Auditable Private Ledgers

by

Willy R. Vasquez

B.S., Massachusetts Institute of Technology (2015)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2017

Author . . . .
Department of Electrical Engineering and Computer Science
August 18, 2017

Certified by . . . .
Dr. Neha Narula
Director of Digital Currency Initiative, Media Lab
Thesis Supervisor

Accepted by . . . .
Dr. Christopher J. Terman
Chairman, Masters of Engineering Thesis Committee

Auditable Private Ledgers

by

Willy R. Vasquez

Submitted to the Department of Electrical Engineering and Computer Science
on August 18, 2017, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

This thesis describes the design and implementation of Auditable Private Ledgers (APL), a privacy solution for distributed ledgers that lets third parties audit private ledger data. With the use of permissioned blockchains, zero-knowledge proofs, and additively homomorphic commitments, we are able to provide a balance between privacy and regulation. For this work, I implemented the cryptographic tools in Go, helped develop zero-knowledge proofs to provide data authenticity and integrity, and performed an evaluation of this system to measure its performance. Our work shows that the system is reasonable to run between a small number of participants, and that we can efficiently run private sums without revealing fine-grained inputs.

Thesis Supervisor: Dr. Neha Narula
Title: Director of Digital Currency Initiative, Media Lab

Acknowledgments

There are many special people that have helped me get to where I am today that I have to give a shout out to.

My parents, Andres and Maria, who taught me the importance of responsibility and being humble, and The Child who will one day fly higher than I have.

My advisor, Neha, whose patience and friendship has been quite invaluable in this endeavor. It was challenging for me at times, but her patience and advising helped me achieve my goals. Thank you for believing in me and teaching me how to be a successful researcher.

My unofficial other advisor, Madars, whose cheerfulness and Taylor Swift references made doing research a blast.

The folks at the DCI who provided stimulating conversations from cryptocurrencies to cricket over tea. It's been an incredible year and it's been an honor to work with y'all.

A special shout out to Jeremy and Brian for introducing me to the DCI, and allowing me to have this awesome opportunity.

The undergraduate Latino community at MIT: LUChA, MAES, and Spanish House. They took this lost freshman and made me the leader I am today. I hope to continue supporting them in an alumni role, and am looking forward to see members succeed.

My brother Alan, whom with I've spent many late nights this past year knocking out work and playing CoD.

Last but not least, my wife Ana, who has believed in me from day one and continues to push me further than I ever thought I could go. I love you.

Contents

List of Figures

Chapter 1

Introduction

Auditability provides a guarantee that a system is behaving as expected. From financial regulation to measurement of systemic risk, auditability provides accountability in systems. Lack of auditability or inaccurate results from auditing can have devastating effects. For example, the 2008 financial crisis was caused, in part, by a lack of insight by regulators and investors into the operations of financial institutions.

In this work, we build upon applications of blockchains with privacy and auditability as a central tenant. We develop a system called Auditable Private Ledgers where blockchain transactions hide who is transacting and the amount transacted, while letting auditors get an understanding of the system as a whole by computing on the hidden transaction data.

Traditionally, auditability has been solved by the use of trusted third party auditors [34]. Auditing companies such as Deloitte, PricewaterhouseCoopers, Ernst and Young, and KPMG (known as the "Big Four") together audit 99% of the companies in the FTSE 100 [21], the hundred companies on the London Stock Exchange with the highest capitalization. The U.S. Government Accountability Office uses a set of standards known as the Yellow Book to audit government entities and government-funded organizations [6]. Financial institutions and exchanges are monitored by federal and state government agencies such as the Office of the Comptroller of the Currency (OCC), the Federal Deposit Insurance Corporation (FDIC), the Securities and Exchange Commission (SEC), and other agencies with specialized focuses in order to ensure a well managed systemic risk and the prevention of deceptive or manipulative trading practices [34].

Unfortunately, this type of auditing is a laborious and time-consuming process, meaning that regulators and investors cannot get access to real-time information about the status of financial institutions. This means auditing information is usually out of date in a world where investments change at a fast pace. In addition, the auditing institutions sometimes make mistakes. The most well-known example of this is the collapse of Arthur Anderson, which failed to catch Enron's $100 billion accounting fraud. In 2008, Ernst and Young failed to catch Lehman Brother's strange accounting with repo 105s, which made it look like Lehman was better capitalized than it really was. From this grew the Dodd-Frank Act (DFA), a 2010 wall-street reform in response to the 2008 financial crisis, that required a transformation of banking regulation and more transparency across financial institutions. The DFA established the Office of Financial Research in order to promote financial stability and collect data from institutions to measure systemic risk, and it produces a quarterly report on U.S. systemic risk [1].

A challenge with financial data is the requirement of privacy of trading strategy. Revealing asset trading strategy could let a competitor capitalize and influence a market to cause significant financial harm to a bank. For example, in 2012, news sources made public JP Morgan's large position in synthetic credit default swaps (known as the London Whale), and competitors were able to take on competing swap positions so that when the swaps suffered losses, JP Morgan's losses were much higher than if their position was kept private [25]. Because banks submit periodic reports that need time for analysis, it is difficult to get a global real-time view of the financial system. By aggregating the data of actors in the system, auditors can better predict failures and growths of the system, and could help prevent fraud as it occurs.

