SWT is trying to fetch and sync the certificates of the Carrier organization in STL.

Recording the STL Carrier's certificates on the ledger creates a shared truth for the entire SWT network, enabling its peers to refer to those certificates in a data transfer session, either for access control or proof verification. Allowing the CMDAC to directly query IINs or foreign networks' agents to fetch these certificates would create a non-determinism hazard. Therefore, we use an application-level flow involving IIN Agents for deterministic invocation of CMDAC.

The SWT Buyer IIN Agent initiates this flow by validating the STL Carrier's Membership VC with the IIN, and subsequently fetches the Carrier's certificates from the Carrier's IIN Agent (whose service endpoint is part of the Carrier's DID record). The Buyer IIN Agent then sends these certificates to its Fabric Client app, which then prepares a *signature collection request*, and sends it to the Fabric Client of the SWT Seller IIN Fabric Client app. The latter validates the Carrier's certificates after consultation with their own IIN Agents (who have presumably fetched those certificates too), and then counter-signs the request on behalf of its organization. The Buyer aggregates the responses (only one here) and submits it as input in a CMDAC transaction. The chaincode checks for the presence of valid signatures from every SWT organization (here: Buyer and Seller) before approving the update of the STL Carrier's identity state on the ledger. Note that this update is idempotent, and

can be carried out concurrently by the IIN Agents of both Buyer and Seller. Also, if the Buyer's and Seller's IIN Agents' copies of Carrier certificates are not in sync, the signature collection flow will fail and must be retried.

This protocol replaces the naive implementation in Abebe et al [15] where organizational identities and root and intermediate certificates were fetched out-of-band manually. The identity plane exchange we have demonstrated makes the process more secure and consensus-based. Further, any changes in organizational memberships or certificate belonging to an organizations can be determined and synced automatically through periodic queries made to the IIN registry (for membership and revocation lists) or whenever a proof validation fails because of expired certificates.

4.4 Analysis

We now analyse our system's ease of use and extensibility, and discuss its limitations with a view to future improvements.

4.4.1 Generality and Flexibility

Our design consists of two distinct sets of components: *1) Distributed Identity Infrastructure* shared by interoperating networks but existing outside them, namely the IINs, and *2) Network Identity Managers* components that lie within networks, namely the IIN Agents. IINs are built using state-of-the-art industry standards (Indy, DID, VC), though the specification is independent of a specific technology. IIN Agents are DLT-specific and decentralized within a network, lying within the scope of a network's participant/organization. In fact, different organizations may implement their own versions of IIN Agents and replace them independently; an Agent just needs to expose the API we have specified earlier in this chapter.

4.4.2 Security

We evaluate the security of our protocol against the standard CIA triad model [152].

Confidentiality: The DIDs of network participants are themselves public by necessity, as the DID Registry (IIN) is a public permissioned network. But a DID by itself only reveals the existence of an organization that participates in a network without revealing anything else about that organization, like membership information, which are known only to identity owners and IIN trust anchors. Also, the intra-network certificates are shared point-to-point among IIN Agents on a need basis and are thus kept confidential from everyone outside the interoperating networks.

Integrity: Identities are registered in an IIN using a fault-tolerant consensus protocol (Indy typically uses RBFT [66]), thereby ensuring a high level of integrity. Trust anchors maintaining Memberlist VCs are assumed to have reputations and are trusted by organizations belonging to a consortium; further, identities and VCs are attested by signatures using keys registered in IINs. Integrity violations are therefore unlikely but can be easily detected, allowing organizations to select more trustworthy anchors.

Availability: The availability of identity records depends on the size of the IIN; the more the number of nodes in an Indy pool, the higher the availability. An IIN Agent or an IIN Trust Anchor by itself can be a point of failure, but this can be mitigated by adding redundancy.

4.4.3 Privacy

Our proposed identity plane deals with issuing, presenting, and verifying digital identities and other credentials of organizations. Naturally, the implication of using the Interoperation Identity Network (IIN) infrastructure on privacy is an important consideration. In our proposed IINs, only decentralized identifiers (DID) are stored in the verifiable data registries. These DIDs by themselves are indistinguishable from any random identifier and does not

contain any personal or sensitive information. The validation of credentials such as identity and membership of organizations are instead carried out through the Verifiable Credential (VC) protocols. The W3C recommended Verifiable Credentials specification has the goal of providing cryptographically secure, privacy respecting, and machine-verifiable digital credentials [1]. A verifiable credential issued by an issuer is only available to its holder, and only the holder is capable of presenting it to any verifier through a Verifiable Presentation (VP). A VC attests to certain claims that are typically about the its holder. However, this VC is never stored as an artifact in the IIN.

A claim about an organization or the VP proving that claim is communicated to a verifier through a device-to-device (D2D) communication channel that is typically end-to-end encrypted. Even then, this process of verifiable presentation has different privacy preserving implementations. One such implementation in Hyperledger Indy [34] uses zero-knowledge proofs to present credentials without revealing the exact claim. Instead, the VC is used to prove a predicate that is provided by the verifier. One example of such claim is age, where without revealing the actual age, a presenter can prove that its age is more than 18 years through a zero-knowledge proof implementation of VP. As a result, the IIN infrastructure in no occasion obtains or saves any sensitive claim information of the participants. Moreover, through the use of zero-knowledge proofs, sharing sensitive claims with the verifier can also be avoided. Therefore, our proposed identity plane infrastructure respects the right to forget, which is an important property of a privacy preserving system.

Taking one step further in this direction, we consider that revealing the trust anchors between the two parties that are trying to validate each other claims can also lead to a breach of privacy and leak of sensitive information. In practice, trust anchors are often well-known organizations such as government bodies, large companies, influential non-government organizations, etc. Having such organizations as trust anchors indicate some kind of positive association with them. As a result, privacy-preserving negotiation of such trust anchors is required. We discuss this problem in the next chapter (Chapter 5) and propose solutions for the same.

4.4.4 Ease of Extensibility

Network Identity Managers

An IIN Agent runs as part of a network, but only portions of it needs to have a DLT-specific implemenation. Examining the protocol steps in Section 4.2.3, we see that step A involves the Agent communicating with IINs and Trust Anchors using a standard API. Similarly, steps B and C involves communication between IIN Agents across network boundaries, again using a standard interface. Only Step D, which involves updating the local ledger via a smart contract transaction must be DLT-specific. Hence, an IIN Agent can be mostly built using off-the-shelf components. The transaction submission component must be DLT-specific, as was the IIN Fabric Client described in Section 5.4. The equivalent of this in Corda would be a CorDapp [153] and in Hyperledger Besu would be a Dapp [154].

Distributed Identity Infrastructure

Though our IINs are implemented using Indy and IIN trust anchors using Aries, their specifications and interfaces are based on W3C standards for DIDs and VCs and VPs. Therefore, they can easily be ported to other verifiable data registries that follow the same standard (e.g. Sidetree [143]).

4.4.5 Possible Technical Improvements

A trust anchor representing a consortium or unilaterally issuing real-world DIDs to organizations is the only centralized component in our implementation. But further decentralization is possible by requiring more than one TA to vouch for a network participant; e.g., using a smart contract in the IIN. Collaborative models, where TAs (e.g. representing Fabric MSPs) corroborate each other using signatures can also enhance safety and liveness of identity plane protocols.

Decoupling of network participants identities from their IIN identities presents another

challenge: syncing the two sets to ensure that networks possess up-to-date info for data plane operations. In general, this only affects liveness and not safety, because proof verification failures can be handled by re-synchronizing identities using strategies like polling to event triggers. While polling may be slightly inefficient from a communication standpoint, it provides a higher level of assurance (depending on the polling interval). Additional watchdogs may be needed to handle all cases, in case of events, in both the identity and data planes.

4.5 Discussion on Real-World Deployment

In this chapter, our objective is to address the technical gap present in blockchain identity management systems and membership protocols of permissioned blockchains, which poses a significant challenge to achieving seamless interoperation across heterogeneous blockchain networks. Notably, we observe that enterprise blockchain networks are established through mutual agreements among real-world organizations, including companies, institutions, government bodies, and non-governmental organizations. These organizations possess their distinct legal identities, which are bestowed upon them by the respective "state", that is governmental authorities. These identities might manifest in the form of certificates of incorporation, tax identification numbers, employer identification numbers, and similar credentials. By relying on these state-provided credentials, the organizations establish a consortium with legally binding terms, thereby constituting a permissioned blockchain network. Moreover, to establish the interoperation process, two or more consortiums reach an agreement to collaborate in terms of data and digital asset transfers, facilitating communication and cooperation between otherwise isolated permissioned ledgers.

The challenge arises in translating the aforementioned offline decisions and agreements into the digital ecosystem of interoperating blockchain networks. Concretely, three aspects of offline legal documents need digital representation - (i) the identifier of an organization, (ii) the legal identity or real-world identity of an organization, and (iii) digital credentials representing the claims and attestations to those claims. At present, without any distributed

identity infrastructure, the process of forming a blockchain network, and more so the process of establishing interoperability between two networks is entirely a manual process where all the identifiers, identity, and credentials have to be manually configured into different ledgers while conforming to their heterogeneous representations and protocols.

Through our proposed distributed identity infrastructure (Figure 4.2), we provide a homogeneous and unified, yet decentralized platform to issue, present, validate and represent digital identifiers and credentials including identities. The Interoperation Identity Networks (IINs) are collections of trust anchors which are identity and other credential issuers, along with the necessary infrastructure including a verifiable data registry for maintaining the digital identifiers. In practice, the offline legal identities, credentials, agreements and terms are to be translated to their digitial representation in the IINs. This process of digitizing credentials would however not be plausible without the IIN architecture, since every blockchain system, as well as other digital systems have their own heterogeneous representations, protocols, and algorithms for identity, membership, and credentials. As a result, our proposed identity plane acts as a bridge between the legal real-world identities and credentials to the digital ledgers. More importantly, the IIN infrastructure does not rely on any central authority to maintain the credential and identity assets, instead it is realized through distributed ledger based registry, decentralized identifier (DID) and verifiable credential (VC) protocols.

Organization identity validators: In practice, the organization identity validators (OIN) in most cases are the government bodies. However, large enterprises are often multi-national entities that are registered in several countries. As a result, they have multiple identifiers and identity credentials which often differ from country to country. Through the DID and VC specifications used in the IIN, such multi-national entities can create a single identifier, that is a single DID. The different government bodies of different countries then may issue separate verifiable credentials attesting to the claims of the country-specific identifiers, as well as country-specific identities such as subsidiaries. During identity interoperation, a counter-party organization trying to validate an identity may rely on the trust anchor that it trusts. Such trust anchor may be the specific country's government that both organizations are registered in. As a result, the identity plane with the help of the IIN infrastructure forms a trust basis between organizations having multiple same or different legal identity providers. Moreover, the single DID provides a unique digital identifier to

the organization which is agnostic of the different legal identity and credential providers.

Participant membership validators: As mentioned earlier, permissioned consortium blockchains are formed through mutual agreements among real-world legally registered organizations. Although the deliberations leading to such agreement often takes place "offline", the outcome of the agreement is a consensus on the structure of the network along with the members included as participants. Thus, after network formation, each participant issues a verifiable credential to the other participants attesting to their membership in the newly formed network. Therefore, all participants are potential participant membership validators (PMV). In practical scenarios however, the consortium blockchain networks are often led by a well known large multi-national organization. For example, IBM and Walmart, both reputed entities of the IBM Food Trust [27] network are PMVs for other participants of that network. Because of their status and reputation in global trade and commerce, the organizations in the counter-party network trying to interoperate would most likely trust them as trust anchors to validate the membership of other organizations in IBM Food Trust. This however, brings some degree of centralization to the system. But none the less our proposed identity plane offers all possible options to validate the participant memberships without relying on any central authority.

Identity exchange and dispute resolution: The only hurdle in deploying the identity exchange protocol (Section 4.2.3) for permissioned blockchain interoperation is to make the legal identity providers such as the organization identity validators and participant membership validators to issue their credentials as VCs in the identity plane. This is not a technical challenge but a challenge to push the adoption of digital credentials to all the countries, government bodies, companies, and other organizations. Once the credentials are represented as VCs in the IINs, the identity exchange protocol can proceed and set up cross-chain interoperation without any manual intervention. In case a dispute takes place, those can be handled through the existing legal process since the credentials issued in the identity plane are just digital representations of the real world legal credentials and thus are technically equally enforceable.

In summary, our proposed identity plane infrastructure and protocols act as a bridge between physical legal identities, credentials, and agreements, and the digital system of blockchain networks. The same legal entities certifying identities and attesting member-

ships in the offline processes act as OINs and PMVs in the IIN. The most significant challenge towards adoption of the proposed identity plane is the fact that its usefulness is only significant if all possible trust anchors including government bodies agree to adopt the verifiable credential specifications for as a means of issuing the credentials.

4.6 Summary

Interoperation for data sharing between permissioned blockchain networks running related business processes requires the networks to have the ability to identify each others' participants and validate their claims/proofs. We have described a way of reasoning about such protocols, separating identity concerns from data and policy concerns into a different communication plane. To give networks the ability to prove memberships, cross-validate identities, and share certificates, we have designed a DLT-agnostic architecture and protocols based on self-sovereign identity and verifiable credential concepts. A proof-of-concept implementation was demonstrated, linking two Hyperledger Fabric networks. This consisted of an identity registry (IIN) built on Hyperledger Indy and agents built on Hyperledger Aries exchanging certificates across network boundaries in a peer-to-peer manner.

A prerequisite for the identity exchange protocol presented in this chapter is the existance of a common trust anchor between the two interoperating permissioned networks, and more specifically between the participant organizations of the same. The step B of the Identity Exchange Protocol (Figure 4.7) can succeed only if there is a common trust anchor between the verifying organization and the proving organization. The common trust anchor is the entity which is trusted by the verifying organization and has issued a VC to the identity proving organization attesting claims about the later's identity and consortium membership. However, the question of how such a common trust anchor (if any) can be negotiatied between these to organizations executing the protocol is left unanswered in this chapter. This problem of trust anchor negotiation turns out to be a challenging one and we focus on the same in the following chapter.

Chapter 5

Cross-chain Negotiation of Common Trust Anchors

Permissioned networks built on the blockchain or distributed ledger technology (DLT) restrict memberships and control access to their ledgers for privacy and performance reasons. Yet, real-world consortium networks built for specific and limited business purposes soon discover compelling needs to link their processes (smart contracts) and allow assets and state to move seamlessly across their boundaries [15, 18]. Blockchain interoperability aims to avoid fragmentation and enable scale in the blockchain ecosystem while allowing business consortiums to retain autonomy. Enabling interoperability by sharing the ledger state along with authenticity proofs across network boundaries is a reasonably well-established concept [15, 18, 20, 137, 155, 156]. However, the infrastructural scaffolding around state proof generation and validation may vary widely.

We consider a model where one autonomous permissioned network may respond to a remote query for the state from another with proofs; proof generation and validation at the ends occur through their networks' respective consensus and smart contract enforcement mechanisms [15, 92, 157, 158]. In permissioned networks, proofs must be constructed as quorums of signatures from participating organizations. Hence, the networks must be familiar with each others' identity and certificate providers. As we proposed in the Chapter 4, *Verifiable credentials* (VCs) [1] along with existing decentralized identity infrastruc-

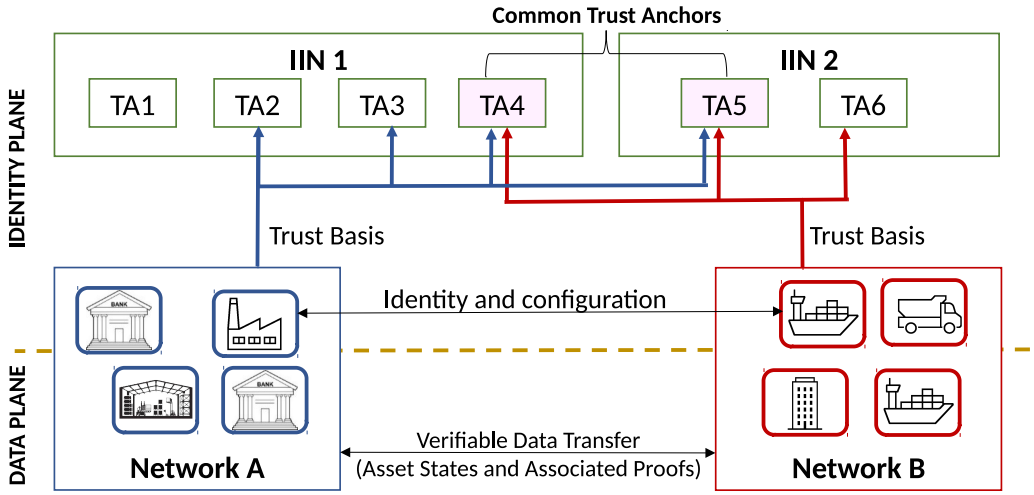


Figure 5.1 Two interoperating networks with some common TAs

ture [25] can be utilized to sync each others' certificate chains [159], which involves trusted authorities (*trust anchors* or TAs) associated with decentralized identity registries to issue credentials to organizations that certify their real-world identities as well as their memberships in a consortium network. Although this protocol serves as a canonical reference solution for networks to establish trust a priori without using any shared third-party infrastructure other than existing decentralized identity registries and TAs, it also opens up a new challenge as follows.

As illustrated in Figure 5.1, the credentials used by organizations to prove their real-world identities and network memberships may be issued by several different TAs associated with different registries. But such a credential is applicable only if the TA that issued it is also trusted by organizations in a counterparty network. For example, considering two networks $\mathcal{N}_1$ and $\mathcal{N}_2$, the members of both the networks must have at least one common trust anchor $\mathcal{T}$ among their individual list of TAs, whereby the certificates issued by $\mathcal{T}$ for the members of $\mathcal{N}_1$ can be verified by the members of $\mathcal{N}_2$ and vice versa. (*In the absence of any, organizations and network consortiums will have to go through logistical hurdles to find reputed certification authorities and obtain new credentials!*) This determination is straightforward if both parties share their TA lists with each other. But an organization may not be comfortable sharing its affiliation with TAs that the counterparty is not affiliated with, as these are personal associations, and their revelations may compromise privacy. Therefore, the problem we would like to solve is as follows: two parties must mutually

identify only the TAs they have in common (i.e., those that certify one party and are trusted by the other) without having to reveal their respective associations with any other TA.

For example, consider the scenario where a trade logistics network (TLN) and a trade finance network (TFN) must interoperate to share state, as illustrated in [15]. The organizations in the logistics network are certified by (i.e., possess VCs from) the Maersk Shipping Company and the American Bureau of Shipping (ABS). The organizations in the finance network trust the ABS as well as the Mediterranean Shipping Company. Organizations in each network can safely reveal their associations with ABS and use it as a common TA. But they may not want to reveal the identities of their other TAs (in this case, Maersk by TLN or Mediterranean by TFN) as that may compromise the competitive advantage of both the organizations and their trust anchors. In this case, Maersk and Mediterranean are competitors in the shipping market, and they would have an interest in learning who else their clients are working with. But on the flip side, they wouldn't want their competitors to know who their clients are either. Though not apparent in this use case, affiliations of an organization, especially if they are government agencies or political organizations, or NGOs, may be sensitive. Indeed, the problem we encountered here is not limited to DLT interoperability, though we are motivated to solve it for that reason. Negotiation between mutually unknown and untrusting parties with information to reveal and privacy constraints to enforce was a challenge faced in the Internet services world in the form of *trust negotiation* and in grid computing to generate service-level agreements (SLAs) [160]. The notion of needing to keep certification authorities (rather than information and policies) private is a novel twist, but one we believe will be encountered more and more as decentralized identities and verifiable credentials grow more ubiquitous in the emerging Web 3.0 world [161].

This chapter analyzes the privacy-preserving TA negotiation problem and provides an efficient and practical solution for the problem. We first highlight a naïve candidate protocol where the TAs actively participate in the process. However, in practice, the TAs are not expected to act as mediators. We therefore propose a robust and more practical protocol that builds on classic *private set intersection* (PSI) [103] and *secure multi-party computation* (*MPC*) [162] techniques. We formally define the trust anchor negotiation problem based on the real-ideal world paradigm popular in the cryptographic literature. Our contribution then covers a candidate solution for trust anchor negotiation for supporting DLT interoperability. We formally prove the security of our protocol based on the proposed real-

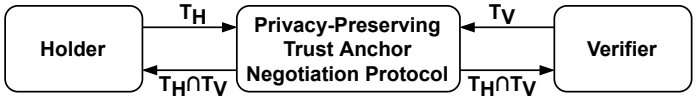


Figure 5.2 Privacy-Preserving Trust Anchor Negotiation

ideal world definition. Our solution is efficient in practice; its complexity is dominated by $\min(a, b)$ secure equality checks where a and b respectively are the number of TAs that the members in networks $\mathcal{N}_1$ and $\mathcal{N}_2$ trust. We also provide a proof-of-concept implementation [163] of the proposed protocol using MP-SPDZ framework [36] and analyze the performance of the privacy-preserving trust anchor negotiation protocol in terms of overall execution time as well as communication bandwidth requirement.

5.1 Problem Statement

Our problem can be modeled with two consortium networks ($\mathcal{N}_1$ and $\mathcal{N}_2$) that need to sync each others' certificate chains and record them to their respective ledgers for interoperation to ensue. Since each network runs this protocol independently, let us select $\mathcal{N}_1$'s syncing and recording of $\mathcal{N}_2$'s certificate chains without loss of generality. Each participant of $\mathcal{N}_1$ syncs the certificates of $\mathcal{N}_2$ independently and then records it to its ledger through consensus with other participants in $\mathcal{N}_1$.

The TAs that provide the trust basis for $\mathcal{N}_1$ and $\mathcal{N}_2$ to interoperate are accepted as certifying authorities by both their consortiums. A VC [1] issued by a TA to a participant of $\mathcal{N}_2$ attests to (i) the identity of the participant, and more importantly (ii) the membership of the participant in $\mathcal{N}_2$. Henceforth, we will refer to the participants of $\mathcal{N}_2$ as VC-holders, or *holders* for short, and participants in $\mathcal{N}_1$ as *verifiers* as they must validate holders' VCs before syncing their certificate chains. A holder may possess VCs from several different TAs. A verifier may likewise trust VCs issued by several different TAs, the list having been agreed upon collectively by its network's participants. The entire process thus involves multiple 1-to-1 negotiations between every $\langle$holder, verifier$\rangle$ pair, and in each instance, both parties must select at least one common TA for a successful sync, but without revealing the identity of any TA they do not have in common (see Figure5.2).

Definition 3 (Privacy-Preserving Trust Anchor Negotiation (PTAN)). Given that:

- There exists a finite universal set of well known TAs $\mathbb{T}$ which is publicly known to the members of all the participating networks.
- A VC holder $\mathcal{H}$ in $\mathcal{N}_2$ has a Membership VC from $\mathbb{T}_\mathcal{H} \subseteq \mathbb{T}$ TAs
- A verifier $\mathcal{V}$ in $\mathcal{N}_1$ trusts $\mathbb{T}_\mathcal{V} \subseteq \mathbb{T}$ TAs

$\mathcal{H}$ and $\mathcal{V}$ must reveal some $\mathbb{T}_C \subseteq \mathbb{T}_\mathcal{H} \cap \mathbb{T}_\mathcal{V}$ to each other while not revealing any $\mathcal{T} \in \mathbb{T} \setminus (\mathbb{T}_\mathcal{H} \cap \mathbb{T}_\mathcal{V})$.

For a successful exchange and verification of identities, revealing only one $\mathcal{T} \in \mathbb{T}_\mathcal{H} \cap \mathbb{T}_\mathcal{V}$ is sufficient, and the holder can present the VC issued by this TA to the verifier. However, in practice, a given TA may be offline or otherwise unreliable, so revealing multiple common TAs may be prudent to ensure fault tolerance. In our model, privacy is not deemed to be compromised until and unless a $\mathcal{T} \notin \mathbb{T}_\mathcal{H} \cap \mathbb{T}_\mathcal{V}$ is revealed; privacy is preserved if a single common TA or a subset of $\mathbb{T}_\mathcal{H} \cap \mathbb{T}_\mathcal{V}$ is revealed.

Verifiers without credentials: Verifiers trying to validate the identity of a remote network's participants do not need to prove or produce their own identity information. Therefore, verifiers do not require any VC from its own TAs to participate in the protocol. Though the consortium network to which a verifier belongs has its own agreed-upon set of TAs, and only this set is acceptable within it, it cannot be validated from outside the network before completing the TA negotiation process.

5.1.1 Threat Model

Malicious parties: Parties can be malicious and not follow the specified TA negotiation protocol. They can provide spurious inputs to the protocol instead of providing their own correct set of TAs in order to infer the TAs belonging to the counterparty's set that lie outside $\mathbb{T}_\mathcal{H} \cap \mathbb{T}_\mathcal{V}$. With verifiers not being required to have VCs from their TAs to participate in the protocol, they may easily fake their input set.

Non-malicious consortiums: While some participants in a permissioned consortium

may be malicious and try to extract private information during identity exchange, the consortium as a whole is not malicious. The network is thus a *trustworthy committee* [92] that is not susceptible to Byzantine failures [40].

5.2 Approaches

Holder-verifier negotiation protocols can come in two distinct flavors: (i) where one or more common TAs are willing to play an active part, and (ii) where no TA needs to be actively involved apart from issuing VCs to the holder. We believe (ii) is more typical and makes fewer assumptions about the environment, but before focusing on that, we will discuss solution approaches that assume an active TA.

5.2.1 Active participation of TAs

We describe a candidate protocol where a TA will respond to both holder $\mathcal{H}$ and verifier $\mathcal{V}$ only when it is queried by both simultaneously. The protocol starts with $\mathcal{H}$ randomly selecting some subset $\Psi_{\mathcal{H}} \subset \mathbb{T}_{\mathcal{H}}$ and $\mathcal{V}$ randomly selecting $\Psi_{\mathcal{V}} \in \mathbb{T}_{\mathcal{V}}$. $\mathcal{H}$ and $\mathcal{V}$ simultaneously send requests $\texttt{connect}(\mathcal{D}_{\mathcal{H}}, \mathcal{D}_{\mathcal{V}})$ and $\texttt{connect}(\mathcal{D}_{\mathcal{V}}, \mathcal{D}_{\mathcal{H}})$ respectively to each TA in their selected sets. Here $\mathcal{D}_{\mathcal{H}}$ and $\mathcal{D}_{\mathcal{V}}$ denote the DID of the holder and the verifier respectively. A TA $\mathcal{T}$ responds to both parties (and is thereby revealed to them as a common TA) only if it gets requests from both $\mathcal{H}$ and $\mathcal{V}$. If no such $\mathcal{T}$ exists, $\mathcal{H}$ and $\mathcal{V}$ select different subsets of $\mathbb{T}_{\mathcal{H}}$ and $\mathbb{T}_{\mathcal{V}}$ respectively, and the above steps are repeated. A subset of the list of TAs are chosen in order avoid requesting all TAs simultaneously which might be a burden for the participants as well as the TAs.

Assuming that $\mathbb{T}_{\mathcal{H}} \cap \mathbb{T}_{\mathcal{V}} \neq \phi$, a common TA is guaranteed to be eventually found by this protocol. However, this approach has several clear limitations:

(a) Every TA of a verifier needs to know apriori that it is being considered as a negotiation mediator. In practical scenarios, every VC verifier in a network will have its own arbitrary list of anchors. But all verifiers in a network must agree on a common set of TAs

they are willing to collectively trust to validate holders' VCs. Each TA in that list must be informed apriori of the fact, thereby imposing a logistical hurdle.

(b) The protocol relies on at least one commonly queried non-malicious (no collusion or refusal of messages) TA being available (i.e., not suffering from a crash fault).

(c) self-sovereign decentralized identity promises that possessing a VC excuses a TA from being involved in the verifiable presentation process. Therefore, this protocol doesn't just negate a key benefit of decentralized identifiers and VCs but may also end up overburdening TAs.

5.2.2 Without active participation of TAs

Avoiding the involvement of the TAs themselves will require a protocol to jointly compute the intersection of the input sets from both parties without revealing the inputs to the counterparty. In literature, this specific problem has been studied and is known as *Private Set Intersection* (PSI) [103]. PSI protocols allow two parties to input their respective private sets and jointly compute the intersection without revealing any information about the elements that are not in the intersection. Therefore, PSI enables a holder $\mathcal{H}$ and a verifier $\mathcal{V}$ to compute $\mathbb{T}_{\mathcal{H}} \cap \mathbb{T}_{\mathcal{V}}$ without revealing any $\mathcal{T} \in \mathbb{T} \setminus (\mathbb{T}_{\mathcal{H}} \cap \mathbb{T}_{\mathcal{V}})$.

However, a direct application of PSI *is insufficient to act as a black box for Privacy-Preserving TA Negotiation* because of a key part of our problem definition: the set of TAs $\mathbb{T}$, from which $\mathbb{T}_{\mathcal{H}}$ and $\mathbb{T}_{\mathcal{V}}$ are drawn, is both finite and universal (which implies that it is known to both $\mathcal{H}$ and $\mathcal{V}$). Treating $\mathcal{H}$ (or $\mathcal{V}$) as an adversary, nothing stops it from forging its own input to the PSI and passing the entire $\mathbb{T}$ instead of just $\mathbb{T}_{\mathcal{H}}$ (or $\mathbb{T}_{\mathcal{V}}$). Therefore, the output of the PSI will be the input set of the other participant since $\mathbb{T} \cap \mathbb{T}_{\mathcal{V}} = \mathbb{T}_{\mathcal{V}}$ (and $\mathbb{T} \cap \mathbb{T}_{\mathcal{H}} = \mathbb{T}_{\mathcal{H}}$). Therefore, any solution to achieve *Privacy-Preserving TA Negotiation* where the parties' input sets are comprised of TAs drawn from a publicly known and finite set must *validate those sets* before (or while) attempting a PSI. We propose a solution to this problem using secure multiparty computing in the next section.

5.3 MPC protocol for TA Negotiation

Consider two parties, each having its own set of trust anchors (from a universal set of trust anchors known to everyone) that certify a specific claim that they have. Based on the claim about the counterparty, the MPC based TA negotiation protocol will allow computing the common set of trust anchors between them, such that for each trust anchor in the intersection, the parties have a valid signature on the claim in the input set. Here a signature denotes a verifiable credential issued by the trust anchor. No information about trust anchors outside this intersection is revealed. This is a case of **bidirectional trust anchor negotiation** where two consortium networks sync each others' certificate chains simultaneously ($\mathcal{N}_1$ from $\mathcal{N}_2$ and vice versa), both counterparties in every bilateral VC presentation flow hold VCs from their respective TAs, i.e., both parties are holders.

In this section, we propose our MPC based Privacy-Preserving Trust Anchor Negotiation (PTAN) protocol. We first provide a high level description of the workings of the protocol, followed by a formal description. For the formal treatment, we first define Secure PTAN protocol in the real-ideal paradigm (Definition 4). Finally, we formally prove its security according to this definition.

5.3.1 Protocol Overview

We propose an MPC based secure PTAN protocol for the two party scenario. As a bidirectional trust negotiation protocol, both the parties have their own set of trust anchors from which they have valid verifiable credentials (VCs) attesting to their claims. For the purpose of attestation in the VCs, we use the *ElGamal* signature scheme [164] for the trust anchor (TA) signatures. We chose to use ElGamal, since it is an extensively used signature scheme that also serves the purpose for our sample design and investigation. ElGamal signature scheme is well accepted and at the same time has a suitable structure that allowed us to adopt it into a secret sharing based MPC scheme. Specifically, the ElGamal signature scheme's verification algorithm was suitable to be adapted for the MPC protocol where neither the signature nor the public key could be revealed during the claim validation process. In this chapter, we demonstrate the feasibility of MPC based protocol as a practical mecha-

nism for our target application - trust anchor negotiation, and we leave experimenting with other signature schemes to future work.

Input: Both the parties participating in the MPC-based PTAN protocol input their sets of trust anchors as well as the VCs obtained from these trust anchors. Each element of this input set is a tuple consisting of (i) the claim of the party - such as identity and membership, (ii) the identity of the trust anchor in the form of its public key, (iii) VC, which is the signed claim from the trust anchor.

Notably, the claim of a party is a public input to the protocol, which means the counter-party is able to see it in plaintext. On the other hand, the public keys of the trust anchors (identity of the trust anchors) are secret shared private inputs to the MPC protocol, and thus remain unknown to the counter-party. The signatures of the VCs are *ElGamal* signatures which have two parts, as denoted by (r, s) (see Section 5.3.3), and are treated separately. The r values are given as public inputs to the protocol, whereas the s values are secret shared private inputs. Later in Section 5.3.5 we show that revealing the r component does not compromise the security of our protocol.

Compute intersection and claim validation: Once the claims, trust anchor public keys and signatures are provided as inputs, the MPC protocol carries out the following operations for each possible pair of trust anchor inputs from the two parties. (i) subtract the public keys of the trust anchor pair such that for an identical trust anchor the result will be zero (ii) validate the VC signature of one party such that the result is zero on successful validation (iii) similarly validate the VC signature of the other party. (iv) combine the results from the previous three steps by multiplying each of them with a random integer and then adding. This becomes the result based on which the common trust anchors are revealed in the output phase.

Output: The results of the computation from the last phase are opened to both parties. This result is zero for a pair of trust anchors only if (i) they have the same identity, implying a common trust anchor (ii) both parties have valid signatures on their claim implying a valid VC from that trust anchor. Therefore, the corresponding trust anchor is finally opened to both parties which reveals the identity of the common trust anchor.

5.3.2 Definition of PTAN in Real-Ideal Paradigm

In order to formally prove the security of our protocol we define the bidirectional TA negotiation problem using the real-ideal paradigm, which is standard in the cryptography literature.