Recently, financial institutions have formed consortia [40] to investigate the use of a different architecture for securities settlement, inspired by blockchain technology. The blockchain is the data structure behind Bitcoin, an open, distributed cryptocurrency. Bitcoin's success has motivated institutions to consider upgrading their technical infrastructure by using permissioned blockchains, often maintained by participants with a consensus protocol. There are many strong players in this area that are making an impact such as IBM's Hyperledger [2], R3's Corda [30], Tendermint [42], Kadena [3], and JP Morgan's Quorum [5]. These players provide a spectrum of privacy solutions, from privacy via segmentation to privacy through cryptography, but they all require a trusted third party to audit ledger content.

With financial institutions participating in these ledgers, auditing becomes a real-time process for quick responses to issues that may arise.

1.1 Contributions

We developed a practical system we call Auditable Private Ledgers (APL) that supports auditing queries for financial institutions while keeping individual transactions private between participants. Using homomorphic commitment schemes, zero-knowledge proofs, and permissioned blockchains we can run any affine or linear function on committed data while achieving efficient transaction throughput.

The contributions of thesis are the implementation of the cryptographic components of APL in Go, the evaluation of APL on a single host, and the design and development of the disjunctive proof applied to proof of assets.

This thesis builds up to our design by first providing related work and background. It then describes the design, the implementation, and the evaluation, and concludes with requirements for use cases outside of financial auditing.

Chapter 2

Related Work

Auditable Private Ledgers (APL) focuses on providing auditability, or the measurement of the state of a system, to private data stored on distributed ledgers without relying on a trusted third party.

Here we present related work in the areas of bank ledger privacy and private auditing of financial institutions. Privacy in this context refers to hiding the amount that is being transacted between entities, as well as hiding who the senders and receivers of each transaction are in the entire system, effectively concealing the complete transaction graph.

2.1 Bank Ledger Privacy

Financial institutions such as banks and exchanges are exploring blockchains for improving transaction speed and verification.

One such exploration is the R3 Consortium [40] which is composed of more than 70 banks that experiment with blockchain technology. R3's system, Corda [30], provides confidentiality, but relies on trusted third parties called notaries to perform transaction verification, achieve consensus, and provide privacy to transactions. Our system similarly is set in a permissioned system and lets banks add items to a distributed ledger, but differs by not inherently relying on a trusted third party to verify transactions, achieve consensus, or provide privacy.

The Initiative for Cryptocurrencies and Contracts (IC3) [8] developed Solidus [19], a system that uses Oblivious RAM to hide the transaction graph and transaction amount between bank customers. While their construction also explores the use case of private transactions, they only provide auditability by revealing all of keys used in the system to an auditor, and do not hide the transaction graph between banks.

Privacy on blockchains deals with hiding either one, some, or all of transaction amounts, the transaction graph, or transaction asset types.

Zcash [7] is a cryptocurrency that hides transaction amounts and the transaction graph using zero-knowledge proofs and other less conservative assumptions. Zcash relies on a trusted setup for its zero-knowledge succinct arguments of knowledge (zk-SNARKs) which, if this trusted setup is tampered with, may result in the invalidation of the currency. Specifically, since Zcash assets are blinded, there is no exact way to measure the total number of assets in Zcash. Thus with a tampered trusted setup, an attacker may be able to issue new money without detection of the network[15]. Zcash does not inherently provide auditability, but its zk-SNARK construction can be extended to enforce policies that give auditability properties [29].

Chain's Confidential Assets [20] is a system for managing multiple assets on a single ledger, and assuring privacy of transaction amount and transaction types. Confidential Assets has the capability to privately issue out new asset types, but does not address the issue of transaction graph privacy on its own; it can be combined with other techniques to provide this property. While we both use tools such as range proofs and Pedersen commitments, Chain does not explore auditability of committed data, and focuses on providing privacy. Confidential Assets has the capability to handle multiple assets, while we do not at the time of writing.

Similar to Chain's construction, Blockstream's Confidential Transactions/Assets [39] provides transaction amount and type privacy to multiple assets on a single ledger, including in issuances, but does not provide any features for auditing of data, nor attempts to hide transaction participants.

Digital Asset Holdings (DAH) has developed a system called the Global Synchronization Log (GSL) which is a distributed ledger that achieves privacy by creating segregated ledgers that are synchronized by authorized entities when necessary [14]. Instead of relying on sophisticated cryptographic tools for privacy, they run the segregated ledgers with parties on a need-to-know basis, and only share fingerprints of the data on the global ledger. The challenge for an auditor in this case is determining all the required ledgers they need to audit, instead of having the required data in a single location.

2.2 Private Auditing of Financial Institutions

Two projects that have the similar goal of privately auditing financial institutions are Provisions [22], and Abbe et al.'s work [12] on multi-party computation (MPC) for financial risk.

Provisions is a system for proving cryptocurrency exchange solvency [22]. They provide auditability for exchanges to prove that they are solvent without revealing the amount that they have. Exchanges provide information to third parties to verify their committed data is correct, but auditors cannot perform computations on exchange data to perform further analysis. Provisions also suffers from the fact that exchanges can potentially share keys and double count the same assets for different exchanges.

In [12], Abbe et al. provide auditability for all banks by using MPC to run functions on banks' private data. Since transaction verification and transaction logging occur separately, banks may not provide all their balance sheet information when running a computation with other banks, or may misrepresent the transactions. By combining both transaction verification and logging into one system, we are able to prevent banks from excluding information and maintain a real-time view of transactions.

Chapter 3

Background

APL relies on three technical tools: permissioned blockchains, non-interactive zero-knowledge proof systems, and additively homomorphic commitment schemes. From these components, we look into how to run financial auditing functions to measure systemic risk.