Ideal World: We first define an ideal protocol using a trusted third party. In the ideal protocol, two participating parties have to submit their disposal to a trusted third party τ that will perform the bidirectional *PTAN*. The parties P_1 and P_2 inform τ their input sets; then τ privately computes the intersection after validating the issuers' signatures and finally reveals the output to both the parties. Let P_1 and P_2 obtain their input sets $\mathbb{S}_{P_1}$ and $\mathbb{S}_{P_2}$, respectively. The input sets contain two tuples – the public key and the signature of the issuing TA over a claim known to both the parties. A TA is identified using its public key, and $\mathcal{ID}_{P_1}$ and $\mathcal{ID}_{P_2}$ are the sets of public keys of the TAs of P_1 and P_2 respectively. Input set $\mathbb{S}_{P_1} = \{(y, \sigma_{y,\mathsf{m}_{P_1}}) \mid y \in \mathcal{ID}_{P_1}\}$. Here $\sigma_{y,\mathsf{m}_{P_1}}$ denotes claim m_{P_1} about the party P_1, signed by issuer y's private key. P_2's input set $\mathbb{S}_{P_2}$ is defined similarly.

Initialization: Party P_1 sends a message (`start`, P_1, P_2) to τ, if it wants to start the protocol with another party P_2. If party P_2 sends the message (`start`, P_2, P_1), then τ responds with `ok` to P_1 and P_2, then the protocol starts. Otherwise, τ sends `abort` to P_1 and P_2, and the protocol is aborted there.

Inputs of P_1: (i) P_1 sends $\mathbb{S}_{P_1}$ to τ; (ii) τ sends $|\mathbb{S}_{P_1}|$ to P_2; (iii) P_2 responds with either `abort` or `ok`.

Inputs of P_2: (i) P_2 sends $\mathbb{S}_{P_2}$ to τ; (ii) τ sends $|\mathbb{S}_{P_2}|$ to P_1; (iii) P_1 responds with either `abort` or `ok`.

Computation: τ validates the signatures $\sigma_{y,m}$ against the public key y and the claim m in the input sets $\mathbb{S}_{P_1}$ & $\mathbb{S}_{P_2}$ and computes the validated sets of public keys of TAs $\mathsf{V}(\mathbb{S}_{P_1})$ and $\mathsf{V}(\mathbb{S}_{P_2})$, respectively. Here $\mathsf{V}(\mathbb{S}_{P_1}) = \{y \mid (y, \sigma_{y,\mathsf{m}_{P_1}}) \in \mathbb{S}_{P_1} \wedge \sigma_{y,\mathsf{m}_{P_1}}\ is\ valid\}$, and similarly $\mathsf{V}(\mathbb{S}_{P_2})$. τ then computes the intersection of validated sets of public keys: $\mathbb{O}^* = \mathsf{V}(\mathbb{S}_{P_1}) \cap \mathsf{V}(\mathbb{S}_{P_2})$.

τ sends $\mathbb{O}^*_{P_1} = \mathbb{O}^*$ to P_1 and $\mathbb{O}^*_{P_2} = \mathbb{O}^*$ to P_2.

Let this ideal world functionality be denoted by $\mathcal{F}$. Therefore, $(\mathbb{O}^*_{P_1}, \mathbb{O}^*_{P_2}) \leftarrow \mathcal{F}(\mathbb{S}_{P_1}, \mathbb{S}_{P_2})$.

Real World: Parties P_1 and P_2 are expected to execute the specified bidirectional *PTAN* protocol (in the absence of any trusted third party). The parties may be malicious i.e., deviate from the protocol and follow arbitrary polynomial time strategies to achieve more than what is allowed by the protocol. For instance, a *PTAN* protocol attempts to prevent a malicious party to: (i) learn about the other party's TAs that are not present in the intersection, (ii) introduce a TA in the output intersection without possessing a claim that it has attested, (iii) prevent the other party to learn about the output after it itself learns about it (fairness property).

Definition 4 (**Secure PTAN Protocol**). For a security parameter λ, a protocol π is a Secure PTAN protocol in the presence of malicious adversaries if for every real-world adversary $\mathcal{A}$ that runs in time polynomial in λ, there exist a simulator $\mathcal{S}$ that runs in time polynomial in λ, such that for all inputs to the honest party B, the following distributions are indistinguishable, except with a probability negligible in λ.

$$\mathsf{Real}_{\pi,\mathcal{A}}(\lambda; \{\mathbb{S}_{P_2}\}) \approx_{\text{comp}} \mathsf{Ideal}_{\mathcal{F},\mathcal{S}}(\lambda; \{\mathbb{S}_{P_2}\}) \tag{5.1}$$

with comp referring to the possibility that additional computational assumptions can be involved.

$\mathsf{Real}_{\pi,\mathcal{A}}(\lambda; \{\mathbb{S}_{P_2}\})$ is running the protocol with the honest party P_2 using its private input $\mathbb{S}_{P_2}$, and the messages of the corrupt party P_1 chosen according to $\mathcal{A}$. The output is $(V_{\mathcal{A}}, \mathbb{O}_{P_2})$, where $V_{\mathcal{A}}$ denotes the view[1] of the adversary $\mathcal{A}$ and $\mathbb{O}_{P_2}$ is the output of P_2.

$\mathsf{Ideal}_{\mathcal{F},\mathcal{S}}(\lambda; \{\mathbb{S}_{P_2}\})$ denotes running a stateful simulator $\mathcal{S}$ (ideal world adversary) to find a set of inputs $\mathbb{S}_{\mathcal{A}}$ for adversary $\mathcal{A}$, then using the ideal functionality $\mathcal{F}$ to compute $(\mathbb{O}^*_{\mathcal{A}}, \mathbb{O}^*_{P_2}) \leftarrow \mathcal{F}(\mathbb{S}_{P_1}, \mathbb{S}_{P_2})$. Then $\mathbb{O}^*_{\mathcal{A}}$ is given as input to $\mathcal{S}$, which outputs V^*. The output of $\mathsf{Ideal}_{\mathcal{F},\mathcal{S}}$ is $(V^*, \mathbb{O}^*_{P_2})$.

[1] View of a party in a protocol refers to all the messages that the party has access to during its participation in the protocol, which also includes its secret inputs and other internal randomness.

5.3.3 Preliminaries

ElGamal Signature Scheme

The essential procedures for ElGamal signature scheme are as follows:

Key generation: Let N be the key length. An N bit prime number p is chosen, and a cryptographic hash function $H(.)$ is chosen with output length L. Let $L < N$. Choose a generator g of the multiplicative group Z_p^*. A private key x is chosen randomly from $\{1 \dots p-2\}$. The public key is computed as $y := g^x \mod p$.

Signing: For signing a message m, choose an integer γ randomly from $\{2 \dots p-1\}$, such that γ is relatively prime to $p-1$. Compute $r := g^\gamma \mod p$ and $s := (H(\mathsf{m}) - xr)\gamma^{-1} \mod (p-1)$. If $s = 0$, start with different γ. The signature is denoted by (r, s).

Verification: A signature (r, s) can be verified with the help of the public key y as follows: Signature is valid if and only if $g^{H(\mathsf{m})} \equiv y^r r^s \mod p$.

Secure Multiparty Computation (MPC)

MPC [162] addresses a problem involving two or more parties that wish to jointly compute a given function result where each party provides an input privately. Only the function result is revealed, while nothing more about the private inputs than the function result is disclosed to other parties. Through the MPC protocol, the parties compute the function by communicating among themselves without unduly giving any information about the private inputs. Over the years, many MPC protocols have been introduced in the literature, such as based on garbled circuits [165], and secret sharing [166].

Notations

For each TA of party P_1, we denote the corresponding ElGamal private key and public key pair as α_i and β_i respectively. Similarly, the private key and public key pair of each TA

of party P_2 is denoted by μ_j and ω_j respectively. The TAs are identified by their public keys, and hence the TA set of P_1 and P_2 are $\mathcal{ID}_{P_1} = \{\beta_1, .., \beta_a\}$ and $\mathcal{ID}_{P_2} = \{\omega_1, .., \omega_b\}$ respectively, where a and b are the number of TAs of P_1 and P_2 respectively.

Let the signature issued by β_i to P_1 on the claim m_{P_1} be represented as $\sigma_{y,\mathsf{m}_{P_1 i}} = (r_{P_1 i}, s_{P_1 i})$. Party P_1's input set to the protocol is thus represented as $\mathbb{S}_{P_1} = \{(\beta_i, (r_{P_1 i}, s_{P_1 i})) \mid \beta_i \in C_{P_1}\}$. Let the signatures over the claim m_{P_2} issued to party P_2 by its TAs $\omega_j \in \mathcal{ID}_{P_2}$ be $(r_{P_2 j}, s_{P_2 j})$. Party P_2's input set to the protocol is therefore $\mathbb{S}_{P_2} = \{(\omega_j, (r_{P_2 j}, s_{P_2 j})) \mid \omega_i \in \mathcal{ID}_{P_2}\}$.

For binary representation of a data d, the n_{th} bit of d starting from least significant bit is denoted as $d[n]$.

5.3.4 Formal Description of the Protocol

Algorithm 3 depicts the steps of our proposed MPC based PTAN protocol. At the beginning of the protocol, through steps (1 to 6), the parties P_1 and P_2 exchange the r values of the signatures (r, s) in their input sets as $\mathcal{R}_{P_1} = \{r_{P_1 i} \mid (\beta_i, (r_{P_1 i}, s_{P_1 i})) \in \mathbb{S}_{P_1}\}$ and $\mathcal{R}_{P_2} = \{r_{P_2 j} \mid (\omega_j, (r_{P_2 j}, s_{P_2 j})) \in \mathbb{S}_{P_2}\}$ respectively. Therefore, this reveals the size of the input sets $a = |\mathbb{S}_{P_1}|$ and $b = |\mathbb{S}_{P_2}|$ to both the parties, and we would argue in Theorem 2 that this does not reveal information about the respective public keys. To aid in fast exponentiation (Algorithm 4) inside the MPC protocol, $\hat{\mathcal{R}_{P_1}}$ and $\hat{\mathcal{R}_{P_2}}$ are calculated in steps (2) and (5) respectively and provided as (public) inputs. Each party also samples a random integer, $rand_{P_1}, rand_{P_2}$ in step (8). Finally the MPC protocol (steps 11 to 24) receives the following public inputs: $\left\{\mathcal{R}_{P_1}, \hat{\mathcal{R}_{P_1}}, \mathcal{R}_{P_2}, \hat{\mathcal{R}_{P_2}}, g^{\mathcal{H}(\mathsf{m}_{P_2})}, g^{\mathcal{H}(\mathsf{m}_{P_1})}\right\}$ *ElGamal* parameters N, g and p are also available as public inputs to the MPC.

Party P_1 gives the following as private input to the MPC protocol in step (9) $rand_{P_1}$, $\{\beta_i^{r_{P_1 i}} \bmod p, s_{P_1 i} \mid (\beta_i, (r_{P_1 i}, s_{P_1 i})) \in \mathbb{S}_{P_1}\}$, and $\{\beta_i^{r_{P_2 j}} \mod p \mid r_{P_2 j}, \beta_i \in \mathcal{R}_{P_2} \times \mathcal{ID}_{P_1}\}$. Similarly the private input by P_2 to the MPC is $rand_{P_2}$, $\{\omega_j^{r_{P_2 j}} \mod p, s_{P_2 j} \mid (\omega_j, (r_{P_2 j}, s_{P_2 j})) \in \mathbb{S}_{P_2}\}$, and $\{\omega_j^{r_{P_1 i}} \mod p \mid r_{P_1 j}, \omega_j \in \mathcal{R}_{P_1} \times \mathcal{ID}_{P_2}\}$ (step (10)).

The MPC protocol first computes a random integer $rand$ combining $rand_{P_1}$ and $rand_{P_2}$

Algorithm 3: MPC based PTAN protocol

P_1 sends the set of r_{P_1} values of signatures: $\mathcal{R}_{P_1} = \{r_{P_1 i} \mid (\beta_i, (r_{P_1 i}, s_{P_1 i})) \in \mathbb{S}_{P_1}\}$, to P_2 as clear input.

$\hat{\mathcal{R}_{P_1}} = \left\{r_{\hat{P}_1 i} \mid r_{P_1 i} \in \mathcal{R}_{P_1}\right\}$
is also computed as public input from $\mathcal{R}_{P_1}$ where $r_{\hat{P}_1 i} = \{r_{P_1 i}, (r_{P_1 i})^2, (r_{P_1 i})^4, .., (r_{P_1 i})^{2^{(N-1)}} \mod p\}$.

P_2 ensures each $r_{P_1 i} \not\equiv 0 \mod p$, and aborts otherwise.

P_2 sends the set of r_{P_2} values of signatures: $\mathcal{R}_{P_2} = \{r_{P_2 j} \mid (\omega_j, (r_{P_2 j}, s_{P_2 j})) \in \mathbb{S}_{P_2}\}$, to P_1 as clear input.

$\hat{\mathcal{R}_{P_2}} = \left\{r_{\hat{P}_2 j} \mid r_{P_2 j} \in \mathcal{R}_{P_2}\right\}$
is also computed as public input from $\mathcal{R}_{P_2}$ where $r_{\hat{P}_2 j} = \{r_{P_2 j}, (r_{P_2 j})^2, (r_{P_2 j})^4, .., (r_{P_2 j})^{2^{(N-1)}} \mod p\}$.

P_1 ensures each $r_{P_2 i} \not\equiv 0 \mod p$, and aborts otherwise.

P_1 and P_2 also input in clear $g^{H(\mathsf{m}_{P_2})}$ and $g^{H(\mathsf{m}_{P_1})}$ respectively corresponding
to the claims m_{P_2} and m_{P_1} to the MPC, based on which they want the counterparty's inputs to be validated.

P_1 and P_2 respectively sample random integers $rand_{P_1}, rand_{P_2} \leftarrow Z_p$ in private.

P_1 gives the following as private input to the MPC protocol:
(i) $rand_{P_1}$
(ii) $\{\beta_i^{r_{P_1 i}} \mod p, s_{P_1 i} \mid (\beta_i, (r_{P_1 i}, s_{P_1 i})) \in \mathbb{S}_{P_1}\}$
$s_{P_1 i}$ is given input in binary representation.
(iii) $\{\beta_i^{r_{P_2 j}} \mod p \mid r_{P_2 j}, \beta_i \in \mathcal{R}_{P_2} \times \mathcal{ID}_{P_1}\}$

P_2 gives the following as private input to the MPC protocol:
(i) $rand_{P_2}$
(ii) $\{\omega_j^{r_{P_2 j}} \mod p, s_{P_2 j} \mid (\omega_j, (r_{P_2 j}, s_{P_2 j})) \in \mathbb{S}_{P_2}\}$
$s_{P_2 j}$ is given input in binary representation.
(iii) $\{\omega_j^{r_{P_1 i}} \mod p \mid r_{P_1 j}, \omega_j \in \mathcal{R}_{P_1} \times \mathcal{ID}_{P_2}\}$

MPC:

Initialize $rand \leftarrow rand_{P_1} + rand_{P_2} \mod p$

for $i \leftarrow 1 \ldots a$ **do**

  Initialize secret integers $c_1, c_2, c_3, c_4 \leftarrow 1$

  ▷ `Validate` P_1`'s input signature`

  Compute $\eta_{P_1} = r_{P_1 i}^{s_{P_1 i}} \mod p = \mathsf{FExp}(r_{\hat{P}_1 i}, s_{P_1 i})$ (Algorithm 4)

  Compute $\theta_{P_1} = \eta_{P_1} \beta_i^{r_{P_1 i}} \mod p$

  $c_1 \leftarrow \theta_{P_1} - g^{\mathcal{H}(\mathsf{m}_{P_1})}$

  for $j \leftarrow 1 \ldots b$ **do**

    ▷ `Match TA`

    $c_2 \leftarrow (\beta_i^{r_{P_1 i}} - \omega_j^{r_{P_1 i}})$

    ▷ `Validate` P_2`'s input signature`

    Compute $\eta_{P_2} = r_{P_2 j}^{s_{P_2 j}} \mod p = \mathsf{FExp}(r_{\hat{P}_2 j}, s_{P_2 j})$

    Compute $\theta_{P_2} = \eta_{P_2} \omega_j^{r_{P_2 j}} \mod p$

    $c_3 \leftarrow \theta_{P_2} - g^{\mathcal{H}(\mathsf{m}_{P_2})}$

    $c_4 \leftarrow c_4 \cdot (c_2 + rand \cdot c_3)$

  if $c_1 + rand \cdot c_4 == 0$ **then**

    Output $(\beta_i^{r_{P_1 i}}, r_{P_1 i})$

P_1 and P_2 collect the outputs from MPC in a set $\mathbb{Q}$.

P_1 computes $\mathbb{O}_{P_1}$ as $\{\beta_i = (\beta_i^{r_{P_1 i}})^{(r_{P_1 i}{}^{-1})} \mod p | (\beta_i^{r_{P_1 i}}, r_{P_1 i}) \in \mathbb{Q}\}$

P_2 computes $\mathbb{O}_{P_2}$ as $\{\beta_i = (\beta_i^{r_{P_1 i}})^{(r_{P_1 i}{}^{-1})} \mod p | (\beta_i^{r_{P_1 i}}, r_{P_1 i}) \in \mathbb{Q}\}$

Algorithm 4: Fast Exponentiation: $\mathsf{FExp}(\hat{r}, s)$

1 **Input:** $\hat{r} = r, r^2, r^4, .., r^{2^{(N-1)}} \mod p$,
s in N bit binary representation.

$$
\begin{aligned}
s &= \sum_{b=0}^{N-1} s[b] \cdot 2^b \\
r^s \mod p &= r^{\sum_{b=0}^{N-1} s[b] \cdot 2^b} \mod p \\
&= \prod_{b=0}^{N-1} r^{s[b] \cdot 2^b} \mod p \\
&= \prod_{b=0}^{N-1} \left(s[b] \cdot r^{2^b} + (1 - s[b]) \right) \mod p
\end{aligned}
$$

2 **Return** $r^s \mod p$

in step (11). Then it iterates through each input from P_1 through step (12), where it first validates signature $(r_{P_1 i}, s_{P_1 i})$ from the TA β_i over claim m_{P_1} through steps (14) to (16). For each $\beta_i^{r_{P_1 i}}$, each input of P_2 is iterated over and $\beta_i^{r_{P_1 i}}$ is compared to $\omega_j^{r_{P_1 i}}$ for $j = 1 \ldots b$ (step (18), while also validating the signature $(r_{P_2 j}, s_{P_2 j})$ from ω_j on m_{P_2} (steps (19) to (22)). Step (23) ensures that for a particular $\beta_i^{r_{P_1 i}}$, the pair $(\beta_i^{r_{P_1 i}}, r_{P_1 i})$ is outputted by the MPC only if the signature from β_i is valid, as well as there is some ω_j for which the signature from ω_j is valid, and $\beta_i^{r_{P_1 i}} == \omega_j^{r_{P_1 i}}$. P_1 and P_2 collect the outputs from MPC in a set $\mathbb{Q}$ in step (25). From $\mathbb{Q}$, the two parties P_1 and P_2 compute $\mathbb{O}_{P_1}$ and $\mathbb{O}_{P_2}$ respectively as $\{\beta_i = (\beta_i^{r_{P_1 i}})^{({r_{P_1 i}}^{-1})} \mod p \mid (\beta_i^{r_{P_1 i}}, r_{P_1 i}) \in \mathbb{Q}\}$. β_i is polynomial time computable given ${\beta_i}^{r_{P_1 i}}$ and $r_{P_1 i}$, by computing ${r_{P_1 i}}^{-1} \mod p - 1$ in Z_p^*.

• *Correctness*: When both parties P_1 and P_2 are honest, for each common TA $\beta_i = \omega_j$, the condition $\beta_i^{r_{P_1 i}} == \omega_j^{r_{P_1 i}}$ is satisfied (step (18)). Therefore the i_{th} signature of $\mathbb{S}_{P_1}$: $(r_{P_1 i}, s_{P_1 i})$ and the j_{th} signature of $\mathbb{S}_{P_2}$: $(r_{P_2 j}, s_{P_2 j})$ are validated within the MPC protocol in steps (14) to (16) and steps (19) to (21) respectively. Then the output set $\mathbb{O} = \mathsf{V}(\mathbb{S}_{P_1}) \cap \mathsf{V}(\mathbb{S}_{P_2})$ is revealed to both the parties which is the correct output as per the ideal world definition of bidirectional PTAN.

• *Complexity*: The MPC based PTAN protocol has two loops, an outer loop iterating over inputs of P_1 in step (13) a times, and an inner loop in step (17) iterating over inputs of P_2, b times. The complexity of the protocol is thus $O(a \cdot b \cdot N)$ multiplications and $O(a)$ equality checks. In a secure MPC protocol for arithmetic circuits [37, 166], the complexity of $O(a)$

equality checks dominate the execution time. In order to optimize the execution time based on the size of the input sets of the two parties, the MPC is initialized to iterate over the smaller input set in the outer loop which admits $O(\mathsf{min}(a, b))$ secure equality checks. We validate this through experiments in §5.4.

5.3.5 Security Analysis

In this section, we analyze the proposed MPC based PTAN protocol (Algorithm 3), and show that it satisfies Definition 4.

First, we show that despite the fact that $r_{P_1 i}$ and $r_{P_2 j}$ are shared in the clear with the counterparties in steps (1) and (4) of Algorithm 3, no information about the input set of TA' public keys is revealed from them.

Theorem 2. Given only r of an *ElGamal* signature (r, s), the public key of the signer y cannot determined.

Proof. Apriori probability of determining the public key of the signer in ElGamal signature scheme (see §5.3.3), without any other information, is given by $\mathcal{P}(y) = O(\frac{1}{2^N})$, where N is key length in bits, since the private key is sampled randomly from $\{1, \ldots, p-2\}$.

When signing a message, the signer samples an integer γ randomly from $\{2, \ldots, p-1\}$ and relatively prime to $p-1$. There are $\varphi(p-1) = \frac{(p-1)}{2} - 1$ group elements relatively prime to $p-1$ when p is a safe prime. And, more importantly, this sampling is done independent of y. Signer computes $r := g^{\gamma} \mod p$, and hence $\mathcal{P}(r|y) = P(r)$.

Therefore, the a posteriori probability of determining y, given an r, is given by $\mathcal{P}(y|r) = \frac{\mathcal{P}(y)\mathcal{P}(r|y)}{\mathcal{P}(r)} = \mathcal{P}(y)$. □

Corollary 1 (To Theorem 2)**.** When provided a set of public keys $\mathcal{ID}$ and an r of a valid signature (r, s), such that $\mathcal{ID}$ may or may not contain the public key y of the signer of (r, s), it cannot be determined whether $y \in \mathcal{ID}$ with more than the guessing probability $1/2$.

Next we show that Algorithm 3 is a secure PTAN protocol in presence of malicious adversaries.

Theorem 3. When constructed with an MPC protocol which is secure against malicious adversaries Algorithm 3 specifies a bidirectional PTAN protocol (Definition 4).

Proof. Without the loss of generality, let us consider an honest party P_2 and an adversary $\mathcal{A}$ participating in the protocol. We construct a simulator $\mathcal{S}$ for the adversary. $\mathcal{S}$ is given an oracle access to the real-world adversary $\mathcal{A}$.

$\mathcal{S}$ uses the simulator $\mathcal{S}^{\text{MPC}}$ of the maliciously-secure MPC protocol π^{MPC} that is used to construct the PTAN protocol. $\mathcal{S}$ plays the role of $\mathsf{Ideal}_{\mathcal{F}^{\text{MPC}},\mathcal{S}^{\text{MPC}}}$ for $\mathcal{S}^{\text{MPC}}$ and makes $\mathcal{S}^{\text{MPC}}$ select inputs $\mathcal{I}^* = \{r^*_{\mathcal{A}i}, {\beta^*_i}^{r^*_{\mathcal{A}i}}, s^*_{\mathcal{A}i}, {\beta^*_i}^{r^*_{P_2 j}}, rand_{\mathcal{A}}\}$, to the MPC protocol for the adversary $\mathcal{A}$, for which the joint distributions of the output of $\mathsf{Ideal}_{\mathcal{F}^{\text{MPC}},\mathcal{S}^{\text{MPC}}}$ is computationally indistinguishable from the output of $\mathsf{Real}_{\pi^{\text{MPC}},\mathcal{A}}$. From this set of adversarial inputs to the MPC, the simulator $\mathcal{S}$ constructs the adversarial inputs for the MPC based PTAN protocol $\mathbb{S}^*_{\mathcal{A}} = \{(\beta^*_i, (r^*_{\mathcal{A}i}, s^*_{\mathcal{A}i}))\}$. β^*_i is polynomial time computable from $r^*_{\mathcal{A}i}, {\beta^*_i}^{r^*_{\mathcal{A}i}} \in \mathcal{I}^*$, by computing ${r^*_{\mathcal{A}i}}^{-1} \mod p-1$. Therefore $\mathcal{S}$ computes $\mathbb{S}^*_{\mathcal{A}}$ in polynomial time from $\mathcal{I}^*$.

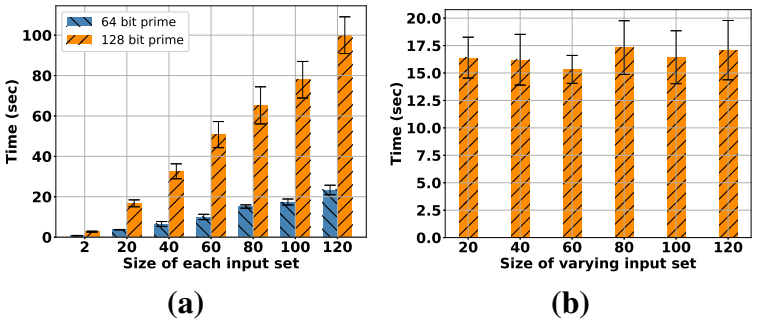
$\mathsf{Ideal}_{\mathcal{F},\mathcal{S}}$ provides $\mathbb{S}^*_{\mathcal{A}}$ and $\mathbb{S}_{P_2}$ as inputs to $\mathcal{F}$ and obtains $(\mathbb{O}^*_{\mathcal{A}}, \mathbb{O}^*_{P_2})$. $\mathcal{S}$ then obtains as input $\mathbb{O}^*_{\mathcal{A}}$ which is the output of the ideal functionality to $\mathcal{A}$. $\mathcal{S}$ starts the MPC based PTAN protocol in the ideal world by first sending (`start`, $\mathcal{A}$, P_2) to τ. Assuming P_2 does not abort the protocol, $\mathcal{S}$ sends $\mathbb{S}^*_{\mathcal{A}}$ to τ, and receives $b = |\mathbb{S}_{P_2}|$. $\mathcal{S}$ randomly chooses $|\mathbb{S}_{P_2}|$ integers from $\{2...p-1\}$, such that they are relatively prime to $p-1$ and constructs $\mathcal{R}^*_{P_2} = \{{r_{P_2 1}}^*, .., {r_{P_2 b}}^*\}$. $\mathcal{S}$ also obtains the view of the adversary during MPC execution $V^{*\text{MPC}}$ from $\mathcal{S}^{\text{MPC}}$. $\mathcal{S}$ finally outputs $V^* = \{\mathbb{S}^*_{\mathcal{A}}, \mathcal{R}^*_{P_2}, V^{*\text{MPC}}\}$. Hence,

$$\mathsf{Ideal}_{\mathcal{F},\mathcal{S}}(\lambda; \{\mathbb{S}_{P_2}\}) = \left(\{\mathbb{S}^*_{\mathcal{A}}, \mathcal{R}^*_{P_2}, V^{*\text{MPC}}\}, \mathbb{O}^*_{P_2}\right)$$

In the real world, the view of the adversary $V_{\mathcal{A}} = \{\mathbb{S}_{\mathcal{A}}, \mathcal{R}_{P_2}, V^{\text{MPC}}_{\mathcal{A}}\}$, where $V^{\text{MPC}}_{\mathcal{A}}$ is the view of the malicious party $\mathcal{A}$ during the MPC protocol. Hence,

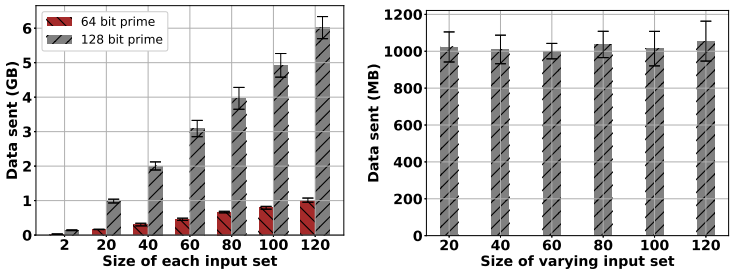
$$\mathsf{Real}_{\pi,\mathcal{A}}(\lambda; \{\mathbb{S}_{P_2}\}) = \left(\{\mathbb{S}_{\mathcal{A}}, \mathcal{R}_{P_2}, V^{\text{MPC}}_{\mathcal{A}}\}, \mathbb{O}_{P_2}\right)$$

It is evident that the distribution of the output of the honest party $\mathbb{O}^*_{P_2}$ from $\mathsf{Ideal}_{\mathcal{F},\mathcal{S}}(\lambda; \{\mathbb{S}_{P_2}\})$



(a) (b)

Figure 5.3 Execution time – (a) with varying set sizes for both parties, (b) keeping one set size constant at 20, while varying the other.



(a) (b)

Figure 5.4 Data communication overhead – (a) with varying set sizes for both parties, and (b) keeping one set size constant at 20, while varying the other.

and $\mathbb{O}_{P_2}$ from $\mathsf{Real}_{\pi,\mathcal{A}}(\lambda;\{\mathbb{S}_{P_2}\})$ are computationally indistinguishable given the security of ElGamal signature scheme that does not allow the parties to forge signatures given public key and not the corresponding secret key. Moreover, considering maliciously-secure MPC protocol, the joint distribution of the output of $\mathsf{Ideal}_{\mathcal{F}^{\text{MPC}},\mathcal{S}^{\text{MPC}}}$ is computationally indistinguishable from the output of $\mathsf{Real}_{\pi^{\text{MPC}},\mathcal{A}}$, and hence $(V^{*\text{MPC}}, \mathbb{Q}^*_{P_2}) \approx_{\text{comp}} (V^{\text{MPC}}_{\mathcal{A}}, \mathbb{Q}_{P_2})$. $V^{*\text{MPC}}$ and $V^{\text{MPC}}_{\mathcal{A}}$ includes the input to the MPC $\mathcal{I}^*$ and $\mathcal{I}$ respectively which contain $\mathcal{R}^*_{P_2}$ and $\mathcal{R}_{P_2}$ respectively and from which $\mathbb{S}^*_{\mathcal{A}}$ and $\mathbb{S}_{\mathcal{A}}$ respectively can be computed in polynomial time. From $\mathbb{Q}^*_{P_2}$ and $\mathbb{Q}_{P_2}$, $\mathbb{O}^*_{P_2}$ and $\mathbb{O}_{P_2}$ respectively can be computed deterministically in polynomial time. This implies $(\mathbb{S}^*_{\mathcal{A}}, \mathcal{R}^*_{P_2}, V^{*\text{MPC}}, \mathbb{O}^*_{P_2}) \approx_{\text{comp}} (\mathbb{S}_{\mathcal{A}}, \mathcal{R}_{P_2}, V^{\text{MPC}}_{\mathcal{A}}, \mathbb{O}_{P_2})$.

Therefore $\mathsf{Real}_{\pi,\mathcal{A}} \approx_{\text{comp}} \mathsf{Ideal}_{\mathcal{F},\mathcal{S}}$. □

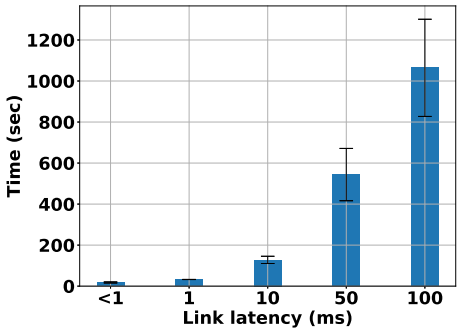


Figure 5.5 Execution time with varying link latency.

5.4 Implementation and Evaluation

We have implemented the MPC based PTAN protocol using MP-SPDZ framework [36] and made the source code available [163]. We have analyzed the performance of the PTAN protocol in terms of overall execution time as well as communication bandwidth requirement. We conducted experiments on two emulated participants running on a workstation equipped with Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz, and 128GB memory. We have used the Mascot [167] protocol, which is a maliciously-secure MPC protocol for arithmetic circuits using oblivious transfers. By varying the size of input sets of both the parties together from 2 to 120, we observe a linear increase in execution time and data communication overhead with the size of inputs in Figure5.3a and Figure5.4a respectively. The experiment is repeated for 64-bit and 128-bit primes. With 64-bit primes, even for large sets of 120 trust anchors, the time taken is less than 25 seconds, and the communication overhead is $\sim$1GB. For smaller sets of 20 trust anchors, the time is less than 4 seconds, with a data communication overhead of $\sim$170MB. With 128bit primes, the execution time and communication overhead increase to $\sim$16seconds and $\sim$1 GB, respectively, for set size 20.

In case the PTAN protocol receives skewed inputs, such that the input set size of one party is small and that of the other party is large, the computation time and the data communication overhead depend on the smaller input set. Figure5.3b and Figure5.4b show the time taken and the data communicated overhead respectively (with 128-bit primes) in a scenario where one party inputs a set of size 20, and the size of the input set of the other party is varied from 20 to 120. From the two figures, we can observe that the time and the communication overhead remain consistent with the size of the smaller set at < 20 seconds

and $\sim$1GB, respectively.

In practical scenarios, any consortium network is not likely to have hundreds of trust anchors, considering that these trust anchors are well-known entities such as companies, organizations, or governments that directly attest to the identity and membership of the consortium participants. Considering privacy-preserving trust negotiation is to be carried out only when configuring identities across two consortium blockchain networks before interoperability, it is a one-off process (or repeated in considerable time intervals), the time and communication overhead are acceptable.

To study the impact of link latency between the two parties, we varied the link latency from $<$ 1ms to 100ms, and Figure6.10 shows the execution time for input set sizes of 20 and 120, using 128bit primes. We observe that the execution time increases significantly with the link latency, which is expected for MPC-based protocols.

5.5 Summary

The decentralized verification of credentials presents an interesting conundrum: how do the credential holder and verifier identify a certifier (or credential issuer) they have in common without revealing the identities of those they don't? This privacy-preserving certifier determination problem holds great importance for decentralized networks that wish to establish a basis for interoperability using decentralized identity and common TAs. We introduced an MPC based construction for privacy-preserving trust anchor negotiation, inspired by the classic PSI. Implementation and analysis shows that our proposed protocol can be viably utilized by blockchain networks' participants to identify the common TAs.

Notably, the problem of determining a common trust basis (i.e is a common trust anchor) is applicable wherever verifiable presentation flows are involved and is not limited to blockchain interoperation scenarios. Our solution of TA negotiation considers the simplest case of credentials attesting a single claim. In a generalized scenario of negotiation of trust basis, the parties might need to validate multiple claims to find out a common certifier for those claims. In the next chapter, we aim to generalize the problem of determining common

certifiers of claims between multiple parties. We further extend the use case to accommodate validation of multiple claims, and incorporate widely used signature schemes such as ECDSA, and BLS.

Chapter 6

Private Certifier Intersection

6.1 Introduction

In the traditional web (Web 2.0), users are dependent on a limited set of identity and service providers and public Certificate Authorities (CAs) [30] to initiate trusted interactions. Recent trends in decentralization towards Web 3.0 aim to remove such dependencies on centralized service providers. A prominent problem in the decentralized web revolves around identity and trust. Decentralized Identifiers (DIDs) [25] and Verifiable Credentials (VCs) [1] enable parties to own and control their identities. This implies a self-sovereign ability to create, update, and selectively share identity records. Importantly, one can prove properties (or *claims*) about themselves without relying on centralized/federated identity providers or a canonical trusted set of CAs [25, 79, 142], as long as the VC issuer (also called a trust anchor [34]) is trusted by both the prover and the verifier of a claim. In a nutshell, existing DID and VC recommendations give users the ability to control their privacy while engaging in a trusted decentralized interaction. But, there are scenarios where these recommendations cannot adequately safeguard user privacy unless we introduce new privacy-preserving mechanisms. In its most general form, the scenario we are concerned about involves two parties wishing to establish a trust basis for future interactions. Service providers in the Semantic Web have encountered such situations, and mechanisms for trust negotiation [102] were proposed to minimize privacy compromise without sacrificing

decentralization, albeit for a specific model of service provider-consumer interaction. In grid computing, service-level agreements (SLAs) [160] followed a similar template. This challenge has returned to salience in today's Web3 world, where private and independent blockchain systems have business imperatives to interoperate [15]. The interaction model common to these scenarios involves no a priori trust between the interacting parties, though they may, unbeknownst to each other, possess VCs (or more generally *certificates*) from common trust anchors (or more generally *certifiers*) attesting to different claims.

A trust basis for interoperation can be established between two parties if they can determine that they both possess valid certificates attesting to certain claims, and that these certificates are issued by one or more certifiers that they both trust. But this is hard to do in the absence of a priori trust or knowledge of the counterparty's intentions, or without compromising one's privacy. We can see why this is so by applying the standard VC recommendation, whereby one party makes a Verifiable Presentation (VP) [1] to another, to our scenario. In a typical VC use case, the relationship between credential presenter and verifier is asymmetric, as the verifier is typically a well-known entity from whom the presenter seeks service or approval. The presenter knows at least one certifier that is trusted by it and the verifier. Typically, this requires the verifier to publish its complete list of certifiers so the presenter can determine ones that are commonly trusted by both parties [114]. But in our interaction model, the relationship between parties is symmetrical, as they are both trying to simultaneously prove something to the other. In a standard VP, the presenter is willing to share credentials (albeit selectively) with the verifier. But, if we use this asymmetric VP-based solution in our scenario where neither party knows anything about the other a priori, the revelation of credentials by the party that presents first will automatically give more leverage to the counterparty (verifier), which learns more about the presenter than it reveals.

A naïve adaptation of an asymmetric solution (such as [114]) to our symmetric setting would require both parties to reveal to each other the list of certifiers from which they have valid certificates, and then identify if there is a mutually trusted certifier. This entails complete loss of privacy for both parties, but especially for an honest party if the other behaves maliciously. There are strong reasons why revealing one's complete list of certifiers might not be in one's interest. A business-oriented certifier, for instance, might not like its clientele to be visible to its market competitors. Consider the blockchain interoperability scenario elaborated in the previous chapter (Chapter 5), where shipment carriers on different

trade networks certify their respective networks' participants, e.g., Maersk Shipping Company (on the TradeLens network [10]) and the American Bureau of Shipping (ABS). But as Maersk and ABS are market competitors, they may not necessarily want their clients (the certificate holders) to reveal their respective associations. Knowing the clientele of Maersk may benefit ABS, and vice versa; hence there is a privacy cost to revealing certifier lists in a symmetrical interaction unless those certifier lists are identical.

The other privacy violation aspect is from the perspective of the certificate holder. Every certificate possessed indicates an affiliation with some real world entity, often a well-known one; this could include government agencies, political organizations, NGOs, etc., and such affiliations might be sensitive information that could potentially be misused. And here lies the biggest hazard in the naïve trust basis establishment solution: one of the two interacting parties could be malicious and is trying to fish for information about its counterparty's affiliations. A simple attack would be for the malicious party to offer a long list of certifiers, regardless of whether it possesses valid certificates from them, and have the honest counterparty reveal its true certifier list. Now the malicious party knows, and can misuse, the honest party's affiliations, without revealing its own true affiliations. In the context of trust anchors (TAs) in the DID & VC world, where any entity can issue a VC and there does not exist a canonical list or registry of global TAs, it would not be a hard task for a malicious counterparty to list as many of them as possible to mount the attack we just described. Therefore, we can identify a compelling need to maintain certifier privacy and authenticity, which are not addressed by the naïve solution for determining common certifiers. This motivates us to ask the following question:

Can parties owning certificates efficiently identify
a common set of certifiers without leaking anything else?

In particular, the parties should not learn any information about certifiers that may be in the lists of other parties but are not in the intersection.

In this chapter, we initiate the study of **Private Certifier Intersection** (PCI) – a cryptographic primitive that aims to answer the above question in the affirmative. Informally speaking, a PCI protocol allows a set of mutually distrusting certificate-holding parties to

achieve a privacy-preserving trust negotiation with the following objectives: (i) find an intersection among the set of certifiers across the parties, (ii) ensure that the certificates issued by these certifiers are valid, and (iii) reveal no information about the certifiers that may be in the lists of individual parties but are not in the intersection.

Comparison with Private Set Intersection. At a first glance, the classic *Private Set Intersection* (PSI) problem [103, 105], where the intersection of two private sets must be determined without a trusted mediator, bears a strong resemblance to PCI (also see Figure 6.1). In both PCI and PSI, a set of mutually distrusting parties holding private sets of entities aim to compute the intersection between their sets without revealing any additional information about the elements in their individual sets that are not in the intersection. However, the non-triviality of PCI arises from the need to additionally validate the certificates issued by the certifiers in the intersection. In this sense, one can think of PCI as a form of "predicated" PSI, where the inclusion of a common certifier in the final output set is predicated on the certificates issued by this certifier to each of the parties being valid (see Figure 6.2 for an illustration). We argue in this chapter that realizing an efficient PCI protocol with ideal security guarantees requires novel techniques beyond simply using PSI as a building block. Consider the hazard we encountered earlier in the naïve solution to establish a trust basis. Using standard PSI, a malicious party could simply supply a long (or universal) list of certifiers as input and determine the list of certifiers of the other (honest) party. To avoid this hazard, we need to enforce the ability of participants to prove that they possess genuine certificates issued by thir claimed certifiers. There is no obvious way to do this using standard PSI, and therefore PCI requires novel mechanisms that are not congruent to PSI's mechanisms.

Achieving Semi-Honest PCI. It turns out that in the setting of semi-honest corruptions (i.e., when the participating parties behave honestly as prescribed in the protocol), one can easily achieve a secure PCI protocol by using any secure PSI protocol in a black-box way. Consider the following simple construction: each party first locally "filters" its private list of certifiers based on the validity of the certificates issued by such certifiers, and then uses this filtered list of certifiers as its input to an execution of a PSI protocol to securely identify their intersection. Correctness is immediate, since, assuming honest behavior, the filtered list for each party only contains certifiers issuing valid certificates. Security follows from the security of the underlying PSI protocol.

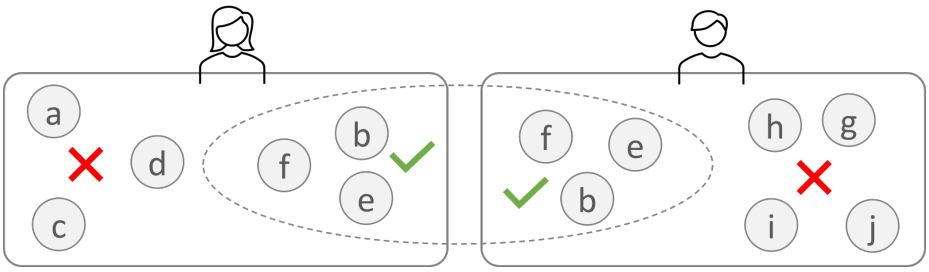


Figure 6.1 Private Set Intersection (PSI): Match Values

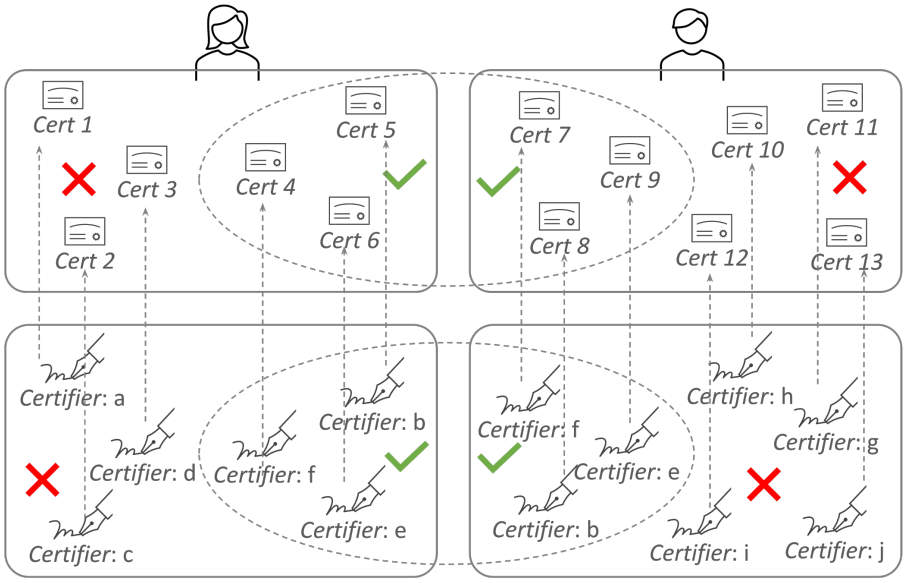


Figure 6.2 Private Certifier Intersection (PCI): Match Certificates with Common Issuers

Upgrading to Malicious Security. Unfortunately, in the setting of malicious corruptions (i.e., when the participating parties can deviate arbitrarily from the protocol), it is seemingly hard to achieve a secure PCI protocol by simply using certification validation and a (maliciously secure) PSI as individual black-boxes. To begin with, we cannot rely on the parties to filter the local sets of certifiers correctly; in fact, the parties can prepare arbitrary sets of certifiers, including those for which it does not have valid certificates.

For example, in the setting of two-party PCI, if one party (say Alice) provides a "universal set" of certifiers as input to a PSI protocol, it can learn the complete set of certifiers of the other party (say Bob). This attack may not be feasible in a general PSI setting where listing the entire range of values in an input set may be infeasible or prohibitively expensive, but is quite feasible in a PCI setting where the range of certifiers (trusted authorities) is

limited. Therefore, it is crucial for both Alice and Bob to verify that the other is not faking its input set, and so the validity of certificates and the signatures within must be proven by both parties during the protocol. This is challenging because neither Alice nor Bob knows a priori which set of certifiers it needs to supply proof for (indeed, this is the objective of PCI), and providing more proof than strictly required (i.e., revealing certifiers outside the intersection) would violate privacy goals. Therefore, we must somehow intertwine certificate validation with a PSI-like protocol to achieve PCI. In other words, a maliciously secure PCI protocol cannot be achieved securely without a mechanism that somehow intertwines certificate validation with the subsequent PSI protocol.

Theoretically, a maliciously secure PCI protocol can be achieved as follows: run a maliciously secure multi-party computation (MPC) protocol for the functionality that: (i) filters the certifier list for each party to identify the certifiers issuing valid certificates attesting to the relevant claims, and (ii) computes the intersection between these filtered sets. This solution is highly inefficient in practice for essentially all widely used cryptographically secure certification mechanisms. For example, the most common method of generating certificates is to sign the claim using a digital signature algorithm. In this case, claim validation would require us to perform signature verifications inside the MPC protocol, which is prohibitively expensive for popular digital signature schemes such as ECDSA [168, 169] and BLS [131, 170, 171], that rely on elliptic curve-based finite-field arithmetic operations. Implementing such a verification algorithm inside a maliciously secure MPC protocol would involve non-black-box usage of the various elliptic-curve (EC) operations, i.e., we would have to express these operations as (potentially complicated) binary/arithmetic circuits with gate operations over $\{0, 1\}$ or over some finite field F_p. Such a maliciously secure MPC protocol is likely to incur huge computational and communication overheads in practice.

Need for Efficient Protocols. The above discussion motivates specialized PCI protocols that efficiently enable computing the intersection of certifier-sets while: (i) achieving the desired security guarantees in the setting where a majority of the parties could be maliciously corrupt, and (ii) minimizing non-black-box usage of the operations in the certificate validation algorithm. In this chapter, we design and implement two concrete PCI protocols – based on the ECDSA signature scheme and the BLS signature scheme – that achieve the above goal while supporting different variations of claim validation (we expand on this later). While our protocols broadly follow the generic approach outlined above, the

main novelty lies in how we validate signatures while using the underlying elliptic curve-based operations in a black-box manner. For an (informal) comparison, the generic MPC-based solution is expected to incur $O(xd)$ computation/communication cost, where x is the corresponding cost of our protocols, and d is the average depth of the arithmetic circuits representing EC operations (e.g., $d = 256$ for constant-time scalar multiplication over curve-ED25519 and curve-BLS12-381).

6.1.1 Overview of Contributions

In this section, we provide an informal overview of the key technical contributions of this chapter.

Defining PCI. We formalize the security guarantees expected of a (multi-party) PCI protocol using the simplified universal composability (SUC) framework due to Canetti, Cohen, and Lindell [172] in the real/ideal world paradigm. We consider two variations of PCI protocols in this chapter:

- Validate-Any PCI: A PCI-Any protocol outputs the set of common certifiers for which each party has at least one valid certificate attesting to *any* one of its (publicly known) claims.
- Validate-All PCI: A PCI-All protocol outputs the set of common certifiers for which each party has valid certificates attesting to *all* of its (publicly known) claims.

We also consider a variant of validate-any PCI which we call validate-any PCI with disclosed claims (abbreviated as PCI-Any-DC) where, for each common certifier in the output set, the parties additionally learn the set of claims attested by the certifier. We refer to Section 6.2 for a formal description.

MPC for Elliptic Curve Pairings. As a fundamental building block of our proposed PCI protocols, we introduce a new secret-sharing based MPC framework that is tuned for elliptic curve pairings. Our overall approach is to design a secret-sharing based MPC protocol that efficiently supports basic elliptic curve operations (i.e., point addition and scalar multiplication) and elliptic curve bilinear pairing operations as fundamental building blocks.

We build upon the SPDZ secret-sharing based MPC protocol [37, 173] to achieve the first secret-sharing based MPC framework that seamlessly supports elliptic curve pairing operations as fundamental gate-level building-blocks with malicious security against a dishonest majority of adversarial parties. A technical cornerstone of our framework is the round-preserving upgradation of SPDZ from basic field operations to the significantly more complicated elliptic curve operations, including pairings. Our framework allows us to directly use standardized and open-source implementations of elliptic curve libraries [174–176], thereby leveraging both the performance improvements/optimizations as well as the protections against evolving implementation-level attacks that such libraries usually offer. We believe that this is a contribution of independent interest.

Efficient Two-Party PCI. We use our proposed MPC framework to design the following provably secure yet practically efficient two-party PCI protocols:

- A two-party PCI-Any-DC protocol using the ECDSA signature scheme [168] – an elliptic-curve-based digital signature scheme which is standardized and widely adopted in multiple real-world applications including X.509 public key infrastructure in the Internet, TLS [177], DNSSEC [178], etc. Moreover, ECDSA is a candidate signature scheme in verifiable credentials [1] which is one of the target applications of PCI. Choosing ECDSA also allows us to use its standard implementation in the OpenSSL [174] library for EC group operations. This naturally motivates designing a PCI protocol supporting ECDSA-based certification of claims.
- A two-party PCI-All protocol using the BLS signature scheme [131, 170, 171]– an elliptic-curve pairing-based digital signature that is popularly used in blockchain applications and is in the process of being standardized [179]. We design a PCI-All protocol supporting BLS-based certification of claims that exploits the signature-aggregation capabilities of BLS to perform efficient validation of certificates over all of the public claims of each party.

The starting point of our protocols is the generic maliciously secure PCI protocol outlined earlier, with several optimizations to obviate or minimize expensive elliptic curve operations inside the MPC protocol. In our ECDSA-based PCI-Any-DC protocol, we develop

techniques that enable securely yet efficiently performing the expensive algebraic operations (such as field inversion) and non-algebraic operations (such finding the x-coordinate of an elliptic curve point) required by the ECDSA verification algorithm *outside* the MPC protocol. The protocol is then implemented using our proposed MPC framework, which allows performing ECDSA signature validations while using all elliptic curve operations in a black-box manner. We also discuss how to upgrade this protocol to full-fledged PCI-Any where the claims are no longer disclosed publicly (see Section 6.4 for details).

Trivially extending the approach used in our ECDSA-based PCI-Any-DC protocol to design a PCI-All protocol would require iterating through all of the public claims, and validating the signatures on these claims by a specific certifier. This results in a claim validation complexity that grows with the number of claims. We overcome this challenge by designing a PCI-All protocol using BLS-based signature-aggregation that only requires a single (aggregate-)signature verification per certifier inside the MPC protocol. We introduce additional optimizations that exploit the deterministic nature of the BLS signature to further reduce the number of elliptic curve pairing operations inside MPC to just one per certifier, which is then implemented in a black-box manner using our proposed MPC framework over pairings.

Implementation and Evaluation. We extend MP-SPDZ [36] to implement our proposed secret-sharing framework supporting elliptic curve operations including bilinear pairings. For the black-box operations on elliptic curves we use OpenSSL [174] and RELIC [176] libraries. We then implement ECDSA-based PCI-Any-DC and BLS-based PCI-All protocols. We make the source code of our implementation available at `https://github.com/ghoshbishakh/pci` for independent benchmarking. We provide a detailed analysis of the performance of the individual components of our MPC framework, followed by the end-to-end performance evaluation of the protocols in realistic setups by placing parties in three geographic regions across two continents. In an intercontinental WAN setup with parties located in different continents, our PCI-Any-DC and PCI-All protocols execute in less than a minute on input sets of size 40. This demonstrates the practicality of our proposed solutions. We refer to Section 6.6 for details.

6.2 Private Certifier Intersection (PCI)

In this section, we formally define Private Certifier Intersection (PCI). We begin by introducing some notations and background material. We subsequently formalize the functionality and security guarantees that a PCI protocol should satisfy.

General Notations. We write $x \leftarrow \chi$ to represent that an element x is sampled uniformly at random from a set/distribution $\mathcal{X}$. The output x of a deterministic algorithm $\mathcal{A}$ is denoted by $x = \mathcal{A}$ and the output x' of a randomized algorithm $\mathcal{A}'$ is denoted by $x' \leftarrow \mathcal{A}'$. For $a, b \in \mathbb{N}$ such that $a, b \geq 1$, we denote by $[a, b]$ the set of integers lying between a and b (both inclusive). We refer to $\lambda \in \mathbb{N}$ as the security parameter, and denote by $\mathsf{poly}(\lambda)$ and $\mathsf{negl}(\lambda)$ any generic (unspecified) polynomial function and negligible function in λ, respectively.[1]

PCI Notations. Let $\mathcal{ID}$ be a set of identities corresponding to the certifiers. Given a claim $\mathsf{m} \in \mathcal{M}$ by a party P, a certifier with identity id can issue a certificate $\sigma \in \mathcal{C}$, such that there exists a relation $\mathbf{R}$ that satisfies the following:

$$\mathbf{R}(\mathsf{id}, \sigma, \mathsf{m}) = 1 \text{ iff } \sigma \text{ is a valid certificate by } \mathsf{id} \text{ on } \mathsf{m}$$

A natural instantiation of the certification process outlined above is a digital signature, where the certificate issuance corresponds to the signing algorithm and the relation $\mathbf{R}$ corresponds to the verification algorithm, with σ being the signature on a claim m under the signing key corresponding to id. Looking ahead, our proposed realizations of PCI protocols in this chapter will use this digital signature-based instantiation of the certification process.

We now introduce some additional notations for ease of exposition, these notations will be useful in understanding our definitions for PCI. Let S be a set of (identity, certificate, claim) tuples of the form

$$S = \{(\mathsf{id}_j, \sigma_j, \mathsf{m}_j) \in \mathcal{ID} \times \mathcal{C} \times \mathcal{M}\}_{j \in [1,n]}$$

where N is the number of tuples in the set S. We define the following projection functions

[1] Note that a function $f : \mathbb{N} \rightarrow \mathbb{N}$ is said to be negligible in λ if for every positive polynomial p, $f(\lambda) < 1/p(\lambda)$ when λ is sufficiently large.

on the set S:

$$\mathsf{id}(S) := \{\mathsf{id} : \exists \sigma, \mathsf{m} \text{ s.t. } (\mathsf{id}, \sigma, \mathsf{m}) \in S\}$$
$$\mathsf{m}(S) := \{\mathsf{m} : \exists \mathsf{id}, \sigma \text{ s.t. } (\mathsf{id}, \sigma, \mathsf{m}) \in S\}$$
$$\overline{\mathsf{m}}(S) := (\mathsf{m}_j)_{(\mathsf{id}_j, \sigma_j, \mathsf{m}_j) \in S}$$

Here, $\overline{\mathsf{m}}(S)$ is a list/multiset of the claims corresponding to each tuple in the set S.

6.2.1 Defining a PCI Protocol

We now formally define a PCI protocol in the two-party setting, which is the focus of this chapter. Our definitions naturally extend to multiple parties, as discussed subsequently.

Two-Party PCI. A two-party PCI protocol Π involves parties P_1 and P_2, where each party P_i for $i \in \{1, 2\}$ inputs a tuple of the form $\mathsf{inp}_i = \left(\mathsf{inp}_{i,1}, \mathsf{inp}_{i,2}\right)$, where:

- The *private* input $\mathsf{inp}_{i,1}$ is a set of (identity, certificate, claim) tuples of the form

  $$\mathsf{inp}_{i,1} = \{(\mathsf{id}_{i,j}, \sigma_{i,j}, \mathsf{m}_{i,j}) \in \mathcal{ID} \times \mathcal{C} \times \mathcal{M}\}_{j \in [1, N_i]}$$

  where N_i is the number of tuples in $\mathsf{inp}_{i,1}$ from party P_i.

- The *public* input $\mathsf{inp}_{i,2}$ is a set of claims of the form $\{\widehat{\mathsf{m}}_{i,j} \in \mathcal{M}\}_{j \in [1, N_i']}$, where N_i' is the number of tuples in $\mathsf{inp}_{i,2}$ from party P_i.

Note that a party P_i can produce multiple certificates from the same certifier on same or different claims. Additionally, a party P_i can also request certifications on the same claim from multiple certifiers. Hence, in the most general setting, a party's input could have multiple tuples with the common id or a common m. Also note that the public input for P_1 is known to P_2 at the start of the protocol and vice versa.[2]

Remark. A couple of remarks on the definition follow:

[2]We assume that these sets are shared between P_1 and P_2 via some apriori mechanism that is not within the purview of the PCI protocol itself.

1. One could have a variant of PCI with the claims being private. This work considers the above defined variant with the claims being public. We leave it to future work for instantiating PCI with private claims.

2. Our definition lets a (corrupt) party provide claims in the public input that are different from those in the tuple in the private input. One could also restrict the public input $\mathsf{inp}_{i,2}$ to be $\mathsf{m}(\mathsf{inp}_{i,1})$, which is the expected behaviour of the honest parties.

At the end of the protocol Π, each party P_i receives as output a set of certifiers. In this chapter, we consider different variations of (two-party) PCI protocols that produce different kinds of output sets, that we outline below:

- **Validate-Any:** In this flavor of PCI protocol, denoted by PCI-Any, both parties P_1 and P_2 receive as output the set of certifiers $\mathsf{out}_{\mathsf{PCI\text{-}Any}}$, such that an identity $\mathsf{id} \in \mathsf{out}$ if and only if both P_1 and P_2 have valid certificates on some $\mathsf{m}_1 \in \mathsf{inp}_{i,2}$ and $\mathsf{m}_2 \in \mathsf{inp}_{i,2}$, respectively, such that both the certificates are issued by id. More formally, for each $i \in \{1, 2\}$, we define the following Boolean predicate:

$$\mathbf{R}_{\mathsf{PCI\text{-}Any},\mathsf{inp}_i}(\mathsf{id}) = 1 \text{ if and only if } \exists \mathsf{m} \in \mathsf{inp}_{i,2} : \\ \exists(\mathsf{id}, \mathsf{m}, \sigma) \in \mathsf{inp}_{i,1} \text{ s.t. } \mathbf{R}(\mathsf{id}, \mathsf{m}, \sigma) = 1$$

  Then we have

$$\mathsf{out}_{\mathsf{PCI\text{-}Any}}(\mathsf{inp}_1, \mathsf{inp}_2) = \big\{\mathsf{id} \in \mathsf{id}(\mathsf{inp}_{1,1}) \cap \mathsf{id}(\mathsf{inp}_{2,1}) : \\ \mathbf{R}_{\mathsf{PCI\text{-}Any},\mathsf{inp}_1}(\mathsf{id}) = \mathbf{R}_{\mathsf{PCI\text{-}Any},\mathsf{inp}_2}(\mathsf{id}) = 1\big\}$$

- **Validate-Any with Disclosed Claims:** We also consider a weaker variant of the aforementioned validate-any PCI protocol (denoted by PCI-Any-DC), where the parties additionally learn the following: (i) the claim $\mathsf{m}_{i,j}$ corresponding to each tuple $(\mathsf{id}_{i,j}, \sigma_{i,j}, \mathsf{m}_{i,j}) \in \mathsf{inp}_{i,1}$ for each party P_i, (ii) for each id in the output set of certifiers $\mathsf{out}_{\mathsf{PCI\text{-}Any}}$, each party learns the set of (public) claims on which the other party has a valid certificate issued by id. Note that no information is revealed about any (valid/invalid) certificates that the parties might have that are issued by some $\mathsf{id}' \notin \mathsf{out}_{\mathsf{PCI\text{-}Any}}$. Formally, for each $i \in \{1, 2\}$, we define the function

$$\mathsf{m}_{\mathsf{inp}_i}(\mathsf{id}) = \big\{\mathsf{m} : \exists(\mathsf{id}, \mathsf{m}, \sigma) \in \mathsf{inp}_{i,1} \text{ s.t. } \mathbf{R}(\mathsf{id}, \mathsf{m}, \sigma) = 1\big\}$$

Then the output set $\mathsf{out}_{\mathsf{PCI\text{-}Any\text{-}DC}}$ is described formally as follows

$$\mathsf{out}_{\mathsf{PCI\text{-}Any\text{-}DC}}(\mathsf{inp}_1, \mathsf{inp}_2) = \Big(\big\{\overline{\mathsf{m}}(\mathsf{inp}_{i,1})\big\}_{i\in[1,2]}, \\ \big\{\big(\mathsf{id}, \{\mathsf{m}_{\mathsf{inp}_i}(\mathsf{id})\}_{i\in\{1,2\}}\big) : \mathsf{id} \in \mathsf{out}_{\mathsf{PCI\text{-}Any}}(\mathsf{inp}_1, \mathsf{inp}_2)\big\} \Big)$$

PCI-Any-DC is relevant in most real-world scenarios since the parties would know the claims of the counterparty that they want to validate, and vice versa. Moreover, traditional VC interactions also work on disclosed claims (see Section 6.1).

- **Validate-All:** In this flavor of PCI protocol, denoted by PCI-All, both parties P_1 and P_2 receive as output the set of certifiers $\mathsf{out}_{\mathsf{PCI\text{-}All}}$, such that for each $\mathsf{id} \in \mathsf{out}_{\mathsf{PCI\text{-}All}}$, P_1 and P_2 have valid certificates issued by id on *all* of the (public) claims in their input sets $\mathsf{inp}_{1,2}$ and $\mathsf{inp}_{2,2}$, respectively. More formally, for each $i \in \{1, 2\}$, we define the following Boolean predicate:

$$\mathbf{R}_{\mathsf{PCI\text{-}All},\mathsf{inp}_i}(\mathsf{id}) = 1 \text{ if and only if } \forall \mathsf{m} \in \mathsf{inp}_{i,2} : \\ \exists(\mathsf{id}, \mathsf{m}, \sigma) \in \mathsf{inp}_{i,1} \text{ s.t. } \mathbf{R}(\mathsf{id}, \mathsf{m}, \sigma) = 1$$

Then we have

$$\mathsf{out}_{\mathsf{PCI\text{-}All}}(\mathsf{inp}_1, \mathsf{inp}_2) = \big\{\mathsf{id} \in \mathsf{id}(\mathsf{inp}_{1,1}) \cap \mathsf{id}(\mathsf{inp}_{2,1}) : \\ \mathbf{R}_{\mathsf{PCI\text{-}All},\mathsf{inp}_1}(\mathsf{id}) = \mathbf{R}_{\mathsf{PCI\text{-}All},\mathsf{inp}_2}(\mathsf{id}) = 1\big\}$$

6.2.2 Security of PCI

We now define the security guarantees expected of a PCI protocol in the two-party setting. Informally, we require that in any PCI protocol Π, party P_1 (resp. party P_2) learns nothing about the inputs of party P_2 (resp. party P_1) except what is revealed by the output out of the protocol Π, and the sizes N_1 and N_2 of the input sets of P_1 and P_2. In the rest of this section, we formalize this security guarantee using the simplified universal composability (SUC) framework due to Canetti, Cohen, and Lindell [172] in the real/ideal world paradigm. We consider a *dishonest majority* in our definitions, wherein the adversary can corrupt one of the two participating parties. For ease of exposition, we assume without loss of generality that P_1 and P_2 are the corrupt party and the honest party, respectively.

$\underline{\mathcal{F}_{\mathsf{PCI}}(\mathsf{mode} \in \{\mathsf{Any}, \mathsf{Any\text{-}DC}, \mathsf{All}\})}$

- For $i \in \{1, 2\}$, let the input of party P_i be $\mathsf{inp}_i = (\mathsf{inp}_{i,1}, \mathsf{inp}_{i,2})$, where

$$\mathsf{inp}_{i,1} = \{(\mathsf{id}_{i,j}, \sigma_{i,j}, \mathsf{m}_{i,j}) \in \mathcal{ID} \times \mathcal{C} \times \mathcal{M}\}_{j \in [1, N_i]}$$
$$\mathsf{inp}_{i,2} = \{\widehat{\mathsf{m}}_{i,j} \in \mathcal{M}\}_{j \in [1, N_i']}$$

  The honest party P_2 provides its input directly to $\mathcal{F}_{\mathsf{PCI}}$, while the input of the corrupt party P_1 is provided to $\mathcal{F}_{\mathsf{PCI}}$ by the simulator $\mathcal{S}$.

- $\mathcal{F}_{\mathsf{PCI}}$ computes $\mathsf{out}_{\mathsf{PCI\text{-}mode}}$, where for $\mathsf{mode} \in \{\mathsf{Any}, \mathsf{Any\text{-}DC}, \mathsf{All}\}$, we have

$$\mathsf{out}_{\mathsf{PCI\text{-}Any}}(\mathsf{inp}_1, \mathsf{inp}_2) = \big\{\mathsf{id} \in \mathsf{id}(\mathsf{inp}_{1,1}) \cap \mathsf{id}(\mathsf{inp}_{2,1}) : \mathbf{R}_{\mathsf{PCI\text{-}Any}, \mathsf{inp}_1}(\mathsf{id}) = \mathbf{R}_{\mathsf{PCI\text{-}Any}, \mathsf{inp}_2}(\mathsf{id}) = 1\big\}$$
$$\mathsf{out}_{\mathsf{PCI\text{-}Any\text{-}DC}}(\mathsf{inp}_1, \mathsf{inp}_2) = \Big(\big\{\overline{\mathsf{m}}(\mathsf{inp}_{i,1})\big\}_{i \in [1,2]}, \big\{ \big(\mathsf{id}, \{\mathsf{m}_{\mathsf{inp}_i}(\mathsf{id})\}_{i \in \{1,2\}}\big) : \mathsf{id} \in \mathsf{out}_{\mathsf{PCI\text{-}Any}}(\mathsf{inp}_1, \mathsf{inp}_2)\big\}\Big)$$
$$\mathsf{out}_{\mathsf{PCI\text{-}All}}(\mathsf{inp}_1, \mathsf{inp}_2) = \big\{\mathsf{id} \in \mathsf{id}(\mathsf{inp}_{1,1}) \cap \mathsf{id}(\mathsf{inp}_{2,1}) : \mathbf{R}_{\mathsf{PCI\text{-}All}, \mathsf{inp}_1}(\mathsf{id}) = \mathbf{R}_{\mathsf{PCI\text{-}All}, \mathsf{inp}_2}(\mathsf{id}) = 1\big\}$$

- $\mathcal{F}_{\mathsf{PCI}}$ sends $\big(\mathsf{out}_{\mathsf{PCI\text{-}mode}}, N_1, \mathsf{inp}_{1,2}\big)$ to the simulator $\mathcal{S}$.
- If $\mathcal{S}$ responds with an abort, $\mathcal{F}_{\mathsf{PCI}}$ aborts.
- Otherwise, $\mathcal{F}_{\mathsf{PCI}}$ sends to P_1 and P_2 the tuple

$$\big(\mathsf{out}_{\mathsf{PCI\text{-}mode}}, N_1, N_2, \mathsf{inp}_{1,2}, \mathsf{inp}_{2,2}\big)$$

Figure 6.3 Ideal functionality $\mathcal{F}_{\mathsf{PCI}}$ in the two-party setting

Ideal Functionality for PCI. We begin by formally defining the first component of our simulation-based security definition, namely the ideal functionality $\mathcal{F}_{\mathsf{PCI}}$, as described in Figure 6.3. This functionality $\mathcal{F}_{\mathsf{PCI}}$ formally defines what each party is meant to learn at the completion of the protocol.