3.1 Elliptic Curve Cryptography

Elliptic curve cryptography (ECC) is the use of elliptic curves to provide cryptographic guarantees. Predominantly they are used in asymmetric cryptography settings where security is guaranteed by the difficulty of the elliptic curve discrete logarithm problem (ECDLP).

Definition 1 (ECDLP) *Given an elliptic curve group $\mathbb{E}(\mathbb{Z}_p)$ with base-point G, the* ***elliptic curve discrete logarithm problem (ECDLP)*** *is the problem of recovering the scalar x given the group elements xG and G.*

From the ECDLP, we can describe the elliptic curve decisional Diffie-Hellman (ECDDH) problem and the elliptic curve computational Diffie-Helmman (ECCDH) problem.

Definition 2 (ECDDH) *Given points $A = aG$, $B = bG$, and $C = cG$ from the elliptic curve group $\mathbb{E}(\mathbb{Z}_p)$ with base-point G, the* ***elliptic curve decisional Diffie-***

Hellman problem (ECDDH) *asks whether an adversary is able to correctly determine if C is chosen at random or is a combination of A and B.*

Definition 3 (ECCDH) *Given points $A = aG$ and $B = bG$ from the elliptic curve group $\mathbb{E}(\mathbb{Z}_p)$ with base-point G, the* ***elliptic curve computational Diffie-Hellman problem (ECCDH)*** *asks whether an adversary is able to compute $C = abG$ given A and B.*

Given the ECDDH and ECCDH, we can now construct cryptographic systems that rely on the difficulty of these problems. In particular, we rely on the security of the ECDLP to construct our zero-knowledge proofs and Pedersen commitments.

3.2 Permissioned Blockchain

A **blockchain** is a decentralized, tamper-resistant, append-only ledger that relies on a consensus protocol to control what is added to it. Blockchains organize data in the ledger as a series of blocks that contain transaction information, and have a reference to a preceding block, chaining them together. This reference is what gives blockchains the append-only properties: a change in a block in the middle of the chain will require a significant amount of computation or coordination to update the following blocks, which would violate the consensus protocol.

Blockchains can have two different access control structures: permissionless or permissioned. In a permissionless blockchain anyone can participate in the consensus protocol of the ledger and propose data to add to it, while in a permissioned blockchain only a set of known entities can propose data to be added.

Depending on the access structure, there are a variety of consensus protocols that can be used on the blockchain. In a permissionless setting, this may require proof systems, where participants show they have exerted some computational effort [35], allocated a certain amount of space on a hard drive [36], have a vested interest in the success of the blockchain [23], or satisfied some conditions based on the randomness of the system [33]. In a permissioned setting, since the set of entities is already known, participants can rely either on proof based systems already mentioned, or byzantine-fault tolerant solutions ([18], [31], [42]) that require a stable set of participants in a blockchain.

3.3 Non-Interactive Zero-Knowledge Proof Systems

A **zero-knowledge proof system** is a method for a party to prove that they have knowledge of some secret information to another party without revealing what that information is. All our proofs use elliptic curve groups with base-points G and H and rely on the difficulty of the ECDLP. We use Σ-protocols [24] for constructing Schnorr proofs of knowledge of discrete logarithms [41], equivalence proofs between two different functions, disjunctive proofs of two boolean statements, range proofs on committed values, and consistency proofs using the Maurer technique for homomorphisms [32]. We achieve non-interactivity in protocols by relying on the Fiat-Shamir heuristic [27] which replaces the verifier's chosen randomness with a random oracle.

Figure 3-1 provides the general structure of all zero-knowledge proof systems we use in APL, and is provided here for intuition for later sections. It shows the generic structure of Σ-protocols with the Fiat-Shamir heuristic, which consists of knowing a secret x from a group G with operator $\star$, a cryptographic hash function $\mathcal{H}$, and a group homomorphism $f : G \to H$ that preserves $\star$ in G by operation $\otimes$. Given a secret x, the prover first chooses a random value k and applies f to k to get t. Next the prover hashes the public parameters y and g alongside t in order to produce a random value c, which will be used to produce r, an offset of the initial secret x. Finally, the verifier, given t, c, r, y can verify c is indeed the hash output of the public parameters and t, and that f applied to r is equivalent to homomorphic operation in H applied to t and y^c.

Σ Protocol

$$
\begin{array}{lcl}
\textbf{Prover} & & \textbf{Verifier} \\
\text{knows } x & & \text{knows } y = f(x) \\
k \xleftarrow{\$} G & & \\
t = f(k) & & \\
c = \mathcal{H}(t, y, g) & & \\
r = k \star x^c & & \\
 & \xrightarrow{t, c, r} & \\
 & & \text{check } c \overset{?}{=} \mathcal{H}(t, y, g) \\
 & & \text{check } f(r) = f(k \star x^c) \\
 & & \qquad\qquad = f(k) \otimes f(x^c) \\
 & & \qquad\qquad = f(k) \otimes f(x)^c \\
 & & \qquad\qquad \overset{?}{=} t \otimes y^c
\end{array}
$$

Figure 3-1: Sigma Protocols

3.4 Additively Homomorphic Commitment Schemes

A cryptographic **commitment scheme** lets parties commit to a particular value and later open the commitment to a verifier with an assurance to the verifier that the committed value and the opened value are the same. The role of a commitment scheme is to *hide* the value that is committed to, but also provide a *binding* to the originally committed value. Commitment schemes can either perfectly bind to a value, thus preventing an unbounded-in-resources adversary from cheating, or perfectly hide the value committed to, preventing an unbounded-in-resources adversary from discovering it, but not both. Given one perfect property, the other property must depend on security from a computationally bounded adversary [10].