The Real World. In the real world, the following participants engage in the protocol Π:

- The honest party P_2 that receives its input from the environment $\mathcal{Z}$ and honestly follows the protocol Π.

- A real-world adversary $\mathcal{A}$ that controls the corrupt party P_1, and interacts with P_2 and the environment $\mathcal{Z}$.
- The environment $\mathcal{Z}$ that provides P_2 with its input, and interacts with the real-world adversary $\mathcal{A}$. The environment $\mathcal{Z}$ also receives the output of P_2, and eventually outputs a bit $b \in \{0, 1\}$.

The Ideal World. In the ideal world, the following participants interact with the ideal functionality $\mathcal{F}_{\mathsf{PCI}}$ described in Figure 6.3.

- The honest party P_2 that receives its input from the environment $\mathcal{Z}$ and directly forwards this input to $\mathcal{F}_{\mathsf{PCI}}$.
- An ideal-world simulator $\mathcal{S}$ that sends inputs to $\mathcal{F}_{\mathsf{PCI}}$ on behalf of the corrupt party P_1 and receives back the corresponding output from $\mathcal{F}_{\mathsf{PCI}}$. $\mathcal{S}$ also interacts with the environment $\mathcal{Z}$, with the aim of making this interaction indistinguishable from the interaction between the real world $\mathcal{A}$ and the environment $\mathcal{Z}$.
- The environment $\mathcal{Z}$ that provides P_2 with its input, and interacts with the simulator $\mathcal{S}$. As in the real world, $\mathcal{Z}$ also receives the output of P_2, and eventually outputs a bit $b \in \{0, 1\}$.

For any two-party PCI protocol Π, any adversary $\mathcal{A}$, any simulator $\mathcal{S}$, and any environment $\mathcal{Z}$, define the following random variables:

- $\mathsf{real}_{\Pi,\mathcal{A},\mathcal{Z}}$: a random variable that denotes the output of the environment $\mathcal{Z}$ after interacting with the adversary $\mathcal{A}$ during an execution of the real-world protocol Π.
- $\mathsf{ideal}_{\mathcal{F}_{\mathsf{PCI}},\mathcal{S},\mathcal{Z}}$: a random variable that denotes the output of the environment $\mathcal{Z}$ after interacting with the simulator $\mathcal{S}$ in the ideal world.

Definition 5 (Secure Two-Party PCI)**.** A PCI protocol Π securely emulates the ideal functionality $\mathcal{F}_{\mathsf{PCI}}$ described in Figure 6.3 if for any security parameter $\lambda \in \mathbb{N}$ and any probabilistic polynomial time (PPT) adversary $\mathcal{A}$, there exists a PPT simulator $\mathcal{S}$ such that, for

any PPT environment $\mathcal{Z}$,

$$|\Pr[\mathsf{real}_{\Pi,\mathcal{A},\mathcal{Z}} = 1] - \Pr[\mathsf{ideal}_{\mathcal{F}_{\mathsf{PCI}},\mathcal{S},\mathcal{Z}} = 1]| \leq \mathsf{negl}(\lambda)$$

Multi-Party PCI. Our definition of two-party PCI naturally extends to the more general setting of multi-party PCI involving n parties $P_1, \ldots, P_n$. We defer a formal treatment of multi-party PCI to Appendix A.

Next we describe a generic approach to achieving a semi-honest secure PCI-mode protocol for mode $\in \{\mathsf{Any}, \mathsf{Any\text{-}DC}, \mathsf{All}\}$ given any semi-honest secure private set intersection (PSI) protocol. We then discuss on why the generic construction is practically infeasible and why we need concretely efficient PCI protocols in practice.

6.2.3 Generic Construction of PCI

In this section, we describe a generic approach to achieving a semi-honest secure PCI-mode protocol for mode $\in \{\mathsf{Any}, \mathsf{Any\text{-}DC}, \mathsf{All}\}$ given any semi-honest secure private set intersection (PSI) protocol. We then discuss some challenges that we face when attempting to upgrade this generic construction to provide malicious security.

Semi-Honest Secure PCI. We show how to construct a semi-honest secure multi-party PCI-Any protocol $\pi_{\mathsf{PCI\text{-}Any,Generic}}$ given a semi-honest secure multi-party PSI protocol π_{PSI}. The constructions of PCI-Any-DC and PCI-All follow analogously.

Suppose that each party P_i for $i \in [1, n]$ inputs a tuple of the form $\mathsf{inp}_i = \big(\mathsf{inp}_{i,1}, \mathsf{inp}_{i,2}\big)$, where $\mathsf{inp}_{i,1} = \{(\mathsf{id}_{i,j}, \sigma_{i,j}, \mathsf{m}_{i,j})\}_{j\in[1,N_i]}$ and $\mathsf{inp}_{i,2} = \{\widehat{\mathsf{m}}_{i,j} \in \mathcal{M}\}_{j\in[1,N_i']}$. The parties proceed as described in Algorithm 5. At a high level, the protocol proceeds in two phases:

- **Phase-1: Filtering.** In this phase, each party filters its input set of (identity, certificate, claim) tuples to identify the subset of identities under which it has a valid certificate on *at least one* public claim.
- **Phase-2: PSI.** The parties then run the secure PSI protocol with these filtered sub-

set of identities as inputs and output the resulting set as the output of the PCI-Any protocol.

Correctness is immediate. Since semi-honest corruption precludes the possibility of malicious behavior, semi-honest security of the overall protocol follows immediately from the semi-honest security of the underlying PSI protocol. Finally, it is straightforward to appropriately modify this protocol for: (i) PCI-Any-DC by additionally including in the filtered subset of identities the set of public claims for which each party has valid certificates under each identity, and (ii) PCI-All by changing Phase-1 to identify the subset of identities under which a party has a valid certificates on *all* of its public claims.

Algorithm 5: $\pi_{\mathsf{PCI\text{-}Any,Generic}}$ **from** π_{PSI}

1 **for** $i := 1 \ldots n$ **do**

2 Each party P_i locally computes a filtered set of identities as:

$$\mathsf{inp}'_i = \left\{\mathsf{id} \in \mathsf{id}(\mathsf{inp}_{1,1}) : \mathbf{R}_{\mathsf{PCI\text{-}Any},\mathsf{inp}_1}(\mathsf{id}) = 1\right\}$$

3 The parties $P_1, \ldots, P_n$ then run the PSI protocol π_{PSI} on the filtered input sets $(\mathsf{inp}'_1, \ldots, \mathsf{inp}'_n)$ to compute an output set

$$\mathsf{out}_{\mathsf{PSI}} = \bigcap_{i \in [1,n]} \mathsf{inp}'_i.$$

4 The parties output $\mathsf{out}_{\mathsf{PCI\text{-}Any}} := \mathsf{out}_{\mathsf{PSI}}$ as the output of the PCI-Any protocol.

Challenges for Malicious Security. The key non-triviality of achieving a secure PCI protocol arises in the setting of malicious corruption, where the generic solution fails (even assuming a maliciously secure PSI protocol) since we can no longer enforce that parties execute Phase-1 honestly. We illustrate this in the simple setting of 2-party PCI. Suppose that in Phase-1 of the generic solution, a malicious P_1 chooses to include in its filtered subset an identity id under which: (i) P_1 does not have a single valid certificate, but (ii) P_2 has one or more valid certificates. Then, the output of Phase-2 allows P_1 to learn more information about the input set of P_2 than is allowed by the ideal functionality $\mathcal{F}_{\mathsf{PCI}}$. Clearly, this is true even if the underlying PSI protocol were maliciously secure.

"Tying" Validation to PSI. In order to enforce malicious security, we need to ensure that for each id in the final result set, each party P_i *proves* to all of the other parties that its input set contains a valid signature under id on some/all of its public claims. This is seemingly hard to achieve efficiently while using certificate validation and the PSI protocol as individual black-boxes since, prior to executing the PSI protocol, the parties do not know the set of identities for which such a proof is required. The parties could choose to provide proofs for all of their inputs, but this leaks more information about their input sets than allowed by $\mathcal{F}_{\mathsf{PCI}}$. To solve this issue, we require a mechanism that somehow "ties" certificate validation to the subsequent PSI protocol, rather than treating these as individual phases.

Maliciously Secure PCI. To upgrade our generic solution for the semi-honest setting outlined in Algorithm 5 to a malicious security setting, we use the following natural approach - run both phases of Algorithm 5 inside a maliciously secure MPC protocol [37, 162]. In particular, Phase-1, which involves validation of claims and creation of filtered identity sets for each party, now happens inside the MPC protocol, and is tied to Phase-2 where the intersection of the identities from the parties is computed[3].

Non-Black-Box Claim Validation. Our generic maliciously-secure MPC protocol is theoretically feasible, but is highly inefficient in practice for almost all widely used cryptographically secure certification mechanisms. For example, the verification algorithms for popular digital signature schemes such as ECDSA [168, 169] and BLS [131, 170, 171] rely on elliptic curve-based finite-field arithmetic operations. Implementing such a verification algorithm inside a maliciously secure MPC protocol would involve non-black-box usage of the various elliptic-curve operations, which is likely to incur huge computational and communication overheads in practice.

6.3 MPC for Elliptic Curve Pairings

As a fundamental building block of our proposed PCI protocols, we introduce a new secret-sharing based MPC framework that is tuned for elliptic curve pairings. In this section, we

[3]Note that Phase-2 can be implemented a simple intersection functionality without the use of a private version in PSI since it's already run inside an MPC.

describe this framework.

On Using Secret-Sharing based MPC. We begin by observing that both the ECDSA and BLS signature schemes fundamentally rely on elliptic curve (EC)-based finite-field arithmetic operations over F_p. Informally speaking, both standard EC operations (i.e., point addition and scalar multiplication over the EC group) and pairing based operations (i.e., algebraic operations over the output group of an EC pairing) share a common algebraic structure with the underlying field F_p (up to group homomorphisms). It turns out that secret-sharing based MPC protocols offer us precisely the desired amount of flexibility to manoeuvre over the algebraic structure of these groups without having to use the group representation/operations in a non-black-box manner. In particular, our overall approach (at a high level) is to design a secret-sharing based MPC protocol that supports EC operations and pairing operations as fundamental building blocks (similar in flavor to addition/multiplication "gates" in standard secret-sharing based MPC over F_p). This enables us to directly use black-box implementations of such operations without having to express them explicitly in terms of the underlying F_p operations.

On Choosing SPDZ. As a concrete instance of secret-sharing based MPC, we use the SPDZ secret sharing based protocol [37] with malicious security against a dishonest majority of adversarial parties. The SPDZ protocol has been widely studied with several extensions [26, 37, 173, 180], optimizations [167, 181], and robust open-source implementations available [36, 182]. In addition, SPDZ naturally supports finite-field arithmetic operations over F_p, which also suits our requirements and overall approach, as outlined above.

Black-Box Usage of Standard Elliptic Curve Libraries. One of our main contributions is augmenting the SPDZ framework as well as the SPDZ open-source implementation to seamlessly support basic elliptic curve operations as well as elliptic curve pairing operations as fundamental gate-level building-blocks. This allows us to directly use standardized and open-source implementations of elliptic curve libraries [174–176]. This is crucial from the point of view of both practical performance and real-world security, since we can immediately leverage both the performance improvements/optimizations as well as the protections against evolving implementation-level attacks that such libraries usually offer. To the best of our knowledge, such a framework was not available before, and this is an independent contribution since it enables an easy implementation of EC pairing-based MPC

protocols.

Our Framework for MPC over EC Pairings. We now detail our framework for designing secret-sharing based MPC protocols over EC pairings. Our framework can be broadly divided into three-tiers, where each tier builds upon the preceding one. We exploit the fact that each tier shares a common algebraic structure (up to group homomorphisms) to progressively support more complicated operations.

- **Tier-1:** This tier of our framework supports the basic operations over F_p for some prime p.
- **Tier-2:** This tier of our framework supports group operations over any generic group $\mathcal{G}$ with order p. We use this tier to implement basic EC operations over the source groups of an EC pairing (i.e., point addition and scalar multiplication), as well as the group operations over the output group of the EC pairing (i.e., multiplication and exponentiation).
- **Tier-3:** This tier of our framework supports EC pairing operations, subject to the restriction that the pairing map e takes its inputs from two source groups $\mathcal{G}_1$ and $\mathcal{G}_2$, both of which have order p, and produces an output in a target group $\mathcal{G}_T$, also of order p.

6.3.1 Tier-1: MPC for Basic F_p Operations

Our starting point is a secret-sharing based MPC engine that implements the ideal functionality $\mathcal{F}[F_P]$ as described in Figure 6.4. For our applications, we use an MPC engine that ensures security against both semi-honest and malicious corruption of parties; the latter would necessitate an additional authentication mechanism to enforce honesty of operations over secret-shared values. We use the representation $[x]_{F_p}$ for any $x \in F_p$ to denote that the value x is secret-shared, i.e., no individual party has access to x, but each party has access to some share of x (for simplicity, we will assume that this notation incorporates the additional authentication components required to ensure malicious security). In the rest of the chapter, we will simply use the notation $[\cdot]$ and drop the subscript F_p when denoting secret-shared values over F_p (we will use explicit subscripts when denoting secret-sharing over other groups, e.g. elliptic curve groups).

$\underline{\mathcal{F}[F_P]}$

Init-F: On input (init, F_p) from all parties, the functionality stores $(domain, F_p)$. A list of identifiers is established for F_p, if not already done before.

Input-F: On input $(\mathsf{inp}F, P_i, \mathsf{varid}, x)$ with $x \in F_p$ from P_i and $(\mathsf{inp}F, P_i, \mathsf{varid}, \phi_{F_p})$ from all other parties, with varid a fresh identifier, the functionality stores (varid, x) in the list of field identifiers.

Rand-F: On input $(\mathsf{rand}, \mathsf{varid})$ from all parties (if varid is not stored in memory), the functionality generates a uniformly random $a \in F_p$ and stores (varid, a) in the list of field identifiers.

Triple-F: On input $(\mathsf{triple}, \mathsf{varid}_1, \mathsf{varid}_2, \mathsf{varid}_3)$ from all parties (if none of the varid_i are stored in memory), the functionality generates a uniformly random $a, b \in F_p$ and computes $c = a \cdot b$ and then stores (varid_1, a), (varid_2, b) and (varid_3, c) in the list of field identifiers.

Add-F: On command $(\mathsf{add}F, \mathsf{varid}_1, \mathsf{varid}_2, \mathsf{varid}_3)$ from all parties where varid1, varid_2 are in the list of field identifiers and varid_3 is not, the functionality retrieves (varid_1, x), (varid_2, y) from the list of field identifiers and stores $(\mathsf{varid}_3, x + y)$ in the list of field identifiers.

Mult-F: On command $(\mathsf{mult}F, \mathsf{varid}_1, \mathsf{varid}_2, \mathsf{varid}_3)$ from all parties where varid_1, varid_2 are in the list of field identifiers and varid_3 is not, the functionality retrieves (varid_1, x), (varid_2, y) from the list of field identifiers and stores $(\mathsf{varid}_3, x \cdot y)$ in the list of field identifiers.

Output-F: On input $(\mathsf{out}F, \mathsf{varid}, i)$ from all honest parties (if varid is present in the list of field identifiers), the functionality retrieves (varid, y) from the set of field identifiers and outputs it to the environment. The functionality waits for an input from the environment. If this input is Deliver then y is output to all parties if $i = 0$, or y is output to party P_i if $i \neq 0$. If the adversarial input is not equal to Deliver then ϕ is output to all parties.

Figure 6.4 Ideal functionality for MPC over field operations in F_p

Linearity-Preservation. Fundamentally, we require that the secret-shared representation $[x]$ is "linearity-preserving", i.e., for any $x, y, z, \alpha, \beta \in F_p$ such that $u = \alpha \cdot x + \beta \cdot y + z$, given the secret shares $[x]$ and $[y]$ and the public values z, α, β, the parties can compute a secret-sharing of u "for free" as

$$[u] = \alpha \cdot [x] + \beta \cdot [y] + z.$$

Note that, in the case of malicious security, we also need this property to be preserved for the authentication components.

Additional Functionalities. We additionally require two deterministic functionalities to be supported by the MPC engine:

1. A functionality that "opens" a secret shared value $[x]$, i.e., reconstructs and distributes the value x to all or a subset of the parties.

2. A functionality that "multiplies" secret shared inputs, i.e., given two secret-shared inputs $[x]$ and $[y]$, produces a secret-shared output $[z]$ such that $z = x \cdot y$.

Finally, we require two randomized functionalities to be supported by the MPC engine:

1. A functionality that generates a secret-shared representation $[a]$ for a randomly sampled value $a \leftarrow F_p$.

2. A functionality that generates secret-shared representations of uniformly random multiplicative "triples", i.e., it generates $[a]$, $[b]$ and $[c]$ for $a, b \leftarrow F_p$ and $c = a \cdot b$.

We refer to $\mathcal{F}[F_P]$ described in Figure 6.4 for a formal description of these functionalities. Note that, for malicious security, we would need each of the above functionalities to also preserve (or, in the case of opening, validate) the authentication components of the output appropriately.

SPDZ-based Realization of Tier-1. While we can use any secret-sharing-based MPC engine that securely realizes $\mathcal{F}[F_P]$, we choose to use SPDZ as a concrete realization, with security against a malicious corruption of the majority of the parties. We briefly recall here that, in addition to securely implementing $\mathcal{F}[F_P]$, SPDZ also implements a MAC-check based authentication mechanism for secret-shared values $[x]$ to achieve active security against malicious corruption of parties. We recall the details of this mechanism at a very high level; the low-level details are not important for understanding our proposed framework. Informally, in SPDZ, each party P_i for $i \in [1, n]$ holds a sharing of a global MAC-key $\alpha \in F_p$ (this sharing follows a slightly different mechanism; we omit the details as our framework is oblivious to the same). Any value $x \in F_p$ is shared as

$$[x] = (\delta, (x_1, \ldots, x_n), (\gamma_1(x), \ldots, \gamma_n(x))),$$

where for each $i \in [n]$, party P_i holds the tuple $(x_i, \gamma_i(x), \delta)$ and where the following invariant holds:

$$x = \sum_{i \in [n]} x_i, \quad \alpha \cdot (x + \delta) = \sum_{i \in [n]} \gamma_i(x).$$

The SPDZ *Opening Protocol.* we briefly recall how the "opening" protocol in SPDZ allows the parties to authenticate, via a MAC-check mechanism, that a secret-shared value has been opened correctly. The opening protocol for a secret-shared value $[x]$ involves the following steps:

- Each party P_i, upon receiving a reconstructed value x', uses its share α_i of the global MAC-key α, as well as $\gamma_i(x)$ and δ, to compute $\sigma_i = \gamma_i(x) - \alpha_i \cdot (x' + \delta)$.
- Each party P_i then broadcasts a commitment $\mathsf{Com}(\sigma_i)$ to all the other parties.
- Finally, each party P_i opens the commitments $\{\mathsf{Com}(\sigma_j)\}$ received from $\{P_j\}_{j \neq i}$, computes $\mathsf{chk} = \sum_{j \in [n]} \sigma_j$, and aborts if $\mathsf{chk} \neq 0$.

We use the term *partial opening* to refer the procedure that just publicly reconstructs the value x without going through the subsequent MAC-check procedure.

Suppose that a malicious adversary $\mathcal{A}$ manages to add an error ϵ during the reconstruction phase, i.e., we have $x' = x + \epsilon$. Suppose also that the adversary $\mathcal{A}$ commits to a subset of false $\{\sigma'_j\}_{j \in \mathcal{C}}$ values corresponding to the subset $\mathcal{C} \subset [n]$ of parties it corrupts. In order to bypass the MAC-check, the adversary $\mathcal{A}$ must ensure that

$$\sum_{j \in \mathcal{C}} (\sigma'_j - \sigma_j) = \alpha\epsilon.$$

However, this happens with probability no greater than $1/p$, since the global MAC value α is uniformly random in F_p and (information-theoretically) unknown to $\mathcal{A}$, and hence, $\mathcal{A}$ cannot bypass the MAC-check protocol except with negligible probability.

Additional Functionalities in SPDZ. We note that the randomized functionalities for generating secret-shared representations of singleton values or multiplicative triples are implemented by the offline phase of SPDZ [167]. We omit the low-level details of these functionalities because they are not necessary to understand our framework and proposed protocols; it suffices to state that our framework uses the native implementations of these functionalities directly from SPDZ. We also directly use SPDZ's implementation of the functionality for multiplying secret-shared values, which is based on generating a random multiplicative

$\underline{\mathcal{F}[\mathcal{G}]}$

Init-G: On input $(\mathsf{init}, \mathcal{G})$ from all parties, the functionality stores $(domain, \mathcal{G})$. A list of identifiers is established for $\mathcal{G}$, if not already done before.

Input-G: On input $(\mathsf{inp}\mathcal{G}, P_i, \mathsf{varid}, g)$ with $g \in \mathcal{G}$ from P_i and $(\mathsf{inp}\mathcal{G}, P_i, \mathsf{varid}, \phi_{\mathcal{G}})$ from all other parties, with varid a fresh identifier, the functionality stores (varid, g) in the list of field identifiers.

Op-G: On command $(\mathsf{op}\mathcal{G}, \mathsf{varid}_1, \mathsf{varid}_2, \mathsf{varid}_3)$ from all parties where varid1, varid_2 are in the list of group identifiers and varid_3 is not, the functionality retrieves (varid_1, g), (varid_2, h) from the list of group identifiers and stores $(\mathsf{varid}_3, g \cdot h)$ in the list of group identifiers, where $\cdot$ is the group operation.

Exp-G-P: On command $(\exp \mathcal{G}P, \mathsf{varid}_1, g, \mathsf{varid}_2)$ from all parties where varid_1 is in the list of field identifiers, $g \in \mathcal{G}$, and varid_2 is a fresh identifier in the list of group identifiers, the functionality retrieves (varid_1, x) from the list of field identifiers and stores (varid_2, g^x).

Exp-G-S: On command $(\exp \mathcal{G}S, \mathsf{varid}_1, \mathsf{varid}_2, \mathsf{varid}_3)$ from all parties where varid_1 is in the list of field identifiers, varid_2 is in the list of group identifiers, and varid_3 is a fresh identifier in the list of group identifiers, the functionality retrieves (varid_1, x) from the list of field identifiers and (varid_2, h) from the list of group identifiers and stores (varid_2, h^x).

Output-G: On input $(\mathsf{out}\mathcal{G}, \mathsf{varid}, i)$ from all honest parties (if varid is present in the list of group identifiers), the functionality retrieves (varid, g) from the set of group identifiers and outputs it to the environment. The functionality waits for an input from the environment. If this input is Deliver then g is output to all parties if $i = 0$, or g is output to party P_i if $i \neq 0$. If the adversarial input is not equal to Deliver then ϕ is output to all parties.

Figure 6.5 Ideal functionality for MPC over the group operations in $\mathcal{G}$, which includes basic EC operations and the operations over the output group of a pairing. We assume that $\mathcal{F}[\mathcal{G}]$ also includes all Tier-1 sub-functionalities in $\mathcal{F}[F_p]$, but we avoid re-writing them for modularity.

triple and then using Beaver's re-randomization technique. We refer to [37, 173] for the details.

6.3.2 Tier-2: MPC over any Generic Group

In Tier-2, we aim to realize an MPC protocol over any generic group $\mathcal{G}$ with prime order p. More concretely, we require the MPC protocol to implement the ideal functionality $\mathcal{F}[\mathcal{G}]$ as described in Figure 6.5. Such a protocol would allow us to support basic EC operations (i.e., point addition and scalar multiplication) over the source groups of an EC pairing,

as well as the operations over the target group of the EC pairing (i.e., group multiplication and exponentiation). We note that prior works [26, 183] have discussed how to realize such an MPC protocol specifically for plain EC groups; here, we generalize their treatment to any group $\mathcal{G}$ of order p. In particular, our generalized treatment also encompasses the target group of an EC pairing, which is not an EC group but a multiplicative group over an extension field of F_p.

As in Tier-1, we aim to design an MPC engine for Tier-2 that ensures security against both semi-honest and malicious corruption of parties; the latter again necessitates some additional authentication mechanism to enforce honesty of operations over secret-shared group elements. We use the representation $[g]_{\mathcal{G}}$ for any group element $g \in \mathcal{G}$ to denote that g is secret-shared, i.e., no individual party has access to g, but each party has access to some share of g (again, for simplicity, we will assume that this notation incorporates the additional authentication components required to ensure malicious security).

Linearity-Preservation. In the context of $\mathcal{G}$, we say that the representation $[\cdot]_{\mathcal{G}}$ is linearity-preserving if for any $g_1, g_2, g_3 \in \mathcal{G}$ and any $\alpha, \beta \in Z_p$ such that $h = g_1^{\alpha} \cdot g_2^{\beta} \cdot g_3$, given the secret shares $[g_1]_{\mathcal{G}}$ and $[g_2]_{\mathcal{G}}$ and the public values g_3, α, β, the parties can compute a secret-sharing of h "for free" as

$$[h]_{\mathcal{G}} = {[g_1]_{\mathcal{G}}}^{\alpha} \cdot {[g_2]_{\mathcal{G}}}^{\beta} \cdot g_3.$$

Once again, in the case of malicious security, we need this property to be preserved for the authentication components.

Additional Functionalities. We additionally require three deterministic functionalities to be supported by the MPC engine:

1. A functionality that "opens" a secret shared value $[g]_{\mathcal{G}}$, i.e., reconstructs and distributes the group element g to all or a subset of the parties.

2. A functionality that "exponentiates" a publicly available group element in $\mathcal{G}$ using a secret-shared value in Z_p, i.e., given a public $g \in \mathcal{G}$ and a secret-shared value $[x]$ for $x \in Z_p$, produces a secret-shared output $[h]_{\mathcal{G}}$ such that $h = g^x$.

3. A functionality that "exponentiates" a secret-shared group element in $\mathcal{G}$ using a secret-shared value in Z_p, i.e., given a secret-shared element $[g]_{\mathcal{G}}$ for $g \in \mathcal{G}$ and a secret-shared value $[x]$ for $x \in Z_p$, produces a secret-shared output $[h]_{\mathcal{G}}$ such that $h = g^x$.

We refer to $\mathcal{F}[\mathcal{G}]$ described in Figure 6.5 for a formal description of these functionalities. Once again, for malicious security, we would need each of the above functionalities to preserve (or, in the case of opening, validate) the authentication components of the output appropriately.

Tier-2 Extension of SPDZ. As a concrete instantiation of $\mathcal{F}[\mathcal{G}]$, we generalize the extensions to SPDZ for basic EC operations proposed in [26, 183] to any generic group of order p. We briefly recall the details of the approach, albeit in its generalized form. At a high level, we exploit the homomorphic relationship between the additive group over Z_p and the group $\mathcal{G}$, which yields a natural way to map the linearity-preserving property of SPDZ over Z_p to its extension over $\mathcal{G}$. Informally speaking, for $h = g^x$ for some publicly available generator g of $\mathcal{G}$, let $[h]_{\mathcal{G}} := g^{[x]}$. Then, observe that the linearity-preservation property in $\mathcal{G}$ follows from the linearity-preservation property in Z_p, albeit implicitly in the exponent of the public group element g.

Concretely, any group element $g \in \mathcal{G}$ is shared as

$$[g]_{\mathcal{G}} = (\delta_{\mathcal{G}}, (g_1, \ldots, g_n), (\gamma_1(g), \ldots, \gamma_n(g))),$$

where for each $i \in [n]$, party P_i holds the tuple $(g_i, \gamma_i(x), \delta_{\mathcal{G}}) \in \mathcal{G} \times \mathcal{G} \times \mathcal{G}$, and where the following invariant holds:

$$g = \prod_{i \in [n]} g_i, \quad (g \cdot \delta_{\mathcal{G}})^{\alpha} = \prod_{i \in [n]} \gamma_i(g),$$

where α is the same global MAC-key as used in Tier-1.

Opening and MAC-Check in $\mathcal{G}$. The opening protocol for a secret-shared group element $[g]_{\mathcal{G}}$ is also analogous to the corresponding protocol for F_p where each party P_i does the following: (a) upon receiving a reconstructed value x', computes $\sigma_i = \gamma_i(g) \big/ (g' \cdot \delta_{\mathcal{G}})^{\alpha_i}$, (b) broadcasts a commitment $\mathsf{Com}(\sigma_i)$ to all the other parties, and (c) opens the commitments

$\{\mathsf{Com}(\sigma_j)\}$ received from $\{P_j\}_{j\neq i}$, computes $\mathsf{chk} = \prod_{j\in[n]} \sigma_j$, and aborts if $\mathsf{chk} \neq \mathrm{id}_{\mathcal{G}}$, where $\mathrm{id}_{\mathcal{G}}$ is the additive identity for the group $\mathcal{G}$. We can use a very similar argument as that in Tier-1 to prove that an adversary $\mathcal{A}$ cannot bypass this extended MAC-check protocol over $\mathcal{G}$, except with negligible probability.

Exponentiating a Public Element in $\mathcal{G}$. As mentioned in prior works [183], exponentiating a publicly available group element in $\mathcal{G}$ using a secret-shared value in Z_p is immediate; given a public group element g and a secret-sharing of x of the form

$$[x] = (\delta, (x_1, \ldots, x_n), (\gamma_1(x), \ldots, \gamma_n(x))),$$

one can easily compute a secret-sharing of $h = g^x$ as

$$[h]_{\mathcal{G}} = g^{[x]} := \left(g^{\delta}, (g^{x_1}, \ldots, g^{x_n}), \left(g^{\gamma_1(x)}, \ldots, g^{\gamma_n(x)}\right)\right).$$

Exponentiating a Secret-Shared Element in $\mathcal{G}$. In order to exponentiate a secret-shared group element $[g]_{\mathcal{G}}$ using a secret-shared value $[x]$, the parties use a protocol that naturally extends SPDZ's implementation of the functionality for multiplying secret-shared values (based on generating a random multiplicative triple and then using Beaver's re-randomization technique). Concretely, the parties follows the following steps:

- Generate $[a]$, $[b]$ and $[c]$ for $a, b \leftarrow Z_p$ and $c = a \cdot b$ using the triple-generation functionality in Tier-1
- Locally compute $[h_1]_{\mathcal{G}} = g^{[b]}$ and $[h_2]_{\mathcal{G}} = g^{[c]}$ using the exponentiation algorithm outlined above.
- Partially open the values $\epsilon = (x - a)$ and $h_3 = g/h_1$.
- Locally compute $[h_4]_{\mathcal{G}} = h_3^{[a]}$ (using the exponentiation algorithm outlined above) and $h_5 = h_3^{\epsilon}$.
- Locally compute $[h]_{\mathcal{G}} = [h_2]_{\mathcal{G}} \cdot \left([h_1]_{\mathcal{G}}\right)^{\epsilon} \cdot [h_4]_{\mathcal{G}} \cdot h_5$.

Note that the final local computation is allowed by the linearity-preserving property of the secret-sharing over $\mathcal{G}$; we omit the explicit details for simplicity.

$\underline{\mathcal{F}[\mathsf{Pair}]}$

Pair-G1-P: On command $(\mathsf{pair}\mathcal{G}P, g_1, \mathsf{varid}_1, \mathsf{varid}_2)$ from all parties where $g_1 \in \mathcal{G}_1$, varid_1 is in the list of group $\mathcal{G}_2$ identifiers, and varid_2 is a fresh identifier in the list of group $\mathcal{G}_T$ identifiers, the functionality retrieves (varid_1, g_2) from the list of $\mathcal{G}_2$ identifiers and stores $(\mathsf{varid}_2, e(g_1, g_2))$, where e is the pairing function.

Pair-G2-P: On command $(\mathsf{pair}\mathcal{G}P, \mathsf{varid}_1, g_2, \mathsf{varid}_2)$ from all parties where varid_1 is in the list of group $\mathcal{G}_1$ identifiers, $g_2 \in \mathcal{G}_2$, and varid_2 is a fresh identifier in the list of group $\mathcal{G}_T$ identifiers, the functionality retrieves (varid_1, g_1) from the list of $\mathcal{G}_1$ identifiers and stores $(\mathsf{varid}_2, e(g_1, g_2))$, where e is the pairing function.

Pair-S: On command $(\mathsf{pair}S, \mathsf{varid}_1, \mathsf{varid}_2, \mathsf{varid}_3)$ from all parties where varid_1 is in the list of group $\mathcal{G}_1$ identifiers, varid_2 is in the list of group $\mathcal{G}_2$ identifiers, and varid_3 is a fresh identifier in the list of group $\mathcal{G}_T$ identifiers, the functionality retrieves (varid_1, g_1) from the list of $\mathcal{G}_1$ identifiers, (varid_2, g_2) from the list of $\mathcal{G}_2$ identifiers and stores $(\mathsf{varid}_3, e(g_1, g_2))$.

Figure 6.6 Ideal functionality for MPC over the EC pairing operation with $\mathcal{G}_1$ and $\mathcal{G}_2$ as the input groups and $\mathcal{G}_T$ as the target group. We assume that $\mathcal{F}[\mathsf{Pair}]$ also includes all Tier-1 and Tier-2 sub-functionalities in $\mathcal{F}[F_p]$ and $\mathcal{F}[\mathcal{G}]$, but we avoid re-writing them for modularity.

Remark. To the best of our knowledge, a full-fledged implementation of this engine was not publicly available (even for plain EC groups). Note that we use all group operations in a black-box manner, which allows us to easily integrate with (i) any MPC framework that supports secret-sharing based MPC protocols realizing the functionality $\mathcal{F}[F_p]$, and (ii) any EC frameworks that implements EC operations including pairings. In this work, we augment the open-source implementation in the MP-SPDZ framework [36] in a black-box manner with an implementation of Tier-2 supporting EC operations over any group $\mathcal{G}$ of order p, using OpenSSL [174] and RELIC [176] which are widely-used libraries for EC operations (here RELIC supports EC pairings).

6.3.3 Tier-3: MPC over EC Pairings

We now build upon the infrastructure set up in Tier-1 and Tier-2 and design the MPC engine to support EC pairing operations. In particular, for a bilinear pairing $e : \mathcal{G}_1 \times \mathcal{G}_2 \to \mathcal{G}_T$, we start with Tier-2 instances for each of the groups $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{G}_T$ (all of which satisfy linearity-preserving and support the operations outlined earlier), and realize the following

three deterministic functionalities for EC pairings:

1. An EC pairing functionality that pairs a publicly available group element in $\mathcal{G}_1$ with a secret-shared group element in $\mathcal{G}_2$, i.e., given a public $g_1 \in \mathcal{G}_1$ and a secret-shared group element $[g_2]_{\mathcal{G}_2}$ for $g_2 \in \mathcal{G}_2$, outputs a secret-shared output $[g_T]_{\mathcal{G}_T}$ such that $g_T = e(g_1, g_2)$.

2. An EC pairing functionality that pairs a secret-shared group element in $\mathcal{G}_1$ with a publicly available group element in $\mathcal{G}_2$, i.e., given a secret-shared group element $[g_1]_{\mathcal{G}_1}$ for $g_1 \in \mathcal{G}_1$ and a public $g_2 \in \mathcal{G}_2$, produces a secret-shared output $[g_T]_{\mathcal{G}_T}$ such that $g_T = e(g_1, g_2)$.

3. An EC pairing functionality that pairs a secret-shared group element in $\mathcal{G}_1$ with a secret-shared group element in $\mathcal{G}_2$, i.e., given a secret-shared element $[g_1]_{\mathcal{G}_1}$ for $g_1 \in \mathcal{G}_1$ and a secret-shared group element $[g_2]_{\mathcal{G}_2}$ for $g_2 \in \mathcal{G}_2$, outputs a secret-shared output $[g_T]_{\mathcal{G}_T}$ such that $g_T = e(g_1, g_2)$.

We refer to $\mathcal{F}[\mathsf{Pair}]$ described in Figure 6.6 for a formal description of these functionalities. Once again, for malicious security, we would need each of the above functionalities to preserve the authentication components of the output appropriately.

Tier-3 Extension of SPDZ**.** One of our technical contributions is an extension of the SPDZ framework to support MPC protocols realizing $\mathcal{F}[\mathsf{Pair}]$, which we describe here.

Pairing with One Secret-Shared Input. We begin by describing how to compute an EC pairing when one of the input group elements is secret-shared and the other input group element is public. We realize this by exploiting the bilinear property of the EC pairing. Recall that if $e : \mathcal{G}_1 \times \mathcal{G}_2 \to \mathcal{G}_T$ is a bilinear pairing, then for any $g_1, h_1 \in \mathcal{G}_1$ and any $g_2, h_2 \in \mathcal{G}_2$, we have

$$e(g_1 \cdot h_1, g_2) = e(g_1, g_2) \cdot e(h_1, g_2),$$

$$e(g_1, g_2 \cdot h_2) = e(g_1, g_2) \cdot e(g_1, h_2).$$

Now, observe that to pair a publicly available group element in $\mathcal{G}_1$ with a secret-shared

group element in $\mathcal{G}_2$, each party can just locally compute

$$[h_T]_{\mathcal{G}_T} = e\left(h_1, [h_2]_{\mathcal{G}_2}\right),$$

and this yields a valid secret-sharing of pairing output h_T because of: (a) the bilinearity property of e as described above, and (b) the linearity-preservation property of the secret-sharing mechanism over $\mathcal{G}_2$. Pairing a publicly available group element in $\mathcal{G}_2$ with a secret-shared group element in $\mathcal{G}_1$ is analogously straightforward, wherein each party locally computes

$$[h_T]_{\mathcal{G}_T} = e\left([h_1]_{\mathcal{G}_1}, h_2\right).$$

Pairing with Two Secret-Shared Inputs. We now propose a protocol that allows the parties to pair a secret-shared group element $[h_1]_{\mathcal{G}_1}$ with a secret-shared group element $[h_2]_{\mathcal{G}_2}$, the parties follows the following steps. The protocol is inspired by SPDZ's implementation of the functionality for multiplying secret-shared values (based on generating a random multiplicative triple and then using Beaver's re-randomization technique), but needs to be carefully adapted to the setting of EC pairings. Concretely, in our proposed protocol, the parties proceed as follows:

- Generate $[a]$, $[b]$ and $[c]$ for $a, b \leftarrow Z_p$ and $c = a \cdot b$ using the triple-generation functionality in Tier-1.
- Locally compute

  $$[u_1]_{\mathcal{G}_1} = g_1^{[a]}, \quad [u_2]_{\mathcal{G}_2} = g_2^{[b]}, \quad [u_3]_{\mathcal{G}_1} = g_1^{[c]},$$

  using the exponentiation algorithm for public group elements in the Tier-2 MPC engine for $\mathcal{G}_1$ and $\mathcal{G}_2$.
- Partially open the values $h_3 = h_1/u_1 \in \mathcal{G}_1$ and $h_4 = h_2/u_2 \in \mathcal{G}_2$.
- Locally compute

  $$[v_1]_{\mathcal{G}_T} = e\left([u_3]_{\mathcal{G}_1}, g_2\right), \qquad [v_2]_{\mathcal{G}_T} = e\left(h_3, [u_2]_{\mathcal{G}_2}\right)$$
  $$[v_3]_{\mathcal{G}_T} = e\left([u_1]_{\mathcal{G}_1}, h_4\right), \qquad v_4 = e(h_3, h_4).$$

- Locally compute $[h_T]_{\mathcal{G}_T} = [v_1]_{\mathcal{G}_T} \cdot [v_2]_{\mathcal{G}_T} \cdot [v_3]_{\mathcal{G}_T} \cdot v_4$.

Note that the final local computation is allowed by the linearity-preserving property of the secret-sharing over $\mathcal{G}_T$; we omit the explicit details for simplicity. To prove correctness, it suffices to prove that $h_T = v_1 \cdot v_2 \cdot v_3 \cdot v_4$; correctness of the sharing again follows immediately from: (a) the bilinearity property of e described above, and (b) the linearity-preservation property of the secret-sharing mechanism over $\mathcal{G}_1$ and $\mathcal{G}_2$. Observe that

$$
\begin{aligned}
& v_1 \cdot v_2 \cdot v_3 \cdot v_4 \\
& \quad = e\left(u_3, g_2\right) \cdot e\left(h_3, u_2\right) \cdot e\left(u_1, h_4\right) \cdot e(h_3, h_4) \\
& \quad = e\left(g_1^c, g_2\right) \cdot e\left(h_1 \cdot g_1^{-a}, g_2^b\right) \cdot e\left(g_1^a, h_2 \cdot g_2^{-b}\right) \\
& \qquad\qquad\qquad\qquad \cdot e\left(h_1 \cdot g_1^{-a}, h_2 \cdot g_2^{-b}\right) \\
& \quad = e\left(g_1, g_2\right)^c \cdot e\left(h_1, g_2\right)^b \cdot e\left(g_1, g_2\right)^{-ab} \cdot e\left(g_1, g_2\right)^{-ab} \cdot e\left(g_1, h_2\right)^a \\
& \qquad \cdot e\left(h_1, h_2\right) \cdot e\left(h_1, g_2\right)^{-b} \cdot e\left(g_1, h_2\right)^{-a} \cdot e\left(g_1, g_2\right)^{ab} \\
& \quad = e(h_1, h_2) = h_T
\end{aligned}
$$

Importantly, the above computations of pairing of a publicly available and a secret-shared group element, as well as pairing of two secret-shared group elements to obtain a share of the pairing result involve the actual pairing computations limited to the local scope of the parties. Therefore, the framework uses the EC pairing operations as well as other EC group operations only as a black-box, performed by the parties locally.

6.4 PCI-Any-DC using ECDSA signature scheme

In this section, we describe a concrete instantiation of two-party PCI-Any-DC using the ECDSA signature scheme. We subsequently discuss how to extend this scheme to support PCI-Any and PCI-All.

Notations. Let the elliptic curve group $\mathcal{G}$ of prime order p be defined over a field F_p as a set of points $(x, y) \in F_p \times F_p$. Though the EC group $\mathcal{G}$ is an additive group of points over the elliptic curve, we will continue to use the multiplicative notation to ensure uniformity

throughout the chapter. Hence, we will denote point addition between two points Q_1 and Q_2 as $Q_1 \cdot Q_2$, and the scalar multiplication between a point Q and $x \in Z_p$ as Q^x. Let $Q \in \mathcal{G}$ be the generator of the group $\mathcal{G}$ (base point in standard EC parlance), and therefore we have $Q^p = \mathcal{O}$, where $\mathcal{O}$ is the point at infinity (the identity element). For any $Q' \in \mathcal{G}$, we use $[Q']_{\mathcal{G}}$ to denote the linearity preserving secret-sharing of Q'.

The ECDSA Signature Scheme. We briefly recall the key generation, signing, and verification equations for ECDSA.

$\mathsf{KeyGen}(\lambda)$: On input a security parameter λ, the key generation algorithm samples a private signing key $x \leftarrow [1, p-1]$, and computes the public verification key $Y := Q^x$. The algorithm outputs the pair (x, Y).

$\mathsf{Sign}(x, m)$: On input a signing key x and a message $m \in \{0,1\}^*$, the signing algorithm does the following: (i) samples a random $k \leftarrow [1, p-1]$, (ii) computes $R = (x, y) := Q^k$ (a random point on the curve), (iii) computes $r = x \mod p$ and $s = k^{-1}(H(m) + r \cdot x) \mod p$, where $H : \{0,1\}^* \rightarrow [0, p-1]$ denotes a hash function, (iv) repeats (i)-(iii) until $r \neq 0$ and $s \neq 0$. The algorithm finally outputs the signature $\sigma = (r, s)$.

$\mathsf{Verify}(Y, \sigma, m)$: On input a verification key Y, a signature σ and a message m, the verification algorithm computes $u_1 = H(m) \cdot s^{-1} \mod p$, $u_2 = r \cdot s^{-1} \mod p$ and computes $R := (x', y') = Q^{u_1} \cdot Y^{u_2}$. The algorithm outputs 1 if $(x', y') \neq \mathcal{O}$ and $x' = r$, and outputs 0 otherwise.

Protocol overview. The starting point of our protocol is the generic maliciously secure protocol outlined in the introduction where we have the certificate validation and creation of the filtered sets of identities followed by the intersection of the sets from the two parties. We note here that we could have a single certifier issue multiple certificates on multiple different claims, or multiple certificates some of the same claims. However, we prescribe the parties to select only one certificate from a single certifier on one claim, i.e., there is a single (certificate, claim) pair for each certifier input to the protocol. We also expect an honest party to only input valid certificates on its set of public claims (although this is not a strict requirement for our protocol).

Optimizing Verify*:* Our main effort here is to reduce or obviate the non-algebraic operations

Algorithm 6: PCI-Any-DC **using ECDSA**

Private inputs from P_1: $\mathsf{inp}_{1,1} = [(Y_{1,\ell}, s_{1,\ell}^{-1}, \mathsf{m}_{1,\ell})]_{\ell \in [1,N_1]}$

Each $Y_{1,\ell}$ is shared as $[Y_{1,\ell}]_{\mathcal{G}_2}$ using Input-G, and each $s_{1,\ell}^{-1}$ is shared as $\left[s_{1,\ell}^{-1}\right]$ using Input-F.

Public inputs from P_1: $\mathsf{inp}_{1,2} = [(r_{1,\ell}, R_{1,\ell}, \mathsf{m}_{1,\ell})]_{\ell \in [1,N_1]}$

Private inputs from P_2: $\mathsf{inp}_{2,1} = [(Y_{2,\ell}, s_{2,\ell}^{-1}, \mathsf{m}_{2,\ell})]_{\ell \in [1,N_2]}$

Each $Y_{2,\ell}$ is shared as $[Y_{2,\ell}]_{\mathcal{G}_2}$ using Input-G, and each $s_{2,\ell}^{-1}$ is shared as $\left[s_{2,\ell}^{-1}\right]$ using Input-F.

Public inputs from P_2: $\mathsf{inp}_{2,2} = [(r_{2,\ell}, R_{2,\ell}, \mathsf{m}_{2,\ell})]_{\ell \in [1,N_2]}$

P_1 validates each $R_{2,\ell} \neq \mathcal{O}$ and has-x coordinate $r_{2,\ell}$.

P_2 validates each $R_{1,\ell} \neq \mathcal{O}$ and has x-coordinate $r_{1,\ell}$.

▷ **Validate P_1's input signatures**

for $\ell := 1 \ldots N_1$ **do**

  $[u_{1,\ell}] := H(\mathsf{m}_{1,\ell}) \cdot \left[s_{1,\ell}^{-1}\right]$

  $[v_{1,\ell}] := r_{1,\ell} \cdot \left[s_{1,\ell}^{-1}\right]$

  $\left[C_\ell^1\right]_{\mathcal{G}} :=$ Exp-G-P$([u_{1,\ell}], Q) \cdot$ Exp-G-S$([v_{1,\ell}], [Y_{1,\ell}]_{\mathcal{G}_2}) / R_{1,\ell}$

▷ **Validate P_2's input signatures**

for $\ell' := 1 \ldots N_2$ **do**

  $[u_{2,\ell'}] := H(\mathsf{m}_{2,\ell'}) \cdot \left[s_{2,\ell'}^{-1}\right]$

  $[v_{2,\ell'}] := r_{2,\ell'} \cdot \left[s_{2,\ell'}^{-1}\right]$

  $\left[C_{\ell'}^2\right]_{\mathcal{G}} :=$ Exp-G-P$([u_{2,\ell'}], Q) \cdot$ Exp-G-S$([v_{2,\ell'}], [Y_{2,\ell'}]_{\mathcal{G}_2}) / R_{2,\ell'}$

▷ **Match certifier**

The parties agree on public random values $\mathsf{rnd}_1, \mathsf{rnd}_2 \leftarrow Z_p$.

for $\ell := 1 \ldots N_1$ **do**

  for $\ell' := 1 \ldots N_2$ **do**

    Generate secret-shared randomness $[\mathsf{rnd}_{\ell,\ell'}] \leftarrow$ Rand-F.

    $[C]_{\mathcal{G}} := [Y_{1,\ell}]_{\mathcal{G}} / [Y_{2,\ell'}]_{\mathcal{G}}$

    $[C']_{\mathcal{G}} := \left[C_\ell^1\right]_{\mathcal{G}} \cdot \left[C_{\ell'}^2\right]_{\mathcal{G}}^{\mathsf{rnd}_1} \cdot [C]_{\mathcal{G}}^{\mathsf{rnd}_2}$

    $\left[C''_{\ell,\ell'}\right]_{\mathcal{G}} :=$ Exp-G-S$([\mathsf{rnd}_{\ell,\ell'}], [C']_{\mathcal{G}})$

    Output-G$(\left[C''_{\ell,\ell'}\right]_{\mathcal{G}})$

    If $C''_{\ell,\ell'} == \mathcal{O}$, then Output-G$([Y_{1,\ell}]_{\mathcal{G}})$

in the Verify algorithm. In addition to the additions and multiplications, Verify requires an inverse operation in F_p and the extraction of the x-coordinate of an EC point from the point description (which is a trivial task to do in the plaintext world but not so inside an MPC). To do this, we make two observations. First, we note that the unforgeability of the signature scheme is retained if s^{-1} is input instead of s; given a signature (r, s), it is trivial to compute (r, s^{-1}) and hence the unforgeability guarantees are equivalent for (r, s) and (r, s^{-1}). This way the inverse can be done outside the MPC and the parties can provide the corresponding s^{-1} as their secret inputs.

Second, in addition to r, we input the point $R = (r, y)$ by calculating the y-coordinate, and check that the signature verification procedure actually yields the point R (recall that the original ECDSA signature verification algorithm first reconstructs the point R and then extracts its x-coordinate r). If r and R were to be private inputs, the MPC algorithm would have to check that the r is the valid x-coordinate of R to prevent maliciously constructed inputs. We obviate this by making r and R public. Observe that, in the ECDSA signing algorithm, the point R is a uniformly random point in the group $\mathcal{G}$, thus R and its x-coordinate r are statistically independent of the corresponding public key. In other words, the public key is not revealed when r and R are provided, even if the universal set of public keys is available to the adversary. We also note that a malicious adversary cannot forge signatures by inputting an invalid point R' since, given the x-coordinate r and the public description of the elliptic curve group $\mathcal{G}$, one can efficiently compute the two possible EC points the form (r, y) in the group $\mathcal{G}$, and either of these would match the point R reconstructed by the verification algorithm if and only if the original signature (r, s) was valid. At this point, we can perform certificate verification inside MPC using the operations in Tier-2 of our proposed MPC engine.

Computing the intersection: We now perform the intersection of the sets of public keys by subtracting the corresponding elliptic curve points (dividing in the multiplicative notation) and checking if it opens to the identity element (point at infinity). It is important to hide the difference value if it is not the identity; otherwise we leak information about the public keys which are not part of the output set, which is not an allowed leakage according to our definition. So, we randomize the difference before opening while retaining the identity value. Another optimization in our protocol is that we store the information on the validity of the certificates in $[C_l^1]_{\mathcal{G}}$s and $[C_{l'}^2]_{\mathcal{G}}$s and open them along with the variable

$[C]_{\mathcal{G}}$ storing the equality of public keys, as a random linear combination of three variables corresponding to the validity of P_1's certificate, validity of P_2's certificate and the equality of the public keys of the certifiers. This opens to the identity element if and only if all of the three requirements are satisfied.

The detailed description of our PCI-Any-DC protocol for ECDSA is provided in Algorithm 6. Here, each party inputs tuples of (identifier, certificate, claim) with the above discussed modifications as its private input, and the corresponding claim and (r, R) for each tuple as its public input. Note that the validation of P_1's certificates and P_2's certificates will be executed in parallel by the MPC algorithm. We describe the protocol in the $\mathcal{F}[\mathcal{G}]$-hybrid model, i.e., we assume that each sub-functionality in $\mathcal{F}[\mathcal{G}]$ has a secure instantiation. This allows us to define and prove the protocols in a modular way. A concrete instance of the protocol would use the SPDZ-based instantiation described in Section 6.3 to perform ECDSA signature validations while using all operations over the EC group $\mathcal{G}$ in a black-box way.

Correctness and Security. Correctness of the protocol follows immediately. We state the following theorem for the security of the protocol:

Theorem 4. Our proposed PCI-Any-DC protocol for ECDSA signatures as described in Algorithm 6 securely emulates $\mathcal{F}_{\mathsf{PCI}}(\mathsf{PCI\text{-}Any\text{-}DC})$ (for the two-party setting).

Proof Overview. We defer a detailed formal proof of this theorem to Appendix B.1. We provide a brief proof overview here. Informally, we construct a PPT simulator $\mathcal{S}$ that simulates the view of a PPT environment $\mathcal{Z}$, such that that this simulated view is computationally indistinguishable from the real view of $\mathcal{Z}$. The crux of the proof is the following observation: prior to the output stage in Line 25 of Algorithm 6, the entire computation of the protocol is local. Thus, the environment's view, up to this point, will not leak whether inputs used by the honest player P_2 are dummy inputs or the ones that the environment actually provided (this guarantee follows immediately from the security of the underlying MPC framework in the $\mathcal{F}[\mathcal{G}]$ hybrid-model). Hence, the simulator $\mathcal{S}$ can *assume* entirely dummy inputs on behalf of the honest party P_2, and proceed with the simulation exactly as in the protocol.

To handle openings of the $C''_{\ell,\ell'}$ values (Line 25 of Algorithm 6), the simulator $\mathcal{S}$ invokes

the ideal functionality $\mathcal{F}_{\mathsf{PCI}}(\mathsf{PCI\text{-}Any\text{-}DC})$ using the inputs of the corrupt party P_1 and obtains the output of the protocol $\mathsf{out}_{\mathsf{PCI\text{-}Any\text{-}DC}}(\mathsf{inp}_1, \mathsf{inp}_2)$. From the output, the $\mathcal{S}$ knows precisely which (ℓ, ℓ') tuples result in the opening of a $C''_{\ell,\ell'}$ value that is equal to $0_{\mathcal{G}}$, since this corresponds to an intersecting public key Y. Based on this information, $\mathcal{S}$ ensures consistent openings by suitably modifying the simulated share of $C''_{\ell,\ell'}$ corresponding to the honest party P_2 by exploiting the algebraic structure of the EC group and its knowledge of the MAC key α used in the simulation. Finally, to handle openings of $Y_{1,\ell}$ values (Line 26 of Algorithm 6), it suffices for the simulator $\mathcal{S}$ to proceed exactly as in the real protocol. This is because the public keys in the input of the corrupted party P_2 are available to the simulator $\mathcal{S}$ in the clear, and were shared by $\mathcal{S}$ exactly as in the real protocol. We refer to Appendix B.1 for a detailed description of the simulation strategy.

Extension to PCI-Any**.** One can naturally upgrade the above PCI-Any-DC protocol to a PCI-Any protocol that additionally guarantees privacy of the input claims for each party. More concretely, the claims would be secret-shared across the participating parties instead of being publicly available, and all operations on the input claims would have to be performed inside the MPC protocol. While the extension is conceptually simple, it incurs some additional costs. For instance, we can no longer directly use our proposed optimizations to reduce or obviate the non-algebraic operations in the Verify algorithm, and we would incur the additional cost of performing these operations *inside* the underlying MPC protocol. We would also incur the additional cost of hashing the claims *inside* the MPC protocol (since the claims would now be secret-shared as opposed to being publicly available). One could use an MPC-friendly family of hash functions [184], but this would be non-compliant with standardized implementations of ECDSA that typically do not use such hash function families. We leave it as an interesting future direction to investigate optimization strategies that would allow performing the above operations efficiently (i.e., outside the MPC protocol) while ensuring privacy of the input claims *and* maintaining compliance with standardized ECDSA implementations.

Extension to PCI-All**.** The above PCI-Any-DC protocol can also be extended naturally to PCI-All by iterating through all the claims to validate the certificates on these claims by a specific certifier. To enable this, the private inputs will be ordered in a 2-D grid, where each row corresponds to the certificates by a certifier on all the claims in $\mathsf{inp}_{i,1}$, and the protocol

needs to validate $|\mathsf{inp}_{i,1}|$ certificates per certifier inside the MPC protocol. The complexity grows with the number of claims which seems unavoidable since the ECDSA signatures cannot be aggregated across different claims. Therefore in the next section, we introduce an optimized PCI-All protocol using the BLS signature scheme [131] that only requires a single signature verification per certifier inside the MPC protocol.

Extension to Multi-Party PCI-Any-DC**.** Finally, we refer to Appendix C for a discussion on how to extend the above PCI-Any-DC protocol (and its upgradation to PCI-Any) from the two-party to the multi-party setting.

6.5 PCI-All using BLS signature

This section provides a concrete instantiation of the PCI-All protocol using the BLS signature scheme [131, 170, 171]. At a high level, we use the aggregatable feature of BLS signatures over different claims to minimize the number of signature verifications inside the PCI-All protocol. Note however that BLS signature verification involves EC pairings, which we handle in a black-box way using **Tier-3** (Section 6.3) of our proposed MPC engine.

Notations. Let $e : \mathcal{G}_1 \times \mathcal{G}_2 \rightarrow \mathcal{G}_T$ be a non-degenerate, efficiently computable bilinear pairing, where $\mathcal{G}_1, \mathcal{G}_2$ are elliptic curve groups and $\mathcal{G}_T$ is a multiplicative group, all of prime order p. Let Q_1 and Q_2 be generators of $\mathcal{G}_1$ and $\mathcal{G}_2$ respectively, and hence $g_T = e(Q_1, Q_2)$ is a generator of $\mathcal{G}_T$.

The BLS Signature Scheme. We briefly describe the key generation, signing and verification algorithms of the BLS signature scheme, followed by the algorithms for signature aggregation (over multiple messages signed under the same verification key) and the verification of aggregate signatures.

$\mathsf{KeyGen}(\lambda)$: On input a security parameter λ, the key generation algorithm samples a private signing key $x \leftarrow [1, p-1]$ and computes the public verification key as $Y = Q_2^x \in \mathcal{G}_2$. The algorithms outputs the key pair (x, Y).

$\mathsf{Sign}(x, m)$: On input a signing key x and message m, the signing algorithm first computes

$M = H(m) \in \mathcal{G}_1$ where $H : \{0,1\}^* \rightarrow \mathcal{G}_1$. The algorithm then computes and outputs the signature $\sigma = M^x \in \mathcal{G}_1$.

$\mathsf{Verify}(Y, \sigma, m)$: On input a verification key Y, a signature σ and a message m, the verification algorithm outputs 1 if $e(\sigma, Q_2) = e(M, Y)$, and 0 otherwise.

Signature aggregation: On input signature-message pairs $\{\sigma_i, m_i\}_{i\in[1,N]}$, the signature aggregation algorithm produces an aggregated signature $\sigma_{(m_1,\ldots,m_N)} = \prod_{i\in[1,N]} \sigma_i$.

Aggregated signature verification: On input a verification key Y, an aggregated signature $\sigma_{(m_1,\ldots,m_N)}$ and a list/multiset of messages $(m_1, \ldots, m_N)$, the aggregated signature verification algorithm outputs 1 if $e(\sigma_{(m_1,\ldots,m_N)}, Q_2) = \prod_{i\in[1,N]} e(M_i, Y)$ where $M_i = H(m_i)$. The algorithm outputs 0 otherwise.

Remark. We note here that BLS signature aggregation is susceptible to a rogue public key attack when aggregating signatures on the same message under different verification keys. However, the attack is not applicable when aggregating signatures over multiple messages signed under the same public verification key, and hence does not impact the security of our proposed protocol.

Protocol overview. We follow the same generic approach as in our ECDSA-based protocol, with some optimizations to reduce BLS signature verifications inside the MPC protocol. We note here that we could have a single certifier issue multiple certificates on the same claim for some of the claims. However, we prescribe the parties to select only one certificate from a single certifier on each claim, i.e., there is a single (certificate, claim) pair for each certifier per claim input to the protocol. We also expect an honest party to only input valid certificates on its set of public claims.

Reducing Claim Validation: As mentioned earlier, trivially extending the approach used in our ECDSA-based PCI-Any-DC protocol to design a PCI-All protocol would require iterating through all of the public claims, and validate the certificates on these claims by a specific certifier. This results in a claim validation complexity that grows with the number of claims, which is undesirable because the straightforward way of claim validation using BLS signatures would require computing two bilinear pairings inside the MPC protocol per validation, which is prohibitively expensive. Our main effort here is to reduce the number

Algorithm 7: PCI-All **using BLS**

P_1 has $\mathsf{inp}_{1,1} = [(Y_{1,\ell_1}, \sigma_{1,\ell_1,\ell_2}, \mathsf{m}_{1,\ell_2})]_{\ell_1 \in [1,N_{1,1}], \ell_2 \in [1,N_{1,2}]}$ and $\mathsf{inp}_{1,2} = \{\mathsf{m}_{1,\ell_2}\}_{\ell_2 \in [1,N_{1,2}]}$

P_2 has $\mathsf{inp}_{2,1} = [(Y_{2,\ell_1}, \sigma_{2,\ell_1,\ell_2}, \mathsf{m}_{2,\ell_2})]_{\ell_1 \in [1,N_{2,1}], \ell_2 \in [1,N_{2,2}]}$ and $\mathsf{inp}_{2,2} = \{\mathsf{m}_{2,\ell_2}\}_{\ell_2 \in [1,N_{2,2}]}$

Private inputs from P_1: the aggregated tuples and the set of preempted pairings

(i) $\overline{\mathsf{inp}}_{1,1} = \left[\left(Y_{1,\ell}, \overline{\sigma}_{1,\ell}, \overline{M}_1\right)\right]_{\ell \in [1,N_{1,1}]}$

(ii) $\{z_{1,\ell} = e(\overline{M}_2, Y_{1,\ell})\}_{\ell \in [1,N_{1,1}]}$

where $\overline{\sigma}_{i,\ell} = \prod_{\ell_2 \in [1,N_{i,2}]} \sigma_{i,\ell,\ell_2}$ and $\overline{M}_i = \prod_{\ell \in [1,N_{i,2}]} H\left(\mathsf{m}_{i,\ell}\right)$. Note that each $Y_{1,\ell}$ is secret-shared as $[Y_{1,\ell}]_{\mathcal{G}_2}$, each $\overline{\sigma}_{1,\ell}$ is secret-shared as $[\overline{\sigma}_{1,\ell}]_{\mathcal{G}_1}$, and each $z_{1,\ell}$ is secret-shared as $[z_{1,\ell}]_{\mathcal{G}_T}$.

Public inputs from P_1: $\mathsf{inp}_{1,2}$.

Private inputs from P_2: the aggregated tuples and the set of preempted pairings

(i) $\overline{\mathsf{inp}}_{2,1} = \left[\left(Y_{2,\ell}, \overline{\sigma}_{2,\ell}, \overline{M}_2\right)\right]_{\ell \in [1,N_{2,1}]}$

(ii) $\{z_{2,\ell} = e(\overline{M}_1, Y_{2,\ell})\}_{\ell \in [1,N_{2,1}]}$

Note that each $Y_{2,\ell}$ is secret-shared as $[Y_{2,\ell}]_{\mathcal{G}_2}$, each $\overline{\sigma}_{2,\ell}$ is secret-shared as $[\overline{\sigma}_{2,\ell}]_{\mathcal{G}_1}$, and each $z_{2,\ell}$ is secret-shared as $[z_{2,\ell}]_{\mathcal{G}_T}$.

Public inputs from P_2: $\mathsf{inp}_{2,2}$.

for $\ell := 1 \ldots N_{1,1}$ **do**

  $\left[z'_{1,\ell}\right]_{\mathcal{G}_T}$:= Pair-G2-P($[\overline{\sigma}_{1,\ell}]_{\mathcal{G}_1}, Q_2$)

for $\ell' := 1 \ldots N_{2,1}$ **do**

  $\left[z'_{2,\ell'}\right]_{\mathcal{G}_T}$:= Pair-G2-P($[\overline{\sigma}_{2,\ell'}]_{\mathcal{G}_1}, Q_2$)

The parties agree on public random $r \leftarrow Z_p$.

for $\ell := 1 \ldots N_{1,1}$ **do**

  for $\ell' := 1 \ldots N_{2,1}$ **do**

    Generate secret-shared randomness $[r_{\ell,\ell'}] \leftarrow$ Rand-F.

    Each party locally computes:

    $[c_{\ell,\ell'}]_{\mathcal{G}_T} := \left([z_{1,\ell}]_{\mathcal{G}_T} / \left[z'_{2,\ell'}\right]_{\mathcal{G}_T}\right) \cdot \left([z_{2,\ell'}]_{\mathcal{G}_T} / \left[z'_{1,\ell}\right]_{\mathcal{G}_T}\right)^r$

    $\left[c'_{\ell,\ell'}\right]_{\mathcal{G}_T}$:= Exp-G-S $\left([r_{\ell,\ell'}], [c_{\ell,\ell'}]_{\mathcal{G}_T}\right)$

    Output-G$\left(\left[c'_{\ell,\ell'}\right]_{\mathcal{G}_T}\right)$

    if $c'_{\ell,\ell'} == 1_T$ **then**

      Output-G$\left([Y_{1,\ell}]_{\mathcal{G}_2}\right)$

of pairing operations inside the MPC protocol as far as possible. To do this, we first use BLS signature aggregation over multiple claims signed under the same public verification key. Concretely, suppose that the private input $\mathsf{inp}_{i,1}$ for each (honest) party P_i is ordered in a 2-D grid of tuples of the form

$$\mathsf{inp}_{i,1} = \left[\left(Y_{i,\ell_1}, \sigma_{i,\ell_1,\ell_2}, \mathsf{m}_{i,\ell_2}\right)\right]_{\ell_1 \in [1,N_{i,1}], \ell_2 \in [1,N_{i,2}]}$$

with $N_{i,1}$ certifiers and $N_{i,2}$ claims to be validated, where row-ℓ_1 contains certificates of the form σ_{i,ℓ_1,ℓ_2} on the claim m_{i,ℓ_2}, signed by the certifier associated with the verification key Y_{i,ℓ_1}. The party P_i performs some pre-processing to aggregate the certificates in each row using the BLS signature aggregation algorithm as:

$$\overline{\sigma}_{i,\ell_1} = \prod_{\ell_2 \in [1,N_{i,1}]} \sigma_{i,\ell_1,\ell_2}, \quad \overline{M}_i = \prod_{\ell_2 \in [1,N_{i,1}]} H\left(\mathsf{m}_{i,\ell_2}\right)$$

and uses an *aggregated private input* of the form

$$\overline{\mathsf{inp}}_{i,1} = \left[\left(Y_{i,\ell_1}, \overline{\sigma}_{i,\ell_1}, \overline{M}_i\right)\right]_{\ell_1 \in [1,N_{i,1}]}$$

for the MPC protocol. This now reduces the number of pairing computations inside the MPC protocol to two per certifier (required to verify each aggregated certificate); in particular, the complexity no longer grows with the number of public claims to be validated.

The next optimization involves further reducing the number of pairing computations inside the MPC to one per certifier. Note that we could avoid the pairing computation that requires pairing the public key with the aggregated claim-hash by having each party pre-compute this and directly input it to the MPC protocol. Note, however, that doing this naïvely would break the "unforgeability" guarantee of our protocol because a malicious party could simply input the pairing of a (potentially) invalid signature with the group generator Q_2 to trivially satisfy the verification check. To counter this, we exploit the uniqueness of BLS signatures for a given (key, claim) pair as follows: each party preempts the output of pairing its own verification keys with the aggregated claim-hashes of the other party (this is possible since the claims are public), which in the case of an intersecting certifier (i.e. when the verification keys are the same), is identical to the pairing of the aggregated public claim-hashes with the other party's verification key. This enables performing certificate verification for one party by using the preempted pairing values com-

puted by the other party. This obviates the need for computing one of the pairings inside the MPC protocol (since the preempted pairing computation is done outside the MPC), while also preserving security of the end-to-end protocol.

Computing the intersection: In addition to certificate verification, the above step also enables computing the intersection of the identity sets between the two parties. In particular, we perform an equality check in $\mathcal{G}_T$ by simply dividing the corresponding group elements, and checking that the result opens to the identity element in $\mathcal{G}_T$. As in our ECDSA-based protocol, it is important to hide the output of this computation if it is not the identity; otherwise we leak information about the public keys which are not part of the output set, which is not an allowed leakage according to our definition. So, we randomize the difference before opening while retaining the identity value.