The Pedersen commitment scheme [37] provides computational binding based on the ECDLP and perfect hiding. In APL, this gives auditors the guarantee that banks cannot lie unless they can solve the ECDLP, while giving banks the guarantee that their transacted data can never be discovered by an unbounded-in-resources adversary.

Definition 4 (Pedersen Commitment Scheme) *Given G, H generators, where the discrete log between G and H is unknown, the Pedersen commitment scheme consists of two algorithms that provide perfect hiding and computational binding:*

- $\mathsf{Commit}(v) \rightarrow (\mathcal{C}, r)$: *sample* $r \overset{\$}{\leftarrow} \mathbb{Z}_p$ *and return commitment* $\mathcal{C} = vG + rH$ *and randomness* r*.*

- $\mathsf{Open}(\mathcal{C}, v, r) \rightarrow 0, 1$: *given commitment* $\mathcal{C}$*, value* v*, and* r*, return 1 iff* $C \overset{?}{=} vG + rH$ *else return 0.*

The Pedersen commitment scheme is additively homomorphic and provides the ability to perform affine or linear functions over committed data. The following two definitions detail how affine and linear functions work in the Pedersen commitment scheme.

Definition 5 (Affine Functions with Pedersen Commitment Scheme) *Given a commitment of* v *with randomness* r*,* $\mathcal{C} = vG + rH$*, and scalars* $\delta, c \in \mathbb{Z}_p$*, we multiply the commitment with* c *and add by* δG *to get a commitment of* $cx + \delta$*:*

$$c\mathcal{C} + \delta G = c(vG + rH) + \delta G = cvG + crH + \delta G = (cx + \delta)G + (cr)H.$$

Definition 6 (Linear Functions with Pedersen Commitment Scheme) *Given a commitment of two values* v *and* w*, we add the two commitments to get a commitment for* $x + y$*:*

$$vG + rH + wG + r'H = (v + w)G + (r + r')H$$

3.5 Financial Auditing

Auditing of financial institutions is performed by authorized private and government entities to ensure that systemic risk in financial systems is at a manageable level, regulations are correctly enforced, and operations are managed effectively [16]. Financial systemic risk auditing is performed by federal and state agencies, depending on the size of the audited institution and the type of asset being audited. Financial auditing can also be used to track fraudulent transactions and funding of criminal behavior [28]. In this thesis, we are interested in financial auditing for measuring systemic risk.

There are two main forms of systemic risk that are measured: contagion between counterparty risk in financial institutions, and contagion in prices of assets held by different institutions [17]. The first deals with the agreements that occur between financial institutions and the size of those agreements, usually referred to as counterparty risk. One example would be estimating the effect of a set of banks failing, and how the counterparty risk these banks held would affect the rest of the network. The second deals with common balance sheet holdings, where the forced sale of illiquid assets by one institution drives prices down and causes other institutions holding the same type of assets to adjust for this price change in their holdings.

There are many ways to model contagion risks, using techniques from graph theory [13], complex physical systems, and statistics [17]. In this work we start with statistical analysis measuring the Herfindahl Index, similar to [11]. The Herfindahl Index provides a measure of competition in a certain industry among a set of firms by calculating the sum-of-squares of the market share of measured firms. Given a total number of shares T amongst N firms each with amount a_i of shares, then the Herfindahl Index H is calculated as:

$$H = \sum_{i=1}^{N} \left(\frac{a_i}{T} \right)^2 .$$

A higher Herfindahl Index means an industry is highly concentrated by a few players, while a lower Herfindahl Index means the industry is competitive.

In our context, we calculate the Herfindahl Index to measure the concentration of a particular asset amongst a group of banks, to correlate with the common balance sheet holdings. A higher Herfindahl Index means a lower possibility of contagion effect from a bank selling the asset in question, thus driving down its price. A lower Herfindahl Index means assets are more uniformly spread and the contagion effect is higher since more banks have the measured asset in their holdings.

Chapter 4

Zero-Knowledge Proofs

Given the building blocks from Section 3, we describe how we use zero-knowledge proofs to provide guarantees about our system. We use specialized proofs that guarantee that computations are done correctly and that banks cannot undetectably lie about transacted values. Our proofs rely on a cryptographic hash function $\mathcal{H}$, and public elliptic curve parameters that all entries can verify: an elliptic curve group $\mathbb{E}(\mathbb{Z}_p)$, group generators G and H, and scalar group modulus p.

We use zero-knowledge proofs with the Fiat-Shamir heuristic [27] to take advantage of the non-interactive proving properties. Non-interactivity makes it so that a single bank can produce a transaction without having to coordinate with the receiving bank, and banks and auditors can verify transactions as they get added to the ledger without having to communicate with each other.

4.1 Proofs of Secret Key

We use the **Generalized Schnorr Proof (GSP)** of knowledge for a discrete logarithm in order to ensure that transactions are valid and that transactions are not forged.

Banks initiating transactions create GSPs to prove that they have sufficient assets in one half of disjunctive proofs, or that they publicly signal an issuance or withdrawal.

Figure 4-1 shows how this works with elliptic curve points. A prover wants to

Discrete Log Σ Protocol

Prover		**Verifier**
knows x		knows $Y = xG$
$k \xleftarrow{\$} \mathbb{Z}_p$		
$T = kG$		
$c = \mathcal{H}(T, Y, G)$		
$r = k + c \cdot x$		
	$\xrightarrow{r, T, c}$	
		check $c \overset{?}{=} \mathcal{H}(T, Y, G)$
		check $rG = (k + c \cdot x)G$
		$= kG + c \cdot xG$
		$\overset{?}{=} T + cY$

Figure 4-1: Generalized Schnorr Proof

convince a verifier that knows $Y = xG$ they know x without revealing x. The prover first chooses a random value k and produces $T = kG$. Given T, Y, and G, the prover uses a cryptographic hash function $\mathcal{H}$ to generate c. The prover computes $r = k + c \cdot x$ and sends to the verifier r, T, and c. The verifier makes sure that c was generated correctly by checking that it is equal to $\mathcal{H}(T, Y, G)$ and then multiplies r by G and compares that to $T + cY$.

4.2 Disjunctive Proofs

With **disjunctive proofs** provers hide what side of a boolean OR statement of knowledge they are proving. This is a crucial element to proving that a bank has a sufficient amount of assets without revealing which bank is transacting.

Since disjunctive proofs only need to be true on one side of the disjunction, the prover simulates the proof for the other side, meaning that constants are chosen so that the statement is vacuously true even if the secret is not known. The simulation only succeeds if the side being proved is in fact true; if a party cannot prove either side,

Disjunctive Σ Protocol

Prover **Verifier**

knows $A = aG$ and $B = bG$

Disjunction (follows one column)

knows a	knows b
$k_1 \xleftarrow{\$} \mathbb{Z}_p$	$k_2 \xleftarrow{\$} \mathbb{Z}_p$
$s_2 \xleftarrow{\$} \mathbb{Z}_p$	$s_1 \xleftarrow{\$} \mathbb{Z}_p$
$c_2 \xleftarrow{\$} \mathbb{Z}_p$	$c_1 \xleftarrow{\$} \mathbb{Z}_p$
$T_1 = k_1 G$	$T_2 = k_2 G$
$T_2 = s_2 G - c_2 B$	$T_1 = s_1 G - c_1 A$
$c = \mathcal{H}(T_1, T_2, G, A, B)$	$c = \mathcal{H}(T_1, T_2, G, A, B)$
$c_1 = c - c_2 \mod p$	$c_2 = c - c_1 \mod p$
$s_1 = k_1 + c_1 \cdot a$	$s_2 = k_2 + c_2 \cdot b$

$$\xrightarrow{T_1, T_2, c, c_1, c_2, s_1, s_2}$$

check $c \overset{?}{=} \mathcal{H}(T_1, T_2, G, A, B)$

check $c \overset{?}{=} c_1 + c_2 \mod p$

check $s_1 G \overset{?}{=} T_1 + c_1 A$

check $s_2 G \overset{?}{=} T_2 + c_2 B$

Figure 4-2: Disjunctive Proofs

both cannot be simulated due to the Fiat-Shamir heuristic preventing predetermined constants from being used.

Figure 4-2 shows how disjunctions are proved. W.L.O.G., a party proving knowledge of a simulates the proof for $B = bG$ as so: they choose random numbers k_1, s_2, and c_2. The prover computes $T_1 = k_1 G$ and $T_2 = s_2 G - c_2 B$ of which the latter will be used to simulate knowledge of $B = bG$. The prover calculates c using $\mathcal{H}$ on public information T_1, T_2, G, A, B and subtracts c_2 from c to calculate c_1. The prover then calculates $s_1 = k_1 + c_1 a$ which is used to prove knowledge of a, and sends $T_{\{1,2\}}, c_{\{1,2\}}$, and $s_{\{1,2\}}$ to the verifier. The verifier makes sure that c is the output of $\mathcal{H}$ on the public information, that $c_1 + c_2 = c$, and that $s_{\{1,2\}}$ applied to G equals $T_{\{1,2\}} + c_{\{1,2\}}\{A, B\}$.

Equivalence Σ Protocol

Prover		**Verifier**
knows x		knows $Y = xG, Z = xH$
$k \xleftarrow{\$} \mathbb{Z}_p$		
$T_1 = kG$		
$T_2 = kH$		
$c = \mathcal{H}(G, H, Z, Y, T_1, T_2)$		
$s = k + c \cdot x$		
	$\xrightarrow{T_1, T_2, s, c}$	
		check $c \stackrel{?}{=} \mathcal{H}(G, H, Z, Y, T_1, T_2)$
		check $sG \stackrel{?}{=} T_1 + cY$
		check $sH \stackrel{?}{=} T_2 + cZ$

Figure 4-3: Equivalence Proofs

4.3 Equivalence Proofs

Equivalence proofs allow a party to prove that the secret values used in two different equations are the same. Given $Y = xG$ and $Z = xH$ the proof certifies that the x's are the same. This is used by banks to prove to auditors that committed values are equal to those on the ledger.

Figure 4-3 shows how to prove equivalence of exponent of two values with different base points. A prover that knows x wants to prove that $Y = xG$ and $Z = xH$ have the same discrete log x. The prover chooses a random k and computes $T_1 = kG$ and $T_2 = kH$ and uses a cryptographic hash function $\mathcal{H}$ on the public parameters G, H, Z, Y, T_1, T_2 to calculate c. The prover computes $s = k + cx$ and sends it to the verifier, which checks if c is the output of $\mathcal{H}$ on the public parameters, and that the same s checks out in $sG = T_1 + cY$ and $sH = T_2 + cZ$.

4.4 Consistency Proofs

Consistency proofs rely on the Maurer proofs for homomorphisms [32] to show consistency in the randomness value used between Pedersen commitments and other elliptic curve group points multiplied by the same value. They are similar to equivalence proofs with the second equation set to a Pedersen commitment. The proof convinces a verifier that the same randomness value is used in the commitment and in the multiplication. The necessity for this consistency proof becomes apparent in Section 5.2.2.