The detailed description of our PCI-All protocol for BLS signatures is provided in Algorithm 7. Here, each party P_i inputs tuples of (identifier, aggregated certificate, aggregated claim-hash) as its private input $\mathsf{inp}_{i,1}$, and the corresponding claims for each tuple as part of its public input $\mathsf{inp}_{i,2}$ (for the honest parties, $\mathsf{inp}_{i,2}$ is expected to be simply the set of public claims as in the definition of PCI-All in Section 6.2). Each party also inputs the preempted pairing outputs as described earlier. We describe the protocol in the $(\mathcal{F}[\mathsf{Pair}])$-hybrid model, i.e., we assume that each sub-functionality in $\mathcal{F}[\mathsf{Pair}]$ has a secure instantiation. A concrete instance of the protocol would use the SPDZ-based instantiation described in Section 6.3 to perform BLS signature validations while using all operations over the EC groups $\mathcal{G}_1, \mathcal{G}_2$ and the target group $\mathcal{G}_T$ and the bilinear pairing e in a black-box way.

Correctness and Security. Correctness of the protocol follows immediately. We state the following theorem for the security of the protocol:

Theorem 5. Our proposed PCI-All protocol for BLS signatures as described in Algorithm 7 securely emulates $\mathcal{F}_{\mathsf{PCI}}(\mathsf{PCI\text{-}All})$ (for the two-party setting).

We defer a formal proof of this theorem to Appendix B.2.

Extension to Multi-Party PCI-All**.** We refer to Appendix C for a discussion on how to extend the above PCI-All protocol from the two-party to the multi-party setting.

6.6 Evaluation

This section details our implementation of the EC building blocks, the ECDSA-based PCI-Any-DC protocol, and the BLS-based PCI-All protocol. We independently benchmark the individual components of our protocols (including the protocols for EC operations) in a local server. We then evaluate the end-to-end performance of our PCI-Any-DC and PCI-All protocols in a LAN, an intra-continental WAN, and an inter-continental WAN by spawning parties over three geographic regions across two continents.

6.6.1 Implementation Details

Our implementation builds on the MP-SPDZ [36] framework to support the EC operations, including pairing described in Section 6.3. To the best of our knowledge, this is the first implementation of an MPC protocol that supports all the EC group operations as basic gates. In particular, we implement all the functionalities described in $\mathcal{F}[F_P]$, $\mathcal{F}[\mathcal{G}]$, and $\mathcal{F}[\mathsf{Pair}]$. The closest prior work [26] had implemented only two selected operations – Output-G and Exp-G-P. Our implementation of ECDSA PCI-Any-DC variant uses the standard OpenSSL (3.0) [174] library for EC operations. For the BLS PCI-All variant, we use the RELIC toolkit [176] to compute pairings and the EC operations on the corresponding groups. Both variants protect against malicious adversaries. As described earlier, our implementation builds on the SPDZ protocol with MASCOT [167] pre-processing. Analyzing the single-threaded CPU bottlenecks of the protocols, we have incorporated multi-threading to parallelize parts that individual parties locally execute without involving any communication (such as steps 9, 10, 14, 15, 22, & 23 in Algorithm 6, and 8, 10, & 16 in Algorithm 7). The source code of the implementation is made available here – `https://github.com/ghoshbishakh/pci`.

6.6.2 Component wise performance analysis

The different types of operations involved in the protocols can be categorized into (i) offline *pre-processing*, (ii) *input sharing*, (iii) *local operations* – performed by a party without any

Table 6.1 Throughput (operations per second) for Local EC Operations using RELIC and OpenSSL

	RELIC - Ed25519	OpenSSL - Secp256k1
Op-G	2,254,758	459,801
Exp-G-P	7,281	2,175

Table 6.2 Throughput (operations per second) for Local EC Operations on Pairing-friendly Curves using RELIC

	BLS12-381	BLS12-446	BN-254	BLS12-638
Op-G : $\mathcal{G}_1$	1,079,688	834,877	687,906	435,223
Exp-G-P : $\mathcal{G}_1$	523,529	404,051	296,905	217,412
Op-G : $\mathcal{G}_2$	6,453	4,535	4,228	1,782
Exp-G-P : $\mathcal{G}_2$	3,684	2,683	1,990	1,019
Pair-G-P : $\mathcal{G}_1$, $\mathcal{G}_2$	960	689	508	307

communication involved, e.g., Exp-G-P, (iv) *communication dependent operations* – which require inter-party communication, e.g., Exp-G-S, (v) *output* – which includes MAC-check. We perform experiments to analyze the performance of these different operations in terms of throughput (operations per second) and the impact of network latency on them. We separately compare the performance of local operations, followed by communication dependent operations including pre-processing, input sharing and output.

Platform Used. We used a workstation with dual Intel Xeon Gold 5118 2.30GHz CPUs, with 24 cores, and having 128 GB RAM. The system runs Ubuntu 18.04 operating system with Linux kernel version 4.15.

Local Operations. We start by benchmarking the local EC operations namely Op-G (point addition) and Exp-G-P (scalar multiplication with a point) separately for OpenSSL and RELIC. The throughput values (using a single thread) depicted in Table 6.1 make it evident that the performance of RELIC with Ed25519 [185] curve is significantly better than that of OpenSSL with Secp256k1 [186] curve. Nevertheless, we use OpenSSL for our ECDSA-based implementation of PCI-Any-DC since it one of the most widely-used libraries implementing the ECDSA algorithm [187, 188]. Following this, we evaluate the performance of EC operations on pairing-friendly curves with RELIC and carry out the experiments on

Table 6.3 Throughput (operations per second) for Operations Requiring Communication

	RTT 1ms		RTT 100ms	
Pre-processing	967		267	
Input	261		245	
Output	457		363	
	Single Threaded	Multi Threaded	Single Threaded	Multi Threaded
Exp-G-S : $\mathcal{G}_1$	547	1,280	473	1,121
Exp-G-S : $\mathcal{G}_2$	277	554	257	554
Exp-G-S : $\mathcal{G}_T$	166	322	164	314
Pair-S: $\mathcal{G}_1$, $\mathcal{G}_2$	80	417	78	409

four different curves, namely BLS12-381 [189–191], BN-254 [192], BLS12-446 [193], and BLS12-638 [191]. Table 6.2 summarizes the throughput for Op-G, Exp-G-P, and Pair-G-P for the above four curves. We observe that Op-G and Exp-G-P operations on $\mathcal{G}_2$ are much slower compared to that on $\mathcal{G}_1$, with Pair-G-P being the slowest operation by far. Among the curves benchmarked, BLS12-381 performs the best, and therefore we select this for the end-to-end experiments in Section 6.6.3.

Operations Requiring Communication. Moving to the more interesting benchmarks of the operations involving inter-party communication, namely *Pre-processing*, *Input*, *Output*, and *EC operations* Exp-G-S and Pair-S, we use two different setups – (a) a LAN setup with RTT between two parties being about 1ms, and (b) an emulated WAN setup with RTT of 100ms. In order to vary the link latency, we use the `tc` tool [194] to manipulate the loopback interface. Table 6.3 shows the throughput observed in the single threaded and multi-threaded implementation for Exp-G-S and Pair-S. We observe that *Pre-processing* slows down significantly with increasing latency, so is *Output* but to a lesser extent. The throughput values of Exp-G-S and Pair-S operations slightly drop with increasing latency but, even with a high RTT of 100ms, multithreading significantly increases the throughput, indicating that CPU is a major bottleneck for these operations. This validates the expectation since Exp-G-S and Pair-S are performed in batches and involve only one round of communication in which a batch of tuples are partially opened (see Section 6.3.2 and 6.3.3), thereby limiting the impact of network latency. However, if the batches are split (when a

single batch becomes too large to handle), the impact of the communication latency will increase. Note that we perform this in a setup where bandwidth is sufficient enough to not be a bottleneck, and therefore, does not impact the benchmarks.

6.6.3 End-to-end performance analysis

In order to get real world performance metrics, we evaluate our implementations by placing the parties in the (a) same region – LAN, (b) different regions in the same continent – Continental WAN (WAN), and (c) different continents – Inter-continental WAN (ICWAN).

Platform Used. To gauge the practical performance of PCI on consumer hardware, we carried out the experiments on **(i) AWS EC2 `c6i.xlarge`** virtual machine instances with only 4 vCPUs and 8 GB RAM. The instances were running the Ubuntu 22.04 operating system and were connected with a network having up to 12.5 Gbps bandwidth [195]. In addition, we evaluated PCI-Any-DC on more powerful hardware - **(ii) AWS EC2 `c6i.12xlarge`** virtual machine (VM) instances with 48 vCPUs and 96 GB RAM, connected with a network having up to 18.75 Gbps bandwidth [195].

For the ICWAN setup, we used instances located in Asia (ap-south-1) and North America (us-east-1), with an RTT latency of about 186ms. For WAN, we use two instances in the USA, one in east coast (us-east-1), and another in the west coast (us-west-1) with an RTT latency of about 62ms. For the LAN setup, we spawned the two parties in two separate VMs in the same data center (ap-south-1).

Overall Latency of PCI-Any-DC on consumer hardware. First, we evaluate the end-to-end ECDSA and BLS-based PCI-Any-DC protocols using the less powerful hardware, with each party's input set sizes varying from 10 to 1000. Here, the BLS PCI-Any-DC refers to the BLS PCI-All (Algorithm 7) with the parties using a single claim and its corresponding signature instead of the aggregated claim and signature. Figures Figure 6.7a, Figure 6.7b, and Figure 6.7c show the mean and standard deviations of the latency in LAN, WAN, and ICWAN setups, respectively, taken over multiple runs. The y-axis shows the time taken in seconds in a logarithmic scale. For the input sets of size 10 from each party, the mean time taken is about 0.69 seconds, 8.8 seconds, and 26.4 seconds for the ECDSA PCI-Any-DC

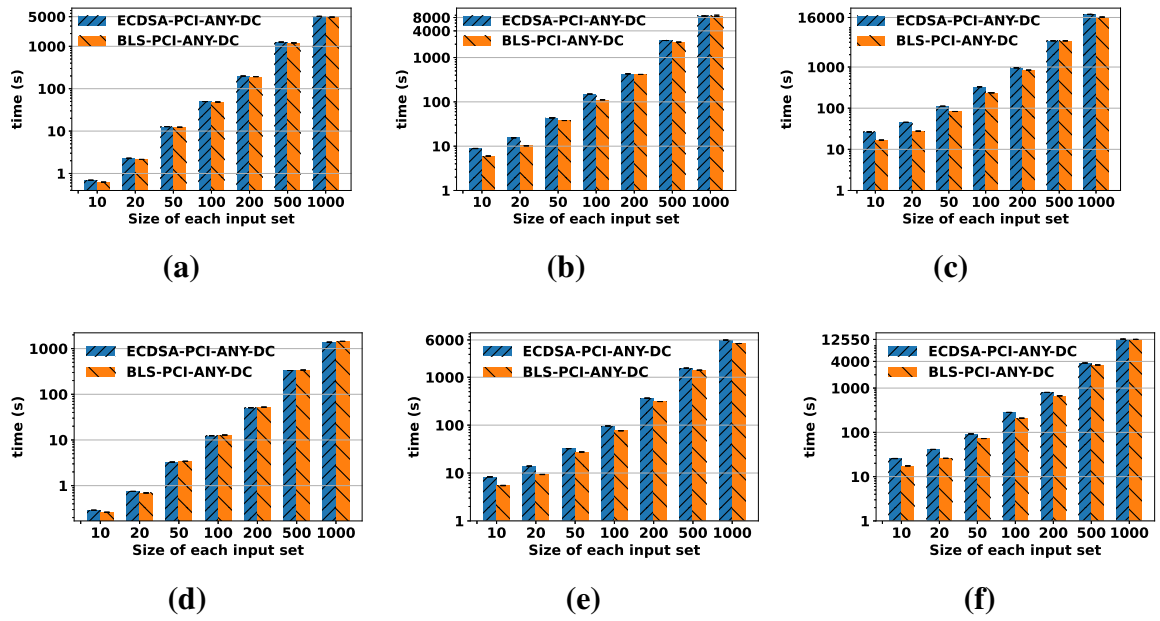


Figure 6.7 (a), (b) and (c) depict latency (in logarithmic scale) of ECDSA PCI-Any-DC vs BLS PCI-Any-DC in LAN, WAN and ICWAN setups respectively on consumer hardware. (d), (e) and (f) depict the latency (in logarithmic scale) in LAN, WAN and ICWAN setups respectively on powerful hardware.

protocol in LAN, WAN, and ICWAN, respectively. In such a setting, the BLS PCI-Any-DC protocol takes 0.62 seconds, 5.9 seconds, and 16.6 seconds respectively. This is better than the ECDSA variant, albeit by a small margin because the ECDSA protocol requires additional Exp-G-S operations in the signature validation steps (lines 11 and 16 of Algorithm 6), which is not required in the BLS variant. Exp-G-S operation requires communication and hence is significantly expensive as analyzed in detail in Section 6.6.2. For 1000 inputs, both ECDSA and BLS PCI-Any-DC takes less than 84 minutes, 149 minutes, and 316 minutes in LAN, WAN, and ICWAN, respectively. Notably, in practice, the size of the centralized trusted set of all CAs on the web is around 200 [30]; therefore, we expect the plausible set of certifiers for a party to be less than 200. Here the number of certifiers do not imply the global set of all possible certifiers, instead it is the number of certifiers that have issued certificates for a given claim to a user. For 200 inputs, both ECDSA and BLS PCI-Any-DC takes less than 3.5 minutes, 7 minutes, and 15 minutes in LAN, WAN, and ICWAN, respectively. This is improved further by using more powerful hardware, which we report next.

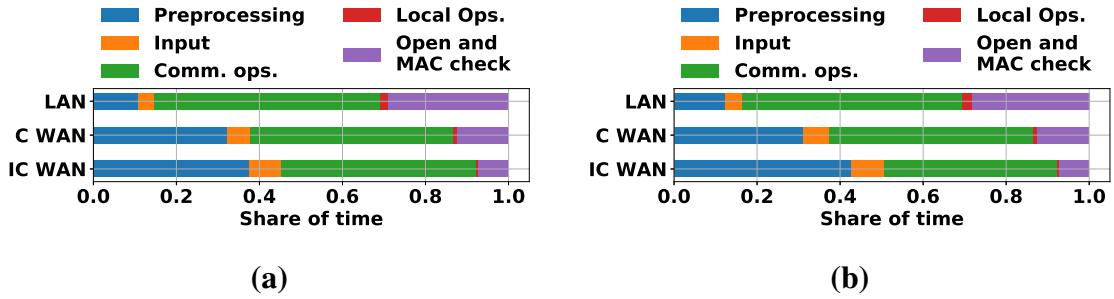
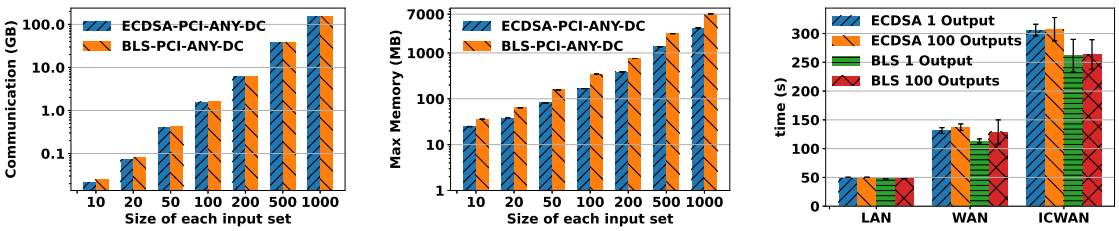


Figure 6.8 (a) and (b) Represents latency of different phases of the ECDSA PCI-Any-DC and BLS PCI-Any-DC respectively with 100 inputs from each party.

Overall Latency of PCI-Any-DC on powerful hardware. We repeat the same set of experiments as mentioned above on powerful hardware. The mean (and standard deviation) of the latency in LAN, WAN, ICWAN setups are presented in Figure 6.7d, Figure 6.7e, and Figure 6.7f respectively. The y-axis shows the time taken in seconds in logarithmic scale. In the LAN setting with 1000 inputs from each party, both ECDSA and BLS PCI-Any-DC take about 24 minutes, which is $\sim 71\%$ less than when using less powerful hardware. However, when the network latency increases in the ICWAN setting, the advantage of more CPU and memory resources reduces. For 1000 inputs from each party, ECDSA PCI-Any-DC takes around 211 minutes and BLS PCI-Any-DC takes 209 minutes. This is $\sim 33\%$ less than when using less powerful hardware where both ECDSA and BLS variant take less than 316 minutes.

Phase-wise Latency Analysis. Next we analyze the latency of different phases of the protocols. Figure 6.8a and 6.8b represent the shares of time taken by different phases, namely pre-processing, input sharing, communication dependent operations, local operations and output (with MAC check) for PCI-Any-DC and PCI-All respectively, with 100 inputs from each party. This experiment is carried out on the more powerful virtual machine instances. We observe that pre-processing, output and input sharing phases have the greatest impact with increases in latency from LAN to ICWAN. The Exp-G-S, and Pair-S operations (denoted as Comm. ops. in the figure) on the other hand are relatively stable with varying latency, but still take the largest share of the entire runtime for our input sizes. Local operations are the cheapest as expected, and their impact on the end-to-end latency drops to



(a) **(b)** **(c)**

Figure 6.9 (a) and (b) represents total communication and maximum memory used respectively (in logarithmic scale) by ECDSA and BLS PCI-Any-DC . (c) presents the latency of PCI-Any-DC with different output intersection sizes.

insignificant percentage shares as latency increases.

Communication and Memory Overhead of PCI-Any-DC**.** We observe that the volume of data communication across parties is deterministic and is defined by the size of their input sets as expected. Hence, there are no variations across the different runs and across LAN, WAN, and ICWAN. We report the communication bandwidth required for different input sizes in Figure 6.9a. With input size of 10 from each party, the total volume of data communicated is 22 MB for ECDSA and 25 MB with BLS PCI-Any-DC. With input sizes of 1000, the total communication goes up to 152.8 GB and 153.4 GB for ECDSA and BLS PCI-Any-DC, respectively. Unlike data communication overhead where ECDSA and BLS variants are close, the memory consumption of BLS is consistently higher as depicted in Figure 6.9b. For 1000 inputs, ECDSA PCI-Any-DC requires around 3.4 GB memory (maximum usage during the runtime), whereas the BLS variant uses around 6.8 GB.

Latency of PCI-Any-DC **with varying output size.** We evaluate the impact of varying overlap in the input certifier sets of the parties implying varying size of output intersection set. Figure 6.9c represents the end-to-end latency of both ECDSA and BLS PCI-Any-DC while keeping the number of input from each party constant at 100, and varying the output size from 1 to 100 using consumer hardware setup. We observe that compared to the output size 1, the end-to-end latency for 100 outputs is higher by a very small margin on an average in all the settings, namely, LAN, WAN, and ICWAN. This is because of the differences in the number of outputs from the protocol that has to be opened (line 26 of Algorithm. 6,

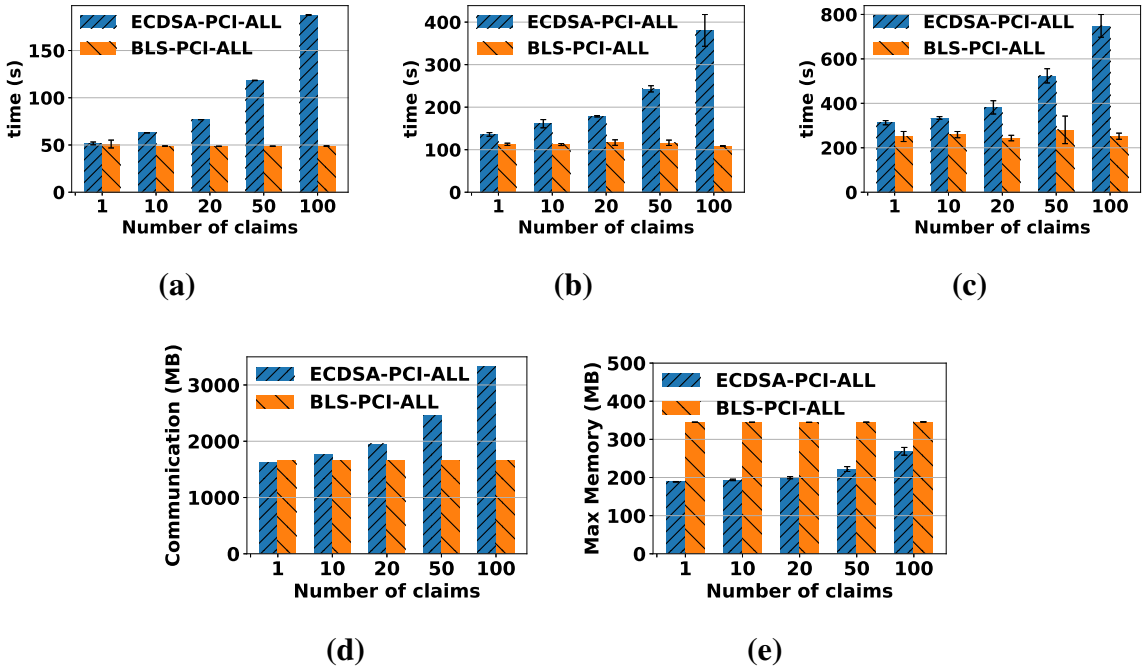


Figure 6.10 (a), (b) and (c) depict latency of ECDSA PCI-All vs BLS PCI-All with 100 certifiers and 1 to 100 claims as input from each party in (a) LAN (b) Continental WAN (c) Inter-continental WAN setups respectively using consumer hardware. (d) and (e) presents the total data communicated and maximum memory consumption of PCI-All respectively.

and line 20 of Algorithm. 7). We note, however, that no additional information is leaked outside what is permitted by the definition of PCI-Any-DC (Section 6.2) from the difference in the latency, since the intersection set is already known to the parties one step prior to this opening phase (line 25 of Algorithm. 6, and line 18 of Algorithm. 7).

Comparing Latency of BLS PCI-All and ECDSA PCI-All.

In order to evaluate the gains of using BLS signature aggregation for PCI-All over the ECDSA implementation, we use a (somewhat artificial) construction of ECDSA-based PCI-All which iterates through all the claims to validate the certificates on them (see Section 6.4). We evaluate the end-to-end latency on consumer hardware by keeping the input set size of each party constant at 100, and increasing the number of claims from 1 to 100. The results in Figure 6.10a, Figure 6.10b, and Figure 6.10c depict the mean and the standard deviation of the overall latency in LAN, WAN and ICWAN setups, respectively, taken over multiple runs. While the BLS PCI-All consistently takes about 50 seconds, 115 seconds and

250 seconds for any number of claims (from 1 to 100) in LAN, WAN, and ICWAN setups, respectively, the time taken by ECDSA PCI-All gradually increases with the increase in the number of claims. ECDSA PCI-All takes on an average 188 seconds, 380 seconds, and 748 seconds for 100 claims in LAN, WAN and ICWAN, respectively. This clearly highlights the gains of using BLS construction of PCI-All.

Communication and Memory Overhead of PCI-All. The volume of data communicated between the two parties for the above scenario is depicted in Figure 6.10d. With increasing number of claims, the communication overhead increases for ECDSA PCI-All, whereas it stays constant for BLS PCI-All which is the expected outcome. For 100 claims, the volume of data communicated by ECDSA PCI-All is 3333 MB, and by BLS PCI-All it is 1658 MB. Memory consumption of ECDSA PCI-All also increases with the increasing number of claims as represented by Figure 6.10e. For 100 certifiers, with 100 claims for each party, the memory usage by ECDSA PCI-All is about 268 MB, and the same by the BLS variant is 345 MB. Overall, the memory consumption overhead of the BLS implementation is more than the ECDSA implementation for up to a reasonable number of claims such as 100.

6.7 Summary

Enabling parties to establish trust by inferring their common certification authorities without revealing their other respective certifiers will emerge as a key privacy goal in any architecture built on decentralized identities and verifiable claims, including Web 3.0. In this chapter, we introduced Private Certifier Intersection (PCI) – a cryptographic primitive that allows mutually distrusting parties to establish a trust basis for cross-validation of claims if they have one or more trust authorities (certifiers) in common. We formalized the security guarantees of PCI and proposed two provably secure and practically efficient PCI protocols supporting validation of digital signature-based certificates: a PCI-Any protocol for ECDSA-based certificates and a PCI-All protocol for BLS-based certificates. Along the way we have introduced a novel framework for efficient secret-sharing-based MPC over elliptic curve pairings. We have implemented and benchmarked our PCI solutions to showcase their practical efficiency.

Our work gives rise to many interesting open questions. We leave it open to study PCI in the setting where claims are private. We also leave it as an open question to define and realize variants of PCI that outputs a priority list of certifiers. Designing practically efficient PCI protocols supporting other widely used cryptographic signature schemes, including quantum-safe schemes, is another challenging open question.

Chapter 7

Conclusion and Future Work

The trend in the adoption of blockchain technology toward enterprise business use cases is a result of its strong decentralization guarantees, which enforce transparency and auditability. Multiple stakeholders form permissioned DLT networks collaborating toward their collective business objectives while not relying on any central trusted authority. Today's permissioned consortium DLT network setups work in isolation with no means of communicating with any external entity. However, enabling interoperability between a DLT network and a separate entity (e.g. another DLT network or an individual / organization) is challenging since the consortium's distributed multi-party trust must be maintained during interoperation. This thesis introduces methods and apparatus for enabling interoperation between different consortium blockchain networks, as well as between consortium networks and other entities outside the consortium boundary.

This thesis first presents a public-private interoperability interface with safety and liveness guarantees. Through a proof-of-concept implementation of a decentralized cloud federation, we show how a permissioned blockchain network of service providers can dispense services to its end users. This public-private interoperability interface, for the first time, enables an end-to-end decentralized supply chain where different business, as well as their end-consumers, communicate and coordinate without any centralized trusted authority. Next, we address the issue of different isolated permissioned networks, which also need to interoperate. While private-private blockchain interoperability protocols exist, we ad-

dress one of their key limitations, which is cross-chain identity configuration. We leverage the DID and VC concepts for designing a decentralized identity management infrastructure to enable different permissioned networks to exchange their identities in a trustworthy manner. Following up in this direction, we explore the problem of privacy-preserving trust negotiation across different permissioned DLTs. We present two solutions for the same, one involving the active participation of the trust anchors, while the other one is based on secure multi-party computation that does not require the trust anchors' participation. In an attempt to generalize the problem of privacy-preserving trust negotiation, even outside the scope of DLTs, we define the problem of Private Certifier Intersection. Through a novel extension of SPDZ MPC protocol for elliptic curve pairings, we devise efficient solutions for this problem. From a bird's-eye view, the contributions of this thesis allow isolated blockchain networks to connect and communicate across themselves and other entities.

7.1 Directions of Future Work

In this section, we look into some of the potential future directions that have opened up with the insights gained from the thesis.

7.1.1 Protocols for Blockchain Network Discovery

Interoperability protocols make otherwise isolated blockchain networks capable of communicating with one another. However, an evolving system of service-providing blockchains coordinating toward more complex business goals can only be realized if there are mechanisms through which such DLT networks can be discovered. A DLT discovery protocol, just like the DNS of the internet, would remove the burden of the manual and ad-hoc process of network configuration. The combination of interoperability protocols along with the discovery protocols could potentially pave the way toward a decentralized internet of blockchains.

7.1.2 Blockchain Network Identifier

Designing a blockchain discovery protocol has an essential prerequisite, – uniquely identifying a DLT network. A blockchain network identifier must be devised to uniquely address a blockchain network as a single entity. Such an identifier has to capture the evolving structure of blockchains over time without altering the address. A governance system has to be designed that would enable the current stakeholders of a DLT network to be in control of its identifier, while allowing new members to join the network and old members to leave.

7.1.3 Efficient Trust Negotiation Protocols

Trust negotiation is the centerpiece of any decentralized identity management system. This thesis introduces protocols for privacy-preserving trust negotiation that are efficient enough for real-world applications. However, the performance of these protocols is limited by the inherent complexity of any secure multi-party computation technique. More efficient protocols for secure privacy-preserving trust negotiation is an exciting open direction to be explored.

Bibliography

[1] M. Sporny, D. Longley, and D. Chadwick, "Verifiable credentials data model v1.0," 2019, (Last accessed: September 7, 2023). [Online]. Available: https://w3c.github.io/vc-data-model/

[2] S. Nakamoto *et al.*, "Bitcoin: A peer-to-peer electronic cash system," 2008, (Last accessed: September 7, 2023). [Online]. Available: https://bitcoin.org/bitcoin.pdf

[3] M. Belotti, N. Božić, G. Pujolle, and S. Secci, "A vademecum on blockchain technologies: When, which, and how," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3796–3838, 2019.

[4] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, 2014, (Last accessed: September 7, 2023). [Online]. Available: https://ethereum.github.io/yellowpaper/paper.pdf

[5] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles*, 2017, pp. 51–68.

[6] B. White, A. Mahanti, and K. Passi, "Characterizing the opensea nft marketplace," in *Companion Proceedings of the Web Conference 2022*, 2022, pp. 488–496.

[7] L. Gudgeon, S. Werner, D. Perez, and W. J. Knottenbelt, "Defi protocols for loanable funds: Interest rates, liquidity and market efficiency," in *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, 2020, pp. 92–112.

[8] X.-J. Jiang and X. F. Liu, "Cryptokitties transaction network analysis: The rise and fall of the first blockchain game mania," *Frontiers in Physics*, vol. 9, p. 57, 2021.

[9] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich *et al.*, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*, 2018, pp. 1–15.

[10] "Tradelens," (Discontinued at the beginning of the year 2023) (Last accessed: September 7, 2023). [Online]. Available: https://www.tradelens.com/

[11] F. Tian, "An agri-food supply chain traceability system for china based on rfid & blockchain technology," in *Proceedings of the 13th International Conference on Service Systems and Service Management*, 2016, pp. 1–6.

[12] M. F. Munoz, K. Zhang, and F. Amara, "Zipzap: A blockchain solution for local energy trading," in *Proceedings of the IEEE International Conference on Blockchain and Cryptocurrency*, 2022, pp. 1–5.

[13] S. Panda, A. Mukherjee, R. Halder, and S. Mondal, "Blockchain-enabled emergency detection and response in mobile healthcare system," in *Proceedings of the IEEE International Conference on Blockchain and Cryptocurrency*, 2022, pp. 1–5.

[14] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, "A cooperative architecture of data offloading and sharing for smart healthcare with blockchain," in *Proceedings of the IEEE International Conference on Blockchain and Cryptocurrency*, 2021, pp. 1–8.

[15] E. Abebe, D. Behl, C. Govindarajan, Y. Hu, D. Karunamoorthy, P. Novotny, V. Pandit, V. Ramakrishna, and C. Vecchiola, "Enabling enterprise blockchain interoperability with trusted data transfer (industry track)," in *Proceedings of the International Middleware Conference Industrial Track*, 2019, pp. 29–35.

[16] "What is a bill of lading?" (Last accessed: September 7, 2023). [Online]. Available: https://www.tradefinanceglobal.com/freight-forwarding/bill-of-lading-bl-bol/

[17] A. Schiff, "Open and closed systems of two-sided networks," *Information Economics and Policy*, vol. 15, no. 4, pp. 425–442, 2003.

[18] R. Belchior, A. Vasconcelos, S. Guerreiro, and M. Correia, "A survey on blockchain interoperability: Past, present, and future trends," *ACM Computing Surveys*, vol. 54, no. 8, pp. 1–41, 2021.

[19] M. Herlihy, "Atomic cross-chain swaps," in *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, 2018, pp. 245–254.

[20] A. Zamyatin, D. Harz, J. Lind, P. Panayiotou, A. Gervais, and W. Knottenbelt, "Xclaim: Trustless, interoperable, cryptocurrency-backed assets," in *Proceedings of the IEEE Symposium on Security and Privacy*, 2019.

[21] I. Bentov, Y. Ji, F. Zhang, L. Breidenbach, P. Daian, and A. Juels, "Tesseract: Real-time cryptocurrency exchange using trusted hardware," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 1521–1538.

[22] M. Westerkamp and A. Küpper, "Smartsync: Cross-blockchain smart contract interaction and synchronization," in *Proceedings of the IEEE International Conference on Blockchain and Cryptocurrency*, 2022, pp. 1–9.

[23] T. Tran, H. Zheng, P. Alvaro, and O. Arden, "Payment channels under network congestion," in *Proceedings of the IEEE International Conference on Blockchain and Cryptocurrency*, 2022, pp. 1–5.

[24] S. Chakraborty and S. Chakraborty, "Proof of federated training: Accountable cross-network model training and inference," in *Proceedings of the IEEE International Conference on Blockchain and Cryptocurrency*, 2022, pp. 1–9.

[25] D. Reed, M. Sporny, D. Longley, C. Allen, R. Grant, M. Sabadello, and J. Holt, "Decentralized identifiers (dids) v1.0," 2020, (Last accessed: September 7, 2023). [Online]. Available: https://w3c.github.io/did-core/

[26] A. Dalskov, C. Orlandi, M. Keller, K. Shrishak, and H. Shulman, "Securing dnssec keys via threshold ecdsa from generic mpc," in *Proceedings of the European Symposium on Research in Computer Security*. Springer, 2020, pp. 654–673.

[27] "IBM Food Trust," (Last accessed: September 7, 2023). [Online]. Available: https://www.ibm.com/in-en/blockchain/solutions/food-trust

[28] "Marco Polo - A Trade Finance Initiative," 2020, (Last accessed: September 7, 2023). [Online]. Available: https://www.marcopolo.finance/

[29] M. Hearn, "Corda: A distributed ledger," *Corda Technical White Paper*, 2016, (Last accessed: September 7, 2023). [Online]. Available: https://www.corda.net/content/corda-technical-whitepaper.pdf

[30] "Ca certificates in firefox," (Last accessed: September 7, 2023). [Online]. Available: https://ccadb-public.secure.force.com/mozilla/CACertificatesInFirefoxReport

[31] E. Syta, I. Tamas, D. Visher, D. I. Wolinsky, P. Jovanovic, L. Gasser, N. Gailly, I. Khoffi, and B. Ford, "Keeping authorities "honest or bust" with decentralized witness cosigning," in *Proceedings of the IEEE Symposium on Security and Privacy*, 2016, pp. 526–545.