Figure 4-4 demonstrates how to prove consistency between commitments and multiplied group values. The prover that knows the value v and randomness r used in a Pedersen commitment CM as well as the point $PK = xH$ wants to prove that $Y = rPK$ is the same r used in the CM. The prover first chooses two random values k_1 and k_2 and computes $A_1 = k_1G + k_2H$ and $A_2 = k_2PK$. Then using a cryptographic hash function $\mathcal{H}$, the prover computes c from the public parameters $\mathcal{H}(G, H, A_1, A_2, PK, CM, Y)$ and produces $s_1 = k_1 + c \cdot v$ and $s_2 = k_2 + c \cdot r$. The prover sends $A_{\{1,2\}}, c, s_{\{1,2\}}$ to the verifier, which makes sure the hash function output of the public parameters matches c, calculates $s_1G + s_2H$ and compares it to $A_1 + cCM$, and calculates s_2PK and compares it to $A_2 + cY$. If all these checks are passed, then the verifier is assured CM and Y use the same r value.

4.5 Range Proofs

We use Blockstream's Confidential Assets [39] zero-knowledge **range proofs** in APL to prove that a bank's value is within the range $[0, 2^{64})$. This gives verifiers the guarantee that a transacting bank is sending a positive number of assets, has sufficient assets to transact, and are not trying to decrease the assets of another bank. The details of the construction can be found in [39].

Consistency Σ Protocol

Prover		**Verifier**
knows $v, r, PK = xH$		knows $CM = vG + rH$
$k_1 \overset{\$}{\leftarrow} \mathbb{Z}_p$		knows $Y = rPK$
$k_2 \overset{\$}{\leftarrow} \mathbb{Z}_p$		
$A_1 = k_1G + k_2H$		
$A_2 = k_2PK$		
$c = \mathcal{H}(G, H, A_1, A_2, PK, CM, Y)$		
$s_1 = k_1 + c \cdot v$		
$s_2 = k_2 + c \cdot r$		
	$\xrightarrow{A_1, A_2, c, s_1, s_2}$	
		check $c \overset{?}{=} \mathcal{H}(G, H, A_1, A_2, PK, CM, Y)$
		check $s_1G + s_2H \overset{?}{=} A_1 + cCM$
		check $s_2PK \overset{?}{=} A_2 + cY$

Figure 4-4: Consistency Proofs

Chapter 5

Auditable Private Ledgers

5.1 Overview

APL provides transaction privacy for ledger participants while letting authorized parties compute on the private data. APL acts as a layer atop a permissioned blockchain, and remains agnostic to the consensus protocol used.

APL has two main types of actors: auditors that run computations over the transactions on the ledger and banks that add transactions to the ledger and/or also audit the ledger. Banks have their own public/private key pair, and use ledger-wide cryptographic parameters to hide transaction details and interact with auditors. Figure 5-1 demonstrates how actors interact in our system.

The ledger in APL is the authoritative source of all transactions that have occurred in the system, and specifies ownership of the digital assets. This guarantees to the auditor that their analysis is on the complete holdings of each bank, and that assets are not inaccessible.

APL currently only supports a single asset stored on the ledger, but it is future work to extend APL to handle multiple assets on a single ledger.

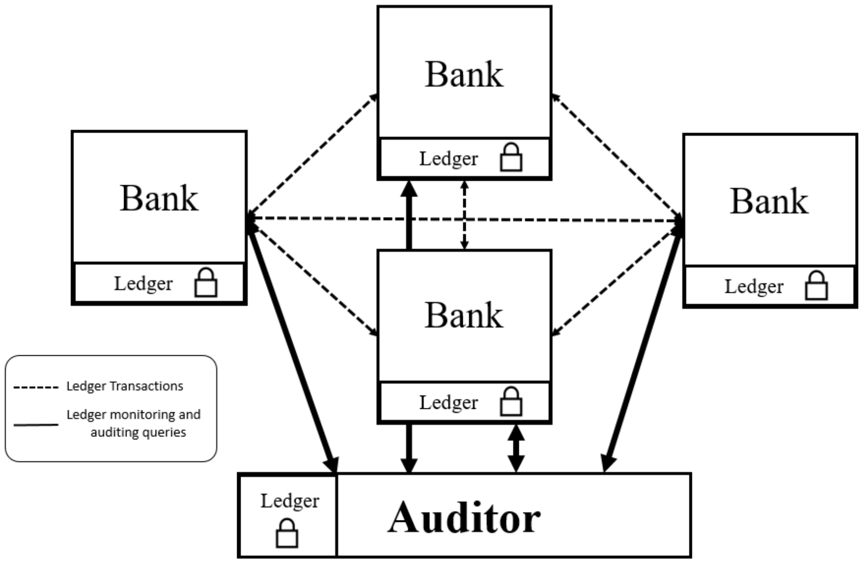


Figure 5-1: APL System Overview

5.1.1 Goals

In designing APL, we want to provide certain correctness guarantees about our system. First, we want to make sure that each transaction is verifiable. Verifiable transactions consist of assets neither being created nor destroyed unless done so publicly, that the spender has sufficient assets to transfer, and the spender has the correct privileges to transfer the claimed assets. Second, when auditing, we want transactions to have sufficient information in them so that banks can respond to auditor queries even for transactions they were not involved in. We want to be able to represent a sufficiently expressive query language that would provide auditors metrics of interest.

Given the correctness properties, maintaining privacy of transaction amounts and the transaction graph is critical. These guarantees, coupled with the inability for participants to undetectably lie about their assets when transacting, were important in influencing design choices and developing a robust system. We adopt the cryptocurrency mantra of focusing on trust first, then efficiency.

We want to maintain a high transaction throughput that scales with the number of banks to provide efficient clearing of securities, so to this end we use lightweight specialized protocols that give us our security guarantees.

5.1.2 Threat Model

We assume a set of untrusted banks appending transaction information to a ledger that is the authoritative source of the holdings of each bank, and an auditor that has read-only access to this ledger. All communication happens through authenticated data channels via public/private key pairs that each bank and the auditor have. Banks may keep local copies of the data stored on the ledger, but only the information stored on the ledger is considered valid. Only transactions deemed valid by the underlying consensus mechanism make it onto the ledger. Banks may update internal accounts as normal, but new or withdrawn assets must be recorded on the ledger.

From these assumptions, we consider the possibility that an individual bank attempts or a set of colluding banks attempt to lie about their holdings to auditors, or to non-colluding banks. Banks may also try to steal assets, transfer the assets of other banks, manipulate their ledger assets, or falsify proofs. Malicious auditors may try to learn more about the holdings of each bank such as fine-grained trading strategy of the bank, and may also collude with banks to negatively impact a target bank.

We assume that the ledger follows the implemented consensus protocol correctly, and that denial of service to the protocol/network layer does not occur.

5.2 Architecture

APL is a distributed system run by banks and auditors that maintains a tamper-resistant append-only ledger. There is no trusted third party that verifies transactions, but all banks and auditors monitor the correctness of transactions appended to the ledger.

5.2.1 API

We define a public transaction TX and private transaction pTX that contain the data that will be stored in the ledger.

Definition 7 (Public Transaction (TX)) *A **public transaction (TX)** in APL is an n-entry row vector that contains a change in asset balance for all n participants in APL. The entries are public and visible to everyone on the ledger.*

Definition 8 (Private Transaction (pTX)) *A **private transaction (pTX)** in APL is an n-entry row vector that contains a change in asset balance for all n participants in APL with the values and transacting parties hidden.*

The ledger has three types of entries: issuances, withdrawals, or transfers. Issuance and withdrawal transactions are public transactions that require the signature of the bank that is performing the public change. A transfer transaction is a private transaction and is valid if the sum of all asset change commitments are equal to 0, and the entries are verified correct via zero-knowledge proofs. In Section 5.2.2 we further explain the zero-knowledge proofs contained in private transactions, and what makes them valid.

The following describes the API that APL follows:

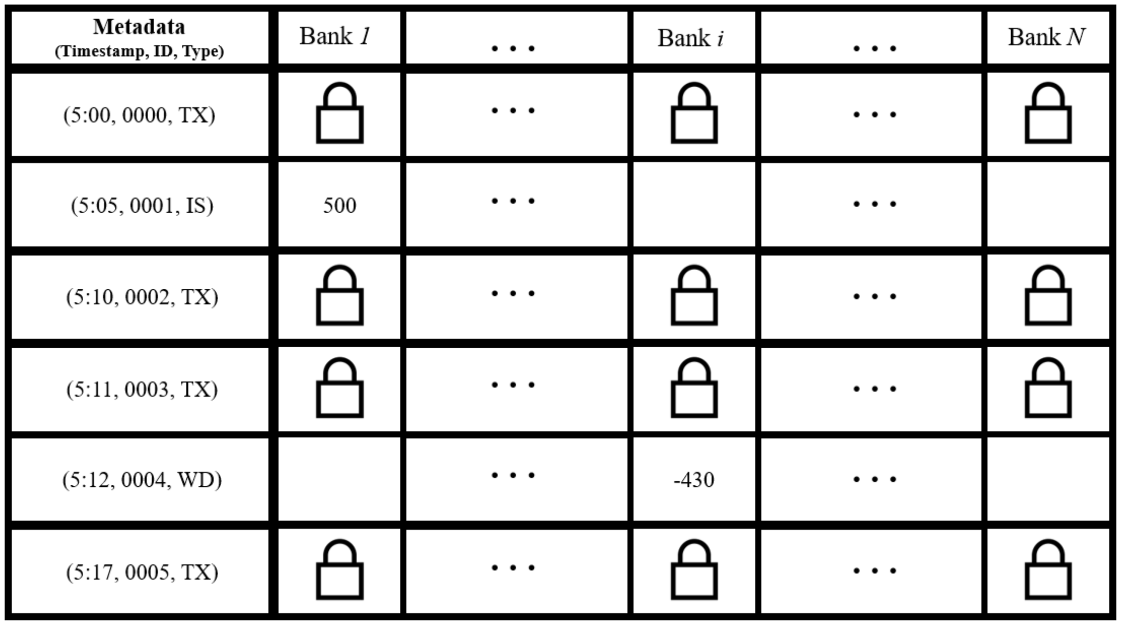
Definition 9 (APL Model) *Auditable Private Ledgers (APL) is a system atop a distributed ledger that provides privacy to ledger participants while letting authenticated third parties perform computations on private data. APL has the following methods:*

- $\mathsf{Setup}(n, \kappa) \rightarrow ([(pk_1, sk_1), \ldots (pk_n, sk_n)], G, H)$*: Given a security parameter κ, produces a public/private key pair for each bank, and group generators G and H.*

- $\mathsf{Transfer}(i, j, v) \rightarrow pTX$*: Produces a private transaction pTX from Bank i to Bank j, with value v, and generates the necessary zero-knowledge proofs for the transaction to be valid.*