[32] "Hyperledger burrow," 2022, (Last accessed: September 7, 2023). [Online]. Available: https://www.hyperledger.org/project/hyperledger-burrow

[33] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *Proceedings of the ACM SIGCOMM Workshop on Hot Topics in Networks*, 2010, pp. 1–6.

[34] "Hyperledger indy," (Last accessed: September 7, 2023). [Online]. Available: https://www.hyperledger.org/use/hyperledger-indy

[35] "Hyperledger aries," (Last accessed: September 7, 2023). [Online]. Available: https://www.hyperledger.org/use/aries

[36] M. Keller, "MP-SPDZ: A versatile framework for multi-party computation," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 1575–1590.

[37] I. Damgård, V. Pastro, N. Smart, and S. Zakarias, "Multiparty computation from somewhat homomorphic encryption," in *Proceedings of the Annual Cryptology Conference*. Springer, 2012, pp. 643–662.

[38] B. Bellaj, A. Ouaddah, E. Bertin, N. Crespi, and A. Mezrioui, "Sok: a comprehensive survey on distributed ledger technologies," in *Proceedings of the IEEE International Conference on Blockchain and Cryptocurrency*, 2022, pp. 1–16.

[39] M. Swan, *Blockchain: Blueprint for a new economy*. " O'Reilly Media, Inc.", 2015.

[40] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, p. 382–401, 1982.

[41] F. B. Schneider, "Implementing fault-tolerant services using the state machine approach: A tutorial," *ACM Computing Surveys*, vol. 22, no. 4, pp. 299–319, 1990.

[42] A. Dorri and R. Jurdak, "Tree-chain: A lightweight consensus algorithm for iot-based blockchains," in *Proceedings of the IEEE International Conference on Blockchain and Cryptocurrency*, 2021, pp. 1–9.

[43] E. J. Scheid, A. Knecht, T. Strasser, C. Killer, M. Franco, B. Rodrigues, and B. Stiller, "Edge2BC: a practical approach for edge-to-blockchain iot transactions," in *Proceedings of the IEEE International Conference on Blockchain and Cryptocurrency*, 2021, pp. 1–9.

[44] S. Huh, S. Cho, and S. Kim, "Managing IoT devices using blockchain platform," in *Proceedings of the International conference on advanced communication technology*. IEEE, 2017, pp. 464–467.

[45] Y. Hu, S. Kumar, and R. A. Popa, "Ghostor: Toward a secure Data-Sharing system from decentralized trust," in *Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation*, 2020, pp. 851–877.

[46] R. D. Garcia, G. S. Ramachandran, R. Jurdak, and J. Ueyama, "A blockchain-based data governance with privacy and provenance: a case study for e-prescription," in *Proceedings of the IEEE International Conference on Blockchain and Cryptocurrency*, 2022, pp. 1–5.

[47] A. Salau, R. Dantu, and K. Upadhyay, "Data cooperatives for neighborhood watch," in *Proceedings of the IEEE International Conference on Blockchain and Cryptocurrency*, 2021, pp. 1–9.

[48] S. Chopra, B. Palanisamy, and S. Sural, "Credit-based peer-to-peer ride sharing using smart contracts," in *Proceedings of the IEEE International Conference on Blockchain and Cryptocurrency*, 2022, pp. 1–3.

[49] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *Proceedings of the ACM SIGSAC conference on computer and communications security*, 2016, pp. 3–16.

[50] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Annual International Cryptology Conference*. Springer, 2017, pp. 357–388.

[51] I. Eyal, A. E. Gencer, E. G. Sirer, and R. V. Renesse, "Bitcoin-NG: A scalable blockchain protocol," in *Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation*, 2016, pp. 45–59.

[52] E. K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford, "Enhancing bitcoin security and performance with strong consistency via collective signing," in *Proceedings of the 25th USENIX Security Symposium*, 2016, pp. 279–296.

[53] K. Wüst and A. Gervais, "Do you need a blockchain?" in *Proceedings of the Crypto Valley Conference on Blockchain Technology*. IEEE, 2018, pp. 45–54.

[54] J. R. Douceur, "The sybil attack," in *Proceedings of the International workshop on peer-to-peer systems*. Springer, 2002, pp. 251–260.

[55] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," in *International conference on financial cryptography and data security*. Springer, 2014, pp. 436–454.

[56] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, "Eclipse attacks on bitcoin's peer-to-peer network," in *Proceedings of the 24th USENIX Security Symposium*, 2015, pp. 129–144.

[57] S. Kasahara and J. Kawahara, "Effect of bitcoin fee on transaction-confirmation process," *arXiv preprint arXiv:1604.00103*, 2016.

[58] M. Castro, B. Liskov *et al.*, "Practical byzantine fault tolerance," in *Proceedings of the 3rd Symposium on Operating Systems Design and Implementation*, 1999, pp. 173–186.

[59] F. Zhang, I. Eyal, R. Escriva, A. Juels, and R. Van Renesse, "{REM}: Resource-efficient mining for blockchains," in *Proceedings of the 26th USENIX Security Symposium*, 2017, pp. 1427–1444.

[60] L. Chen, L. Xu, N. Shah, Z. Gao, Y. Lu, and W. Shi, "On security analysis of proof-of-elapsed-time (poet)," in *Proceedings of the International Symposium on Stabilization, Safety, and Security of Distributed Systems*. Springer, 2017, pp. 282–297.

[61] H. Sukhwani, J. M. Martínez, X. Chang, K. S. Trivedi, and A. Rindos, "Performance modeling of pbft consensus process for permissioned blockchain network (hyperledger fabric)," in *Proceedings of the IEEE Symposium on Reliable Distributed Systems*, 2017, pp. 253–255.

[62] A. Bessani, J. Sousa, and E. E. Alchieri, "State machine replication for the masses with bft-smart," in *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks*, 2014, pp. 355–362.

[63] J. Sousa, A. Bessani, and M. Vukolic, "A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform," in *Proceedings of the IEEE/IFIP international conference on dependable systems and networks*, 2018, pp. 51–58.

[64] J. Kwon, "Tendermint: Consensus without mining," *Draft v. 0.6, fall*, vol. 1, no. 11, 2014, (Last accessed: September 7, 2023). [Online]. Available: https://tendermint.com/static/docs/tendermint.pdf

[65] F. Muratov, A. Lebedev, N. Iushkevich, B. Nasrulin, and M. Takemiya, "Yac: Bft consensus algorithm for blockchain," *arXiv preprint arXiv:1809.00554*, 2018.

[66] P.-L. Aublin, S. B. Mokhtar, and V. Quéma, "RBFT: Redundant byzantine fault tolerance," in *Proceedings of the 33rd IEEE International Conference on Distributed Computing Systems*, 2013, pp. 297–306.

[67] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song, "The honey badger of bft protocols," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 31–42.

[68] S. Duan, M. K. Reiter, and H. Zhang, "Beat: Asynchronous bft made practical," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 2028–2041.

[69] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, "Hotstuff: Bft consensus with linearity and responsiveness," in *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, 2019, p. 347–356.

[70] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou, "A survey of distributed consensus protocols for blockchain networks," *IEEE Communications Surveys & Tutorials*, 2020.

[71] N. Szabo, "Formalizing and securing relationships on public networks," *First Monday*, vol. 2, no. 9, 1997.

[72] R. Cheng, F. Zhang, J. Kos, W. He, N. Hynes, N. Johnson, A. Juels, A. Miller, and D. Song, "Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contracts," in *Proceedings of the IEEE European Symposium on Security and Privacy*, 2019, pp. 185–200.

[73] P. A. Bernstein and N. Goodman, "Multiversion concurrency control—theory and algorithms," *ACM Transactions on Database Systems*, vol. 8, no. 4, pp. 465–483, 1983.

[74] N. Leavitt, "Internet security under attack: The undermining of digital certificates," *Computer*, vol. 44, no. 12, pp. 17–20, 2011.

[75] J. Aas, R. Barnes, B. Case, Z. Durumeric, P. Eckersley, A. Flores-López, J. A. Halderman, J. Hoffman-Andrews, J. Kasten, E. Rescorla *et al.*, "Let's encrypt: an automated certificate authority to encrypt the entire web," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 2473–2487.

[76] C. Allen, "The path to self-sovereign identity," (Last accessed: 28 July, 2022). [Online]. Available: http://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-identity.html

[77] N. Vallina-Rodriguez, J. Amann, C. Kreibich, N. Weaver, and V. Paxson, "A tangled mass: The android root certificate stores," in *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*, 2014, pp. 141–148.

[78] O. Avellaneda, A. Bachmann, A. Barbir, J. Brenan, P. Dingle, K. H. Duffy, E. Maler, D. Reed, and M. Sporny, "Decentralized identity: Where did it come from and where is it going?" *IEEE Communications Standards Magazine*, vol. 3, no. 4, pp. 10–13, 2019.

[79] A. Tobin and D. Reed, "The inevitable rise of self-sovereign identity," *The Sovrin Foundation*, vol. 29, no. 2016, 2016, (Last accessed: September 7, 2023). [Online]. Available: https://sovrin.org/library/inevitable-rise-of-self-sovereign-identity/

[80] C. Li, P. Li, D. Zhou, Z. Yang, M. Wu, G. Yang, W. Xu, F. Long, and A. C.-C. Yao, "A decentralized blockchain with high throughput and fast confirmation," in *Proceedings of the USENIX Annual Technical Conference*, 2020, pp. 515–528.

[81] A. Zamyatin, M. Al-Bassam, D. Zindros, E. Kokoris-Kogias, P. Moreno-Sanchez, A. Kiayias, and W. J. Knottenbelt, "Sok: Communication across distributed ledgers," Cryptology ePrint Archive, 2019, https://eprint.iacr.org/2019/1128.

[82] G. Malavolta, P. Moreno-Sanchez, C. Schneidewind, A. Kate, and M. Maffei, "Anonymous multi-hop locks for blockchain scalability and interoperability." in *Proceedings of the Network and Distributed Systems Security Symposium*, 2019.

[83] "Hashed time-locked contract transactions." *Bitcoin Wik*, 2020, (Last accessed: September 7, 2023). [Online]. Available: https://en.bitcoin.it/wiki/Hash_Time_Locked_Contracts

[84] J. Xu, D. Ackerer, and A. Dubovitskaya, "A game-theoretic analysis of cross-chain atomic swaps with htlcs," in *Proceedings of the 41st IEEE International Conference on Distributed Computing Systems*. IEEE, 2021, pp. 584–594.

[85] R. Pass, E. Shi, and F. Tramer, "Formal abstractions for attested execution secure processors," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2017, pp. 260–289.

[86] S. A. Thyagarajan, G. Malavolta, and P. Moreno-Sanchez, "Universal atomic swaps: Secure exchange of coins across all blockchains," in *Proceedings of the IEEE Symposium on Security and Privacy*. IEEE, 2022, pp. 1299–1316.

[87] S. A. K. Thyagarajan, A. Bhat, G. Malavolta, N. Döttling, A. Kate, and D. Schröder, "Verifiable timed signatures made practical," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 1733–1750.

[88] V. Sivaraman, S. B. Venkatakrishnan, K. Ruan, P. Negi, L. Yang, R. Mittal, G. Fanti, and M. Alizadeh, "High throughput cryptocurrency routing in payment channel networks," in *Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation*, 2020, pp. 777–796.

[89] P. Gaži, A. Kiayias, and D. Zindros, "Proof-of-stake sidechains," in *Proceedings of the IEEE Symposium on Security and Privacy*. IEEE, 2019, pp. 139–156.

[90] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "Omniledger: A secure, scale-out, decentralized ledger via sharding," in *Proceedings of the IEEE Symposium on Security and Privacy*, 2018.

[91] R. Belchior, A. Vasconcelos, M. Correia, and T. Hardjono, "Hermes: Fault-tolerant middleware for blockchain interoperability," *Future Generation Computer Systems*, vol. 129, pp. 236–251, 2022.

[92] E. Abebe, Y. Hu, A. Irvin, D. Karunamoorthy, V. Pandit, V. Ramakrishna, and J. Yu, "Verifiable observation of permissioned ledgers," in *Proceedings of the IEEE International Conference on Blockchain and Cryptocurrency*, 2021.

[93] D. Behl, P. Kodeswaran, V. Ramakrishna, S. Sen, and D. Vinayagamurthy, "Trusted data notifications from private blockchains," in *Proceedings of the IEEE International Conference on Blockchain*. IEEE, 2020, pp. 53–61.

[94] M. Westerkamp and M. Diez, "Verilay: A verifiable proof of stake chain relay," in *Proceedings of the IEEE International Conference on Blockchain and Cryptocurrency*, 2022, pp. 1–9.

[95] "Ens - ethereum name service," 2020, (Last accessed: September 7, 2023). [Online]. Available: https://ens.domains/

[96] "Identity - polkadot wiki," 2020, (Last accessed: September 7, 2023). [Online]. Available: https://wiki.polkadot.network/docs/en/learn-identity/

[97] "uport - tools for decentralized identity and trusted data," 2020. [Online]. Available: https://www.uport.me/

[98] "Ontology - a blockchain for self-soverign id and data," 2020, (Last accessed: September 7, 2023). [Online]. Available: https://ont.io/

[99] "Corda network," 2020, (Last accessed: September 7, 2023). [Online]. Available: https://corda.network/

[100] Hyperledger, "Hyperledger besu," 2020. [Online]. Available: https://besu.hyperledger.org/

[101] H. Montgomery, H. Borne-Pons, J. Hamilton, M. Bowman, P. Somogyvari, S. Fujimoto, T. Takeuchi, T. Kuhrt, and R. Belchior, "Hyperledger cactus whitepaper: Version 0.1 (early draft)," (Last accessed: September 7, 2023). [Online]. Available: https://github.com/hyperledger/cactus/blob/master/whitepaper/whitepaper.md

[102] M. Winslett, T. Yu, K. E. Seamons, A. Hess, J. Jacobson, R. Jarvis, B. Smith, and L. Yu, "Negotiating trust in the web," *IEEE Internet Computing*, vol. 6, no. 6, pp. 30–37, 2002.

[103] B. Pinkas, T. Schneider, and M. Zohner, "Faster private set intersection based on {OT} extension," in *Proceedings of the 23rd USENIX Security Symposium*, 2014, pp. 797–812.

[104] Y. Huang, D. Evans, and J. Katz, "Private set intersection: Are garbled circuits better than custom protocols?" in *Proceedings of the Network and Distributed Systems Security Symposium*, 2012.

[105] H. Chen, K. Laine, and P. Rindal, "Fast private set intersection from homomorphic encryption," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1243–1255.

[106] B. Pinkas, T. Schneider, G. Segev, and M. Zohner, "Phasing: Private set intersection using permutation-based hashing," in {*USENIX*} *Security*, 2015, pp. 515–530.

[107] P. Rindal and M. Rosulek, "Malicious-secure private set intersection via dual execution," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1229–1242.

[108] L. Kissner and D. Song, "Privacy-preserving set operations," in *Annual International Cryptology Conference*. Springer, 2005, pp. 241–257.

[109] M. J. Freedman, K. Nissim, and B. Pinkas, "Efficient private matching and set intersection," in *Proceedings of the International conference on the theory and applications of cryptographic techniques*. Springer, 2004, pp. 1–19.

[110] D. Dachman-Soled, T. Malkin, M. Raykova, and M. Yung, "Efficient robust private set intersection," in *Proceedings of the International Conference on Applied Cryptography and Network Security*. Springer, 2009, pp. 125–142.

[111] E. D. Cristofaro and G. Tsudik, "Practical private set intersection protocols with linear complexity," in *Proceedings of the Financial Cryptography and Data Security, 14th International Conference, FC 2010*. Springer, 2010.

[112] M. Chase and P. Miao, "Private set intersection in the internet setting from lightweight oblivious PRF," in *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference*. Springer, 2020.

[113] J. Camenisch and G. M. Zaverucha, "Private intersection of certified sets," in *Proceedings of the International Conference on Financial Cryptography and Data Security*. Springer, 2009, pp. 108–127.

[114] D. Bosk, D. Frey, M. Gestin, and G. Piolle, "Hidden issuer anonymous credential," *Proceedings on Privacy Enhancing Technologies*, vol. 4, pp. 571–607, 2022.

[115] J. Bobolz, F. Eidens, S. Krenn, S. Ramacher, and K. Samelin, "Issuer-hiding attribute-based credentials," in *Proceedings of the International Conference on Cryptology and Network Security*. Springer, 2021, pp. 158–178.

[116] D. Balfanz, G. Durfee, N. Shankar, D. Smetters, J. Staddon, and H.-C. Wong, "Secret handshakes from pairing-based key agreements," in *Proceedings of the IEEE Symposium on Security and Privacy*. IEEE, 2003, pp. 180–196.

[117] G. Ateniese, J. Kirsch, and M. Blanton, "Secret handshakes with dynamic and fuzzy matching." in *Proceedings of the Network and Distributed Systems Security Symposium*, vol. 7, no. 24, 2007, pp. 43–54.

[118] J. Haucap and U. Heimeshoff, "Google, Facebook, Amazon, eBay: Is the Internet driving competition or market monopolization?" *International Economics and Economic Policy*, vol. 11, 2014.

[119] M. Hindman, *The Internet trap: How the digital economy builds monopolies and undermines democracy*. Princeton University Press, 2018.

[120] H. Subramanian, "Decentralized blockchain-based electronic marketplaces," *Communications of the ACM*, vol. 61, no. 1, pp. 78–84, 2017.

[121] N. Hynes, D. Dao, D. Yan, R. Cheng, and D. Song, "A demonstration of sterling: a privacy-preserving data marketplace," *Proceedings of the VLDB Endowment*, vol. 11, no. 12, pp. 2086–2089, 2018.

[122] P. Pal and S. Ruj, "BlockV: A blockchain enabled peer-peer ride sharing service," in *Proceedings of the IEEE International Conference on Blockchain*, 2019.

[123] Z. Wang, L. Yang, Q. Wang, D. Liu, Z. Xu, and S. Liu, "ArtChain: Blockchain-enabled platform for art marketplace," in *Proceedings of the IEEE International Conference on Blockchain*, 2019.

[124] Y.-W. Chang, K.-P. Lin, and C.-Y. Shen, "Blockchain technology for e-marketplace," in *IEEE PerCom Workshops*, 2019.

[125] S. Narang, M. Byali, P. Dayama, V. Pandit, and Y. Narahari, "Design of trusted B2B market platforms using permissioned blockchains and game theory," in *Proceedings of the IEEE International Conference on Blockchain and Cryptocurrency*, 2019.

[126] "Onapp-Federation," (Last accessed: 5 Jan, 2021). [Online]. Available: https://onapp.com/onapp-federation/

[127] S. Liu, P. Viotti, C. Cachin, V. Quéma, and M. Vukolić, "XFT: Practical fault tolerance beyond crashes," in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation*, 2016, pp. 485–500.

[128] C. Dwork, N. Lynch, and L. Stockmeyer, "Consensus in the presence of partial synchrony," *Journal of the ACM*, vol. 35, no. 2, pp. 288–323, 1988.

[129] U. Pavloff, Y. Amoussou-Guenou, and S. Tucci-Piergiovanni, "Ethereum proof-of-stake under scrutiny," in *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing*, 2023, pp. 212–221.

[130] Y. Shahsavari, K. Zhang, and C. Talhi, "A theoretical model for fork analysis in the bitcoin network," in *Proceedings of the IEEE International Conference on Blockchain*. IEEE, 2019.

[131] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *Advances in Cryptology — ASIACRYPT 2001*, C. Boyd, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 514–532.

[132] J. Mei, K. Li, Z. Tong, Q. Li, and K. Li, "Profit maximization for cloud brokers in cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 1, pp. 190–203, 2018.

[133] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," in *Symposium Proceedings on Communications Architectures & Protocols*, ser. SIGCOMM '89. ACM, 1989.

[134] S. De Angelis, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, and V. Sassone, "Pbft vs proof-of-authority: Applying the cap theorem to permissioned blockchain," in *Proceedings of the Italian Conference on Cyber Security*, 2018.

[135] "we.trade," (Last accessed: September 7, 2023). [Online]. Available: https://www.ibm.com/case-studies/wetrade-blockchain-fintech-trade-finance/

[136] M. Curry, "Blockchain for kyc: Game-changing regtech innovation," 2018, (Last accessed: September 7, 2023). [Online]. Available: https://www.ibm.com/blogs/insights-on-business/banking/blockchain-kyc-game-changing-regtech-innovation/

[137] A. Kiayias, N. Lamprou, and A.-P. Stouka, "Proofs of proofs of work with sublinear complexity," in *Proceedings of the International Conference on Financial Cryptography and Data Security*. Springer, 2016, pp. 61–78.

[138] A. Kiayias and D. Zindros, "Proof-of-work sidechains," in *Proceedings of the International Conference on Financial Cryptography and Data Security*. Springer, 2019, pp. 21–34.

[139] "Cosmos network - internet of blockchains," 2020, (Last accessed: September 7, 2023). [Online]. Available: https://cosmos.network/

[140] "Polkadot: Decentralized web 3.0 blockchain interoperability platform," 2020. [Online]. Available: https://polkadot.network/

[141] "Membership service provider (msp)," 2020, (Last accessed: September 7, 2023). [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/latest/membership/membership.html

[142] "Did specification registries," 2020, (Last accessed: September 7, 2023). [Online]. Available: https://www.w3.org/TR/did-spec-registries

[143] "Sidetree protocol," (Last accessed: September 7, 2023). [Online]. Available: https://identity.foundation/sidetree/spec/

[144] K. Hamilton-Duffy, R. Grant, and A. Gropper, "Use cases and requirements for decentralized identifiers," 2020, (Last accessed: September 7, 2023). [Online]. Available: https://www.w3.org/TR/did-use-cases/

[145] J. Benaloh and M. De Mare, "One-way accumulators: A decentralized alternative to digital signatures," in *Workshop on the Theory and Application of of Cryptographic Techniques*. Springer, 1993, pp. 274–285.

[146] "Letters of credit," (Last accessed: September 7, 2023). [Online]. Available: http://tfig.unece.org/contents/letters-of-credit.htm

[147] "Sovrin," (Last accessed: September 7, 2023). [Online]. Available: https://sovrin.org/

[148] "Hyperledger fabric ca (certificate authority)," 2020, (Last accessed: September 7, 2023). [Online]. Available: https://hyperledger-fabric-ca.readthedocs.io/en/release-1.4/

[149] "Indy sdk," 2020, (Last accessed: September 7, 2023). [Online]. Available: https://github.com/hyperledger/indy-sdk

[150] "Hyperledger aries cloud agent - python," 2020, (Last accessed: September 7, 2023). [Online]. Available: https://github.com/hyperledger/aries-cloudagent-python

[151] "Hyperledger fabric sdks," 2020, (Last accessed: September 7, 2023). [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/latest/fabric-sdks.html

[152] M. E. Whitman and H. J. Mattord, *Principles of Information Security*, 4th ed. Boston, MA, United States: Course Technology Press, 2011.

[153] "Cordapp," (Last accessed: September 7, 2023). [Online]. Available: https://docs.corda.net/docs/corda-os/4.6/cordapp-overview.html

[154] "Dapp," (Last accessed: September 7, 2023). [Online]. Available: https://ethereum.org/en/developers/docs/dapps/

[155] "Simplified payment verification," (Last accessed: Dec 20, 2021). [Online]. Available: https://wiki.bitcoinsv.io/index.php/Simplified_Payment_Verification

[156] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. K. Miller, A. Poelstra, J. Timón, and P. Wuille, "Enabling blockchain innovations with pegged sidechains," 2014, Accessed on September 7, 2023. [Online]. Available: https://bitcoin.fr/public/divers/docs/sidechains.pdf

[157] Z. Liu, Y. Xiang, J. Shi, P. Gao, H. Wang, X. Xiao, B. Wen, and Y.-C. Hu, "Hyperservice: Interoperability and programmability across heterogeneous blockchains," in *Proceedings of the ACM SIGSAC conference on computer and communications security*, 2019, pp. 549–566.

[158] M. Westerkamp, "Verifiable smart contract portability," in *Proceedings of the IEEE International Conference on Blockchain and Cryptocurrency*, 2019.

[159] Z. Jaroucheh and I. A. Álvarez, "Secretation: Toward a decentralised identity and verifiable credentials based scalable and decentralised secret management solution," in *Proceedings of the IEEE International Conference on Blockchain and Cryptocurrency*, 2021, pp. 1–9.

[160] K. Czajkowski, I. Foster, C. Kesselman, V. Sander, and S. Tuecke, "Snap: A protocol for negotiating service level agreements and coordinating resource management in distributed systems," in *Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing*. Springer, 2002, pp. 153–183.

[161] A. Group, "Decentralized identity: Passport to web3," Nov 2021, (Last accessed: Dec 20, 2021). [Online]. Available: https://medium.com/amber-group/decentralized-identity-passport-to-web3-d3373479268a

[162] A. C. Yao, "Protocols for secure computations," in *Proceedings of the 23rd annual symposium on foundations of computer science (sfcs 1982)*. IEEE, 1982, pp. 160–164.

[163] "PCI protocol - source code." [Online]. Available: https://github.com/ghoshbishakh/private_certifier_intersection

[164] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE transactions on information theory*, vol. 31, no. 4, pp. 469–472, 1985.

[165] A. C.-C. Yao, "How to generate and exchange secrets," in *27th Annual Symposium on Foundations of Computer Science*. IEEE, 1986, pp. 162–167.

[166] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game or A completeness theorem for protocols with honest majority," in *Proceedings of the 19th ACM Symposium on Theory of Computing (STOC)*, 1987, pp. 218–229.

[167] M. Keller, E. Orsini, and P. Scholl, "Mascot: faster malicious arithmetic secure computation with oblivious transfer," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 830–842.

[168] X. ANSI, "62: public key cryptography for the financial services industry: the elliptic curve digital signature algorithm (ecdsa)," *Am. Nat'l Standards Inst*, 1999.

[169] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ecdsa)," *International journal of information security*, vol. 1, no. 1, pp. 36–63, 2001.

[170] D. Boneh, M. Drijvers, and G. Neven, "Compact multi-signatures for smaller blockchains," in *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2018, pp. 435–464.

[171] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *International conference on the theory and applications of cryptographic techniques*. Springer, 2003, pp. 416–432.

[172] R. Canetti, A. Cohen, and Y. Lindell, "A simpler variant of universally composable security for standard multiparty computation," in *Annual Cryptology Conference*. Springer, 2015, pp. 3–22.

[173] I. Damgård, M. Keller, E. Larraia, V. Pastro, P. Scholl, and N. P. Smart, "Practical covertly secure mpc for dishonest majority–or: breaking the spdz limits," in *Proceedings of the European Symposium on Research in Computer Security*. Springer, 2013, pp. 1–18.

[174] "OpenSSL," (Last accessed: September 7, 2023). [Online]. Available: https://www.openssl.org/

[175] B. Lynn, "Pbc library-pairing-based cryptography," *http://crypto.stanford.edu/pbc/*, (Last accessed: September 7, 2023).

[176] D. F. Aranha, C. P. L. Gouvêa, T. Markmann, R. S. Wahby, and K. Liao, "RELIC is an Efficient LIbrary for Cryptography," https://github.com/relic-toolkit/relic, (Last accessed: September 7, 2023).

[177] B. Moeller, N. Bolyard, V. Gupta, S. Blake-Wilson, and C. Hawk, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)," RFC 4492, May 2006, (Last accessed: September 7, 2023). [Online]. Available: https://www.rfc-editor.org/info/rfc4492

[178] P. E. Hoffman and W. Wijngaards, "Elliptic Curve Digital Signature Algorithm (DSA) for DNSSEC," RFC 6605, Apr. 2012, (Last accessed: September 7, 2023). [Online]. Available: https://www.rfc-editor.org/info/rfc6605

[179] D. Boneh, S. Gorbunov, R. S. Wahby, H. Wee, and Z. Zhang, "BLS Signatures," Internet Engineering Task Force, Internet-Draft draft-irtf-cfrg-bls-signature-04, Sep. 2020, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-bls-signature-04

[180] R. Cramer, I. Damgård, D. Escudero, P. Scholl, and C. Xing, "Spd $\mathbb{Z}_{2^k}$: efficient mpc mod 2^k for dishonest majority," in *Proceedings of the Annual International Cryptology Conference*. Springer, 2018, pp. 769–798.

[181] M. Keller, V. Pastro, and D. Rotaru, "Overdrive: making spdz great again," in *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2018, pp. 158–189.

[182] A. Aly, K. Cong, D. Cozzo, M. Keller, E. Orsini, D. Rotaru, O. Scherer, P. Scholl, N. Smart, T. Tanguy *et al.*, "Scale–mamba," https://github.com/KULeuven-COSIC/SCALE-MAMBA, (Last accessed: September 7, 2023).

[183] N. P. Smart and Y. Talibi Alaoui, "Distributing any elliptic curve based protocol," in *Proceedings of the IMA International Conference on Cryptography and Coding*. Springer, 2019, pp. 342–366.

[184] L. Grassi, C. Rechberger, D. Rotaru, P. Scholl, and N. P. Smart, "Mpc-friendly symmetric key primitives," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 430–443.

[185] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang, "High-speed high-security signatures," *Journal of cryptographic engineering*, vol. 2, no. 2, pp. 77–89, 2012.

[186] "Sec 2: Recommended elliptic curve domain parameters," (Last accessed: September 7, 2023). [Online]. Available: https://www.secg.org/sec2-v2.pdf

[187] Z. Durumeric, F. Li, J. Kasten, J. Amann, J. Beekman, M. Payer, N. Weaver, D. Adrian, V. Paxson, M. Bailey *et al.*, "The matter of heartbleed," in *Proceedings of the 2014 conference on internet measurement conference*, 2014, pp. 475–488.

[188] M. Nemec, D. Klinec, P. Svenda, P. Sekan, and V. Matyas, "Measuring popularity of cryptographic libraries in internet-wide scans," in *Proceedings of the 33rd Annual Computer Security Applications Conference*, 2017, pp. 162–175.

[189] P. S. Barreto, B. Lynn, and M. Scott, "Constructing elliptic curves with prescribed embedding degrees," in *Proceedings of the International conference on security in communication networks*. Springer, 2002, pp. 257–267.

[190] R. S. Wahby and D. Boneh, "Fast and simple constant-time hashing to the BLS12-381 elliptic curve," *Cryptology ePrint Archive*, 2019.

[191] S. Yonezawa, T. Kobayashi, and T. Saito, "Pairing-friendly curves," *Network Working Group. Internet-Draft. January*, 2019.

[192] P. S. Barreto and M. Naehrig, "Pairing-friendly elliptic curves of prime order," in *Proceedings of the International workshop on selected areas in cryptography*. Springer, 2005, pp. 319–331.

[193] A. de la Piedra, M. Venema, and G. Alpár, "ABE Squared: Accurately benchmarking efficiency of attribute-based encryption," *Cryptology ePrint Archive*, 2022.

[194] "tc - show / manipulate traffic control settings," (Last accessed: September 7, 2023). [Online]. Available: https://man7.org/linux/man-pages/man8/tc.8.html

[195] "Amazon ec2 c6i instances," (Last accessed: September 7, 2023). [Online]. Available: https://aws.amazon.com/ec2/instance-types/c6i/

Appendix A

Multi-Party PCI Definition

In this section, we extend the definition of two-party PCI in Chapter 6.2 to the more general setting of multi-party PCI involving n parties $P_1, \ldots, P_n$. Similar to a two-party PCI protocol, in a multi-party PCI protocol, each party P_i for $i \in [1, n]$ inputs a tuple of the form $\mathsf{inp}_i = \big(\mathsf{inp}_{i,1}, \mathsf{inp}_{i,2}\big)$. We consider the following analogous variations of multi-party PCI protocols:

- **Validate-Any:** Each party P_i for $i \in [1, n]$ receives as output the set

$$\mathsf{out}_{\mathsf{PCI\text{-}Any}}(\mathsf{inp}_1, \ldots, \mathsf{inp}_n) = \big\{\mathsf{id} \in \bigcap_{i \in [1,n]} \mathsf{id}(\mathsf{inp}_{i,1}) : \\ \forall i \in [1, n] \;\; \mathbf{R}_{\mathsf{PCI\text{-}Any},\mathsf{inp}_i}(\mathsf{id}) = 1\big\}$$

  where $\mathbf{R}_{\mathsf{PCI\text{-}Any},\mathsf{inp}_i}(\cdot)$ for each $i \in [1, n]$ is as defined in the two-party case.

- **Leaky Validate-Any:** Each party P_i for $i \in [1, n]$ receives as output the set

$$\mathsf{out}_{\mathsf{PCI\text{-}Any\text{-}DC}}(\mathsf{inp}_1, \ldots, \mathsf{inp}_n) = \Big\{ \big(\mathsf{id}, \{\mathsf{m}_{\mathsf{inp}_i}(\mathsf{id})\}_{i \in [1,n]}\big) : \\ \mathsf{id} \in \mathsf{out}_{\mathsf{PCI\text{-}Any}}(\mathsf{inp}_1, \ldots, \mathsf{inp}_n)\Big\}$$

  where $\mathsf{m}_{\mathsf{inp}_i}(\cdot)$ for each $i \in [1, n]$ is again as defined in the two-party case.

- **Validate-All:** Each party P_i for $i \in [1, n]$ receives as output the set

$$\mathsf{out}_{\mathsf{PCI\text{-}All}}(\mathsf{inp}_1, \ldots, \mathsf{inp}_n) = \big\{\mathsf{id} \in \bigcap_{i \in [1,n]} \mathsf{id}(\mathsf{inp}_{i,1}) : \\ \forall i \in [1, n] \;\; \mathbf{R}_{\mathsf{PCI\text{-}All},\mathsf{inp}_i}(\mathsf{id}) = 1\big\}$$

  where $\mathbf{R}_{\mathsf{PCI\text{-}All},\mathsf{inp}_i}(\cdot)$ for each $i \in [1, n]$ is again as defined in the two-party case.

Security of Multi-Party PCI. We now define the security guarantees expected of a multi-party PCI protocol. Similar to the two-party setting, informally, we require that in any multi-party PCI protocol Π, each party P_i learns nothing about the inputs of the other parties P_j for $j \neq i$ except what is revealed by the output out of the protocol Π, and the size N_j of the input set of party P_j. We again formalize this security guarantee using the simplified universal composability (SUC) framework due to Canetti, Cohen, and Lindell [172] in the real/ideal world paradigm. Our definition is a natural generalization of the security definition of two-party PCI in Section 6.2 to the multi-party setting. We again consider a *dishonest majority* in our definitions, wherein the adversary can corrupt upto $(n-1)$ parties in an n-party PCI protocol.

Ideal Functionality for Multi-Party PCI. We begin by formally defining the ideal functionality $\mathcal{F}_{\mathsf{PCI}}^{(n)}$ for n-party PCI, as described in Figure A.1. This ideal functionality is basically a generalization of the ideal functionality $\mathcal{F}_{\mathsf{PCI}}$ for two-party PCI (defined earlier in Figure 6.3) to the n-party setting.

The Real World. In the real world, the following participants engage in the protocol Π:

- A set $H \subseteq [1, n]$ of honest parties, where for each $i \in H$, the honest party P_i receives its input from the environment $\mathcal{Z}$ and honestly follows the specified protocol Π.
- A real-world adversary $\mathcal{A}$ controlling a set $C \subseteq [1, n]$ of corrupt parties, that interacts with the set H of honest parties and the environment $\mathcal{Z}$.
- The environment $\mathcal{Z}$ that provides each honest party $\{P_i\}_{i \in H}$ with its input, and interacts with the real-world adversary $\mathcal{A}$. The environment $\mathcal{Z}$ also receives the outputs of the honest parties, and eventually outputs a bit $b \in \{0, 1\}$.

$\underline{\mathcal{F}_{\mathsf{PCI}}^{(n)}(\mathsf{mode} \in \{\mathsf{Any}, \mathsf{Any\text{-}DC}, \mathsf{All}\})}$

- Let $H \subseteq [1, n]$ be the set of honest parties, and let $C \subseteq [1, n]$ be the set of corrupt parties.
- For each $i \in [1, n]$, let the input of party P_i be $\mathsf{inp}_i = (\mathsf{inp}_{i,1}, \mathsf{inp}_{i,2})$, where

$$\mathsf{inp}_{i,1} = \{(\mathsf{id}_{i,j}, \sigma_{i,j}, \mathsf{m}_{i,j}) \in \mathcal{ID} \times \mathcal{C} \times \mathcal{M}\}_{j \in [1, N_i]}$$
$$\mathsf{inp}_{i,2} = \{\widehat{\mathsf{m}}_{i,j} \in \mathcal{M}\}_{j \in [1, N_i']}$$

  For each $i \in H$, the input of an honest party P_i is provided directly to $\mathcal{F}_{\mathsf{PCI}}^{(n)}$ by P_i, while for each $i \in C$, the input of a corrupt party P_i is provided to $\mathcal{F}_{\mathsf{PCI}}^{(n)}$ by the simulator $\mathcal{S}$.

- $\mathcal{F}_{\mathsf{PCI}}^{(n)}$ computes $\mathsf{out}_{\mathsf{PCI\text{-}mode}}$, where for $\mathsf{mode} \in \{\mathsf{Any}, \mathsf{Any\text{-}DC}, \mathsf{All}\}$, we have

$$\mathsf{out}_{\mathsf{PCI\text{-}Any}}(\mathsf{inp}_1, \ldots, \mathsf{inp}_n) = \Big\{\mathsf{id} \in \bigcap_{i \in [1,n]} \mathsf{id}(\mathsf{inp}_{i,1}) : \forall i \in [1, n] \;\; \mathbf{R}_{\mathsf{PCI\text{-}Any}, \mathsf{inp}_i}(\mathsf{id}) = 1\Big\}$$
$$\mathsf{out}_{\mathsf{PCI\text{-}Any\text{-}DC}}(\mathsf{inp}_1, \ldots, \mathsf{inp}_n) = \Big\{\Big(\mathsf{id}, \{\mathsf{m}_{\mathsf{inp}_i}(\mathsf{id})\}_{i \in [1,n]}\Big) : \mathsf{id} \in \mathsf{out}_{\mathsf{PCI\text{-}Any}}(\mathsf{inp}_1, \ldots, \mathsf{inp}_n)\Big\}$$
$$\mathsf{out}_{\mathsf{PCI\text{-}All}}(\mathsf{inp}_1, \ldots, \mathsf{inp}_n) = \Big\{\mathsf{id} \in \bigcap_{i \in [1,n]} \mathsf{id}(\mathsf{inp}_{i,1}) : \forall i \in [1, n] \;\; \mathbf{R}_{\mathsf{PCI\text{-}All}, \mathsf{inp}_i}(\mathsf{id}) = 1\Big\}$$

- $\mathcal{F}_{\mathsf{PCI}}^{(n)}$ sends $\big(\mathsf{out}_{\mathsf{PCI\text{-}mode}}, \{N_i, \mathsf{inp}_{i,2}\}_{i \in H}\big)$ to the simulator $\mathcal{S}$.
- If $\mathcal{S}$ responds with an abort, $\mathcal{F}_{\mathsf{PCI}}^{(n)}$ aborts.
- Otherwise, $\mathcal{F}_{\mathsf{PCI}}^{(n)}$ sends $\big(\mathsf{out}_{\mathsf{PCI\text{-}mode}}, \{N_i, \mathsf{inp}_{i,2}\}_{i \in [1,n]}\big)$ to all the parties.

Figure A.1 Ideal functionality $\mathcal{F}_{\mathsf{PCI}}^{(n)}$ for multi-party PCI

The Ideal World. In the ideal world, the following participants interact with the ideal functionality $\mathcal{F}_{\mathsf{PCI}}^{(n)}$ described in Figure 6.3.

- A set $H \subseteq [1, n]$ of honest parties, where for each $i \in H$, the honest party P_i receives its input from the environment $\mathcal{Z}$ and directly forwards this input to $\mathcal{F}_{\mathsf{PCI}}^{(n)}$.
- An ideal-world simulator $\mathcal{S}$ that sends inputs to $\mathcal{F}_{\mathsf{PCI}}^{(n)}$ on behalf of a set $C \subseteq [1, n]$ of corrupt parties, and receives back the corresponding output from $\mathcal{F}_{\mathsf{PCI}}^{(n)}$. $\mathcal{S}$ also

interacts with the environment $\mathcal{Z}$, with the aim of making this interaction indistinguishable from the interaction between the real world $\mathcal{A}$ and the environment $\mathcal{Z}$.

- The environment $\mathcal{Z}$ that provides each honest party $\{P_i\}_{i\in H}$ with its input, and interacts with the simulator $\mathcal{S}$. As in the real world, $\mathcal{Z}$ also receives the outputs of the honest parties, and eventually outputs a bit $b \in \{0,1\}$.

For any multi-party PCI protocol Π, any adversary $\mathcal{A}$, any simulator $\mathcal{S}$, and any environment $\mathcal{Z}$, define the following random variables:

- $\mathsf{real}_{\Pi,\mathcal{A},\mathcal{Z}}$: a random variable that denotes the output of the environment $\mathcal{Z}$ after interacting with the adversary $\mathcal{A}$ during an execution of the real-world protocol Π.
- $\mathsf{ideal}_{\mathcal{F}_{\mathsf{PCI}}^{(n)},\mathcal{S},\mathcal{Z}}$: a random variable that denotes the output of the environment $\mathcal{Z}$ after interacting with the simulator $\mathcal{S}$ in the ideal world.

Definition 6 (Secure Multi-Party PCI)**.** A multi-party PCI protocol Π securely emulates the ideal functionality $\mathcal{F}_{\mathsf{PCI}}^{(n)}$ described in Figure A.1 if for any security parameter $\lambda \in \mathbb{N}$ and any probabilistic polynomial time (PPT) adversary $\mathcal{A}$, there exists a PPT simulator $\mathcal{S}$ such that, for any PPT environment $\mathcal{Z}$, we have

$$\left|\Pr\left[\mathsf{real}_{\Pi,\mathcal{A},\mathcal{Z}} = 1\right] - \Pr\left[\mathsf{ideal}_{\mathcal{F}_{\mathsf{PCI}}^{(n)},\mathcal{S},\mathcal{Z}} = 1\right]\right| \leq \mathsf{negl}(\lambda)$$

Appendix B

Formal Proofs of Security of PCI

In this section we provide a formal proof of both ECDSA-based PCI-Any-DC and BLS-based PCI-All. For an informal overview of the proof of ECDSA PCI-Any-DC see Chapter 6.4.

B.1 Proof of Security of ECDSA-based PCI-Any-DC

In this section, we prove the SUC security of our PCI-Any-DC protocol based on ECDSA. More formally, we prove Theorem 4 by constructing a PPT simulator $\mathcal{S}$ such that no PPT environment $\mathcal{Z}$, who corrupts one of the parties (say P_1 without loss of generality) and chooses the input for P_1, can distinguish with significant probability, a view obtained by running our proposed PCI-Any-DC protocol for ECDSA signatures in Algorithm 6 between a PPT adversary $\mathcal{A}$ and honest party (say P_2 without loss of generality), and a simulated execution of the protocol between $\mathcal{S}$ and $\mathcal{F}_{\mathsf{PCI}}(\mathsf{PCI\text{-}Any\text{-}DC})$ (for the two-party setting). The environment $\mathcal{Z}$'s view consists of the intermediate messages sent and received by the adversary $\mathcal{A}$, the input he chose for the honest party P_2, along with output of P_2.

We now describe the construction of the simulator $\mathcal{S}$, which proceeds as follows:

Input Phase: The simulator internally runs the real-world adversary $\mathcal{A}$ to obtain the private

and public inputs for the corrupt party P_1. Let the inputs be as follows.

Private inputs for P_1: $\mathsf{inp}_{1,1} = [(Y_{1,\ell}, s_{1,\ell}^{-1}, \mathsf{m}_{1,\ell})]_{\ell\in[1,N_1]}$, where each $Y_{1,\ell}$ is shared as $[Y_{1,\ell}]_{\mathcal{G}_2}$ using Input-G, and each $s_{1,\ell}^{-1}$ is shared as $\left[s_{1,\ell}^{-1}\right]$ using Input-F.

Public inputs for P_1: $\mathsf{inp}_{1,2} = [(r_{1,\ell}, R_{1,\ell}, \mathsf{m}_{1,\ell})]_{\ell\in[1,N_1]}$.

Invoking $\mathcal{F}_{\mathsf{PCI}}$(PCI-Any-DC): The simulator $\mathcal{S}$ now invokes the ideal functionality $\mathcal{F}_{\mathsf{PCI}}$(PCI-Any-DC) using the inputs of the corrupt party P_1 (the input of the honest party P_2 is directly provided to $\mathcal{F}_{\mathsf{PCI}}$(PCI-Any-DC) by the environment $\mathcal{Z}$) to receive $(\mathsf{out}_{\mathsf{PCI\text{-}Any\text{-}DC}}(\mathsf{inp}_1, \mathsf{inp}_2), N_2)$, where N_2 is the number of inputs for the honest party P_2, and

$$\mathsf{out}_{\mathsf{PCI\text{-}Any\text{-}DC}}(\mathsf{inp}_1, \mathsf{inp}_2) = \Big(\left\{\overline{\mathsf{m}}(\mathsf{inp}_{i,1})\right\}_{i\in[1,2]}, \\ \left\{\left(Y, \{\mathsf{m}_{\mathsf{inp}_i}(Y)\}_{i\in\{1,2\}}\right) : Y \in \mathsf{out}_{\mathsf{PCI\text{-}Any}}(\mathsf{inp}_1, \mathsf{inp}_2)\right\} \Big)$$

where

$$\mathsf{out}_{\mathsf{PCI\text{-}Any}}(\mathsf{inp}_1, \mathsf{inp}_2) = \big\{Y \in \mathsf{id}(\mathsf{inp}_{1,1}) \cap \mathsf{id}(\mathsf{inp}_{2,1}) : \\ \mathbf{R}_{\mathsf{PCI\text{-}Any},\mathsf{inp}_1}(Y) = \mathbf{R}_{\mathsf{PCI\text{-}Any},\mathsf{inp}_2}(Y) = 1\big\}$$

Simulating Honest Party's Inputs: The simulator $\mathcal{S}$ now simulates a dummy private and public input on behalf of the honest party P_2 for the rest of the protocol as follows:

Simulated private inputs for P_2: $\overline{\mathsf{inp}}_{2,1} = [(\overline{Y}_{2,\ell}, \overline{s}_{2,\ell}^{-1}, \mathsf{m}_{2,\ell})]_{\ell\in[1,N_2]}$, where:

- Each $\overline{Y}_{2,\ell}$ is sampled uniformly at random from $\mathcal{G}$ and input to the Input-G sub-functionality in $\mathcal{F}[\mathcal{G}]$.
- Each $\overline{s}_{2,\ell}^{-1}$ is sampled uniformly at random from Z_p and input to the Input-F sub-functionality in $\mathcal{F}[F_p]$.
- Each $\mathsf{m}_{2,\ell}$ is provided to $\mathcal{S}$ by $\mathcal{F}_{\mathsf{PCI}}$(PCI-Any-DC).

Simulated public inputs for P_2: $\overline{\mathsf{inp}}_{2,2} = [(\overline{r}_{2,\ell}, \overline{R}_{2,\ell}, \mathsf{m}_{2,\ell})]_{\ell\in[1,N_2]}$, where:

- Each $\overline{R}_{2,\ell}$ is sampled uniformly at random from $\mathcal{G}$.

- Each $\overline{r}_{2,\ell}$ is set to be the x-coordinate of $\overline{R}_{2,\ell}$.
- Each $\mathsf{m}_{2,\ell}$ is provided to $\mathcal{S}$ by $\mathcal{F}_{\mathsf{PCI}}(\mathsf{PCI\text{-}Any\text{-}DC})$.

Proceeding with the Protocol: The simulator now proceeds exactly as in the real protocol described in Algorithm 6. We note here that for each $(\ell, \ell') \in [N_1, N_2]$, prior to the output stage in Line 25 of Algorithm 6, the entire computation of the protocol is local. Thus, the environment's view, up to this point, will not leak whether inputs used by honest players' are dummy inputs or the ones the environment provided. Note that in the meantime, the simulator $\mathcal{S}$ can query the respective sub-functionalities from $\mathcal{F}[\mathcal{G}]$ (which includes the sub-functionalities from $\mathcal{F}[F_p]$).

Handling Openings of $C''_{\ell,\ell'}$: We note that for each $(\ell, \ell') \in [N_1, N_2]$, Line 25 of Algorithm 6 involves openings that reveal $C''_{\ell,\ell'}$ values that are either $0_{\mathcal{G}}$ (the point of infinity, which is the additive identity of the group of EC points $\mathcal{G}$) or a uniformly random element in the group of EC points $\mathcal{G}$. We note that $\mathcal{S}$ knows precisely which (ℓ, ℓ') tuples result in the opening of a $C''_{\ell,\ell'}$ value that is equal to $0_{\mathcal{G}}$: this corresponds to an intersecting public key Y which $\mathcal{S}$ can figure out deterministically given $\mathsf{out}_{\mathsf{PCI\text{-}Any\text{-}DC}}(\mathsf{inp}_1, \mathsf{inp}_2)$. $\mathcal{S}$ now proceeds as follows:

- If (ℓ, ℓ') tuples result in the opening of a $C''_{\ell,\ell'}$ value that is equal to $0_{\mathcal{G}}$: Suppose the current execution using the dummy inputs for the honest party P_2 leads to a value $C''_{\ell,\ell'} = Q'$ for some EC point $Q' \in \mathcal{G}$. $\mathcal{S}$ modifies the simulated share of $C''_{\ell,\ell'}$ corresponding to the honest party P_2 by dividing (i.e., subtracting the EC point) Q' from it locally, and modifies the MAC value by dividing (i.e. subtracting the EC point) $Q^{r\alpha}$ from the original MAC value. This is possible since $\mathcal{S}$ knows the MAC key α.
- If (ℓ, ℓ') tuples result in the opening of a $C''_{\ell,\ell'}$ value that is a uniformly random element in the group of EC points $\mathcal{G}$: in this case, $\mathcal{S}$ samples $r \leftarrow Z_p$, randomizes the simulated share of $C''_{\ell,\ell'}$ corresponding to the honest party P_2 by multiplying (i.e. adding the EC point) Q^r to it locally, and modifies the MAC value by multiplying (i.e., adding the EC point) $Q^{r\alpha}$ to the original MAC value.

Both of the above steps are possible since $\mathcal{S}$ knows the MAC key α.

Handling Openings of $Y_{1,\ell}$ values: Finally, Line 26 of Algorithm 6 involves openings that reveal public keys $Y_{1,\ell}$. Note that here, it suffices for the simulator $\mathcal{S}$ to proceed exactly as in the real protocol, since the public keys in the input of the corrupted party P_2 are available to the simulator $\mathcal{S}$ in the clear, and were shared by $\mathcal{S}$ exactly as in the real protocol.

It is easy to see that the view of $\mathcal{Z}$ is identical to that in the $\mathcal{F}[\mathcal{G}]$-hybrid model. Hence, assuming a secure SPDZ-based instantiation of the $\mathcal{F}[\mathcal{G}]$-hybrid model using our proposed MPC framework for generic group operations, our ECDSA-based PCI-Any-DC protocol securely emulates $\mathcal{F}_{\mathsf{PCI}}(\text{PCI-Any-DC})$. This concludes the proof of Theorem 4.

B.2 Proof of Security of BLS-based PCI-All

In this section, we prove the SUC security of our PCI-All protocol based on BLS. More formally, we prove Theorem 5 by constructing a PPT simulator $\mathcal{S}$ such that no PPT environment $\mathcal{Z}$, who corrupts one of the parties (say P_1 without loss of generality) and chooses the input for P_1, can distinguish with significant probability, a view obtained by running our proposed PCI-Any-DC protocol for ECDSA signatures in Algorithm 7 between a PPT adversary $\mathcal{A}$ and honest party (say P_2 without loss of generality), and a simulated execution of the protocol between $\mathcal{S}$ and $\mathcal{F}_{\mathsf{PCI}}(\text{PCI-All})$ (for the two-party setting). The environment $\mathcal{Z}$'s view consists of the intermediate messages sent and received by the adversary $\mathcal{A}$, the input he chose for the honest party P_2, along with output of P_2.

We now describe the construction of the simulator $\mathcal{S}$, which proceeds as follows:

Input Phase: The simulator internally runs the real-world adversary $\mathcal{A}$ to obtain the private and public inputs for the corrupt party P_1. Let the inputs be as follows.

Private inputs for P_1: aggregated tuples $\overline{\mathsf{inp}}_{1,1} = \left[\left(Y_{1,\ell}, \overline{\sigma}_{1,\ell}, \overline{M}_1\right)\right]_{\ell\in[1,N_{1,1}]}$ and the set of preempted pairings $\{z_{1,\ell} = e(\overline{M}_2, Y_{1,\ell})\}_{\ell\in[1,N_{1,1}]}$, where $\overline{\sigma}_{i,\ell} = \prod_{\ell_2\in[1,N_{i,2}]} \sigma_{i,\ell,\ell_2}$ and $\overline{M}_i = \prod_{\ell\in[1,N_{i,2}]} H\left(\mathsf{m}_{i,\ell}\right)$. Each $Y_{1,\ell}$ is secret-shared as $[Y_{1,\ell}]_{\mathcal{G}_2}$, each $\overline{\sigma}_{1,\ell}$ is secret-shared as $[\overline{\sigma}_{1,\ell}]_{\mathcal{G}_1}$, and each $z_{1,\ell}$ is secret-shared as $[z_{1,\ell}]_{\mathcal{G}_T}$.

Public inputs for P_1: $\mathsf{inp}_{1,2} = \{\mathsf{m}_{1,\ell_2}\}_{\ell_2\in[1,N_{1,2}]}$.

Invoking $\mathcal{F}_{\mathsf{PCI}}(\mathsf{PCI\text{-}All})$: The simulator $\mathcal{S}$ now invokes the ideal functionality $\mathcal{F}_{\mathsf{PCI}}(\mathsf{PCI\text{-}All})$ using the inputs of the corrupt party P_1 (the input of the honest party P_2 is directly provided to $\mathcal{F}_{\mathsf{PCI}}(\mathsf{PCI\text{-}All})$ by the environment $\mathcal{Z}$) to receive $(\mathsf{out}_{\mathsf{PCI\text{-}All}}(\mathsf{inp}_1, \mathsf{inp}_2), N_2)$, where N_2 is the number of inputs for the honest party P_2, and

$$\mathsf{out}_{\mathsf{PCI\text{-}All}}(\mathsf{inp}_1, \mathsf{inp}_2) = \big\{Y \in \mathsf{id}(\mathsf{inp}_{1,1}) \cap \mathsf{id}(\mathsf{inp}_{2,1}) : \mathbf{R}_{\mathsf{PCI\text{-}All},\mathsf{inp}_1}(Y) = \mathbf{R}_{\mathsf{PCI\text{-}All},\mathsf{inp}_2}(Y) = 1\big\}$$

Simulating Honest Party's Inputs: The simulator $\mathcal{S}$ now simulates a dummy private and public input on behalf of the honest party P_2 for the rest of the protocol as follows:

Simulated private inputs for P_2: $\overline{\overline{\mathsf{inp}}}_{2,1} = \big[\big(\overline{Y_{2,\ell}}, \overline{\overline{\sigma}}_{2,\ell}, \overline{M}_2\big)\big]_{\ell\in[1,N_{2,1}]}$ and the set of pre-empted pairings $\{\overline{z_{2,\ell}} = e(\overline{M}_1, \overline{Y_{2,\ell}})\}_{\ell\in[1,N_{2,1}]}$, where:

- Each $\overline{Y_{2,\ell}}$ is sampled uniformly at random from $\mathcal{G}_2$ and input to the Input-G sub-functionality in $\mathcal{F}[\mathcal{G}]$ instantiated for $\mathcal{G}_2$.
- Each $\overline{\overline{\sigma}}_{2,\ell}$ is sampled uniformly at random from $\mathcal{G}_1$ and input to the Input-G sub-functionality in $\mathcal{F}[\mathcal{G}]$ instantiated for $\mathcal{G}_1$.
- Each preempted pairing $\overline{z_{2,\ell}}$ is computed as $e(\overline{M}_1, \overline{Y_{2,\ell}})$, where $\overline{M}_1 = \prod_{\ell\in[1,N_{2,2}]} H\left(\mathsf{m}_{2,\ell}\right)$ is the aggregated hashed-claim corresponding to the corrupt party P_1.
- Each $\mathsf{m}_{2,\ell}$ in the simulated public inputs for P_2 is provided to $\mathcal{S}$ by $\mathcal{F}_{\mathsf{PCI}}(\mathsf{PCI\text{-}Any\text{-}DC})$, from which the simulator $\mathcal{S}$ computes $\overline{M}_2 = \prod_{\ell\in[1,N_{2,2}]} H\left(\mathsf{m}_{2,\ell}\right)$.

Simulated public inputs for P_2: $\overline{\mathsf{inp}}_{2,2} = \{\mathsf{m}_{2,\ell})\}_{\ell\in[1,N_{2,2}]}$, where:

- Each $\mathsf{m}_{2,\ell}$ is provided to $\mathcal{S}$ by $\mathcal{F}_{\mathsf{PCI}}(\mathsf{PCI\text{-}All})$.

Proceeding with the Protocol: The simulator now proceeds exactly as in the real protocol described in Algorithm 7. We note here that for each $(\ell, \ell') \in [N_{1,1}, N_{2,1}]$, prior to the output stage in Line 18 of Algorithm 7, the entire computation of the protocol is local. Thus, the environment's view, up to this point, will not leak whether inputs used by honest players' are dummy inputs or the ones the environment provided. Note that in the meantime,

the simulator $\mathcal{S}$ can query the respective sub-functionalities from $\mathcal{F}[\mathsf{Pair}]$ (which includes the sub-functionalities from $\mathcal{F}[F_p]$ and $\mathcal{F}[\mathcal{G}]$, initialized appropriately for $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{G}_T$).

Handling Openings of $c'_{\ell,\ell'}$: We note that for each $(\ell, \ell') \in [N_1, N_2]$, Line 18 of Algorithm 7 involves openings that reveal $c'_{\ell,\ell'}$ values that are either $1_{\mathcal{G}}$ (identity element of the group $\mathcal{G}_T$) or a uniformly random element in $\mathcal{G}_T$. We note that $\mathcal{S}$ knows precisely which (ℓ, ℓ') tuples result in the opening of a $c'_{\ell,\ell'}$ value that is equal to $1_{\mathcal{G}_T}$: this corresponds to an intersecting public key Y which $\mathcal{S}$ can figure out deterministically given $\mathsf{out}_{\mathsf{PCI\text{-}All}}(\mathsf{inp}_1, \mathsf{inp}_2)$. $\mathcal{S}$ now proceeds as follows:

- If (ℓ, ℓ') tuples result in the opening of a $c'_{\ell,\ell'}$ value that is equal to $1_{\mathcal{G}_T}$: Suppose the current execution using the dummy inputs for the honest party P_2 leads to a value $c'_{\ell,\ell'} = h_T$ for some element $h_T \in \mathcal{G}_T$. $\mathcal{S}$ modifies the simulated share of $c'_{\ell,\ell'}$ corresponding to the honest party P_2 by dividing h_T from it locally, and modifies the MAC value by dividing h_T^{α} from the original MAC value. This is possible since $\mathcal{S}$ knows the MAC key α.
- If (ℓ, ℓ') tuples result in the opening of a $c'_{\ell,\ell'}$ value that is a uniformly random element in the group $\mathcal{G}_T$: $\mathcal{S}$ samples $r \leftarrow Z_p$, randomizes the simulated share of $c'_{\ell,\ell'}$ corresponding to the honest party P_2 by multiplying g_T^r to it locally, and modifies the MAC value by multiplying (i.e., adding the EC point) $g_T^{r\alpha}$ to the original MAC value.

Both of the above steps are possible since $\mathcal{S}$ knows the MAC key α.

Handling Openings of $Y_{1,\ell}$ values: Finally, Line 20 of Algorithm 7 involves openings that reveal public keys $Y_{1,\ell}$. Note that here, it suffices for the simulator $\mathcal{S}$ to proceed exactly as in the real protocol, since the public keys in the input of the corrupted party P_2 are available to the simulator $\mathcal{S}$ in the clear, and were shared by $\mathcal{S}$ exactly as in the real protocol.

It is easy to see that the view of $\mathcal{Z}$ is identical to that in the $\mathcal{F}[\mathcal{G}]$-hybrid model. hence, assuming a secure SPDZ-based instantiation of the $\mathcal{F}[\mathcal{G}]$-hybrid model using our proposed MPC framework for generic group operations, our ECDSA-based PCI-All protocol securely emulates $\mathcal{F}_{\mathsf{PCI}}(\mathsf{PCI\text{-}All})$. This concludes the proof of Theorem 5.

Appendix C

Extensions to Multi-Party PCI

In this section, we discuss extensions of our proposed PCI protocols, namely the PCI-Any-DC protocol based on ECDSA signatures in Chapter 6.4 and the PCI-All protocol based on BLS signatures in Chapter 6.5, to the n-party setting.

C.1 Extending ECDSA PCI-Any-DC to n-Party PCI-Any-DC

We first discuss how to extend our ECDSA-based PCI-Any-DC protocol (Algorithm 6, Chapter 6.4) to the n-party setting. We divide the discussion into three phases - the input phase, the certificate validation phase, and the certifier matching phase.

Input Phase. To begin with, we can directly replicate the input phase of our two-party PCI-Any-DC protocol in the n-party setting. Concretely, as in the original two-party protocol (Lines 1-6 of Algorithm 6), each of the n participating parties inputs tuples of (identifier, certificate, claim) (with the same modifications/optimizations as described in Section 6.4) as its private input, and the corresponding claim and (r, R) for each tuple as its public input.

Certificate Validation Phase. We again directly replicate the certificate validation phase of our two-party PCI-Any-DC protocol in the n-party setting. Concretely, as in the original two-party protocol (Lines 7-11 and 12-16 of Algorithm 6), the protocol validates the sig-

natures for each of the n participating parties. The validation proceeds in parallel for each of the parties.

Certifier Matching Phase. This is where the n-party version of our protocol involves a non-trivial extension of the original two-party protocol (Lines 19-26 of Algorithm 6). Note that a trivial extension of the protocol would involve n-nested loops (one for each participating party), thereby yielding a protocol with computational and communication complexity $O\left(\prod_{i\in[1,n]} |\mathsf{inp}_i|\right)$, which is clearly undesirable (this is, in fact, approximately $O(c^n)$ times more expensive that the two-party protocol, assuming a minimum input size of $O(c)$ per party, i.e., the overheads grow exponentially in the number of parties). It turns out, however, that this trivial extension essentially matches the certifier public keys across "all" parties in a pair-wise fashion, which is clearly unnecessary. In fact, without loss of generality, it suffices to simply match each certifier public key input by party P_1 with the corresponding certifier public keys across parties $P_2, \ldots, P_n$. This reduces the required number of checks from $O\left(\prod_{i\in[1,n]} |\mathsf{inp}_i|\right)$ to $O\left(|\mathsf{inp}_1| \cdot \sum_{i\in[2,n]} |\mathsf{inp}_i|\right)$, which is significantly more efficient (in fact, we now incur approximately $O(n)$ times more checks that the two-party protocol). In fact, by choosing P_1 to be the party with the smallest input size, we can optimize the overheads even further.

Concretely, in the certifier matching phase of the n-party version of our PCI-Any-DC protocol, we run a two-nested set of loops – an outer loop for party P_1 and $(n-1)$ inner loops for parties P_2 through P_n. Each inner loop performs essentially the same computation as in Lines 22-23 of Algorithm 6, except that we now accumulate the outcomes of each of these *intra-loop* computations into a global check variable C maintained across all of these inner loops. This two-layered accumulation step (requiring an intra-loop accumulation followed by an inter-loop accumulation), however, reduces to computing a Boolean formula in the conjunctive normal form (CNF). This is because we require a multiplication operation for the intra-loop accumulation (in order to check whether there is at least one matching certifier identity between P_1 and P_j) and an addition operation across the loops to compute the final intersection. Computing such a CNF formula necessitates using field elements for accumulation, which is not immediate from Algorithm 6, where the variable $C''_{\ell,\ell'}$ is actually a group element and is amenable to field operations. Hence, we need to hash $C''_{\ell,\ell'}$ into a corresponding field element $c_{\ell,\ell'}$, which incurs the additional cost of computing

$O\left(|\mathsf{inp}_1| \cdot \sum_{i\in[2,n]} | \ inp_i|\right)$ instances of a collision-resistant hash function inside the MPC protocol[1]. Along the way, we also incur some other additional costs, such as sampling additional randomnesses for each inner loop and an n-party opening protocol for the final accumulation variable.

Assuming $O(n)$ overheads for each such operation, the overall computational and communication complexities scales as $O\left(n \cdot |\mathsf{inp}_1| \cdot \sum_{i\in[2,n]} |\mathsf{inp}_i|\right)$, which is approximately $O(n^2)$ times more expensive than the two-party protocol for realistic input sizes. In particular, we expect that for large values of n (i.e. for multi-party PCI involving a large number of parties), our proposed solution will be significantly more efficient that the naïve extension of the two-party solution outlined above (even taking into the hidden constants due to the additional overheads incurred by our proposed solution). We leave it an as open question to investigate additional optimizations (or alternative solutions) that could allow reducing the overheads even further.

Extension to PCI-Any**.** Finally, analogous to the two-party setting, one can naturally upgrade the above n-party PCI-Any-DC protocol to an n-party PCI-Any protocol that additionally guarantees privacy of the input claims for each party by treating the claims as part of the private input. More concretely, the claims would be secret-shared across all of the n participating parties instead of being publicly available, and all operations on the input claims would have to be performed inside the MPC protocol. As discussed in Section 6.4, this would incur additional costs of performing certain operations (such as field inversions, extraction of point coordinates, and hashing of claims) inside the MPC protocol. We again leave it as an interesting future direction to investigate optimization strategies that would allow performing the above operations efficiently (i.e., outside the MPC protocol) in the n-party setting.

C.2 Extending BLS PCI-All to n-Party PCI-All

We can similarly extend our BLS-based PCI-All protocol (Algorithm 7, Chapter 6.5) to the n-party setting. In particular, as in the case of our PCI-Any-DC protocol, we directly repli-

[1]We can use an MPC-friendly hash here [184] to optimize the associated overheads

cate the input phase (and its associated pre-processing: Lines 1-6 of Algorithm 7) as well as the certificate validation phase (Lines 7-10 of Algorithm 7) of our two-party PCI-Any-DC protocol in the n-party setting, with these phases executed in parallel for each of the n participating parties.

Finally, for certifier matching, we again exploit the fact it suffices to simply match each certifier public key input by party P_1 with the corresponding certifier public keys across parties $P_2, \ldots, P_n$ to design an n-party certifier matching phase with $O\left(|\mathsf{inp}_1| \cdot \sum_{i \in [2,n]} |\mathsf{inp}_i|\right)$ computational and communication overhead. Concretely, in the certifier matching phase of the n-party version of our PCI-All protocol, we run a two-nested set of loops – an outer loop for party P_1 and $(n-1)$ inner loops for parties P_2 through P_n. Each inner loop performs essentially the same computation as in Lines 14-16 of Algorithm 6, except that we again accumulate all of these computations into a global variable C maintained across all of these inner loops. Along the way, we similarly incur additional costs (such as hashing the accumulation variables into field elements, sampling additional randomnesses for each inner loop and running an n-party opening protocol for the final global variable), but once again, assuming $O(n)$ overheads for each such operation, the overall computational and communication complexities scale as $O\left(n \cdot |\mathsf{inp}_1| \cdot \sum_{i \in [2,n]} |\mathsf{inp}_i|\right)$. This is again approximately $O(n^2)$ times more expensive than the two-party protocol for realistic input sizes, and is expected to be significantly more efficient that a naïve extension of the two-party solution. We again leave it an as open question to investigate additional optimizations (or alternative solutions) that could allow reducing the overheads even further.