- $\mathsf{Issue}(i, v) \rightarrow TX$*: Produces a public transaction TX by Bank i that publicly issues value v, and generates a valid signature for this issuance.*

- $\mathsf{Withdraw}(i, v) \rightarrow TX$*: Produces a public transaction TX by Bank i that publicly withdraws value v, and generates a valid signature for this withdrawal.*



Metadata (Timestamp, ID, Type)	Bank *1*	. . .	Bank *i*	. . .	Bank *N*
(5:00, 0000, TX)	🔒	. . .	🔒	. . .	🔒
(5:05, 0001, IS)	500	. . .		. . .	
(5:10, 0002, TX)	🔒	. . .	🔒	. . .	🔒
(5:11, 0003, TX)	🔒	. . .	🔒	. . .	🔒
(5:12, 0004, WD)		. . .	-430	. . .	
(5:17, 0005, TX)	🔒	. . .	🔒	. . .	🔒

Figure 5-2: Ledger description with locks representing private transactions, the visible numbers on participants public transactions, and metadata listing the time, ordering, and transaction type (IS: issuance, WD: withdrawal, TX: transfer transaction)

- $\mathsf{Audit}(f(\cdot), \{1, ..., n\}) \rightarrow \{0,1\}^*$*: Given a function* $f(\cdot)$ *and indices of ledger rows* $\{1, ..., n\}$*, evaluate* $f(\cdot)$ *over all entries at the given indices and return the function output.*

5.2.2 Ledger Construction

The ledger contents are stored in a table format, where columns are banks and a row is a transaction. Each transaction contains entries for each bank, as well as metadata such as index, timestamp, and transaction type. Figure 5-2 shows what the ledger looks like with locks representing data that is private.

With the table format we have each bank included in each transaction, hiding who is transacting with whom. With each transaction entry having specially constructed data indistinguishable from random, we are able to obfuscate the sender and receiver of a transaction, preventing an adversary that watches the ledger from producing a transaction graph. Because of the way we designed our table, our design requires banks to be able to use information in transactions in which they were not involved when responding to auditors.

For a transaction of value v from bank A to bank B, the value of $-v$ is committed to in A's entry, v in B's entry, and 0s in all other entries. We build here the different components of the transaction entry and how they all relate to achieve our goals.

Commitments Each column entry i includes a Pedersen commitment CM_i to the transacted value v_i. The purpose of this Pedersen commitment is to hide v_i while providing a binding to the auditor to v_i. The sending bank chooses the Pedersen commitment randomness so that the sum of the commitments in the transaction add up to 0, preventing the creation or destruction of assets in a pTX. Given $n-1$ randomly chosen r values, the sending bank chooses r_n so that $\sum^{n-1} r_i + r_n \equiv 0 \mod p$ where p is the order of the elliptic curve group $\mathbb{E}(\mathbb{Z}_p)$. When verifying, banks and auditors check the sum of commitments:

$$\sum_{i=1}^{n} CM_i = \sum_{i=1}^{n} v_i G + r_i H = \sum_{i=1}^{n} v_i G = 0$$

Since the r_i are chosen to add up to 0, the v_i must also add up to 0 to guarantee a balance of assets.

Proof of Assets When transacting, verifiers need to be assured that banks have sufficient funds to transact while still hiding which banks are transacting. We use a disjunctive zero-knowledge proof on a second commitment we call $CM_{\mathsf{AUX},i}$ with a new randomness r'_i, which is either a recommitment of the same value v_i or a commitment to the sum of values in column i, $\sum_{j=0}^{n} CM_{i,j}$. We also add a range proof proving that $CM_{\mathsf{AUX},i}$ is a nonnegative value upwardly bounded by 2^{64}. The disjunctive proof claims that $CM_{\mathsf{AUX},i}$ is one or the other, without revealing which side it proves thus hiding which bank is transacting.

Supplemental Recovery and Proving Information In order to allow the receiving bank to recover v_i and have the disjunctive proof incorporate the bank's assets, we need to provide supplemental information in the transaction. We create what we refer to as a $B_{\mathsf{Token},i}$ (pronounced "b-token"), which is the multiplication of the randomness used in CM_i with the public key pk_i of bank i. Instead of involving each bank in the creation of a transaction, which could allow a bank to block the creation of a transaction, using $B_{\textsf{Token},i}$, a bank not involved in the transaction can recover v_i by multiplying CM_i by the inverse of their secret key sk_i^{-1} to recover r_iH, subtracting r_iH from CM_i to get v_iG, and iterating through potential values ρ until confirming ρG equals v_iG. We also store the randomness used in $CM_{\textsf{AUX},i}$ as $B_{\textsf{Token,AUX},i}$ to recover the value in $CM_{\textsf{AUX},i}$ and incorporate it into the disjunctive proof.

Given $B_{\textsf{Token},i} = r_i \cdot pk_i$, the bank generating the transaction proves they either know the secret key of column i, and hence the assets of the column, or that they know the difference between r_i in CM_i and r_i' in $CM_{\textsf{AUX},i}$, which they always know. Given $SA_i = \sum_{j=0}^{n} CM_{i,j}$ and $SB_i = \sum_{j=0}^{n} B_{\textsf{Token},i,j}$ we construct the disjunctive proof to prove one of the bullets, and simulate the other:

- If $CM_{\textsf{AUX},i}$ is a recommitment of $SA_i = \sum_{j=0}^{n} CM_{i,j}$, then:

$$\begin{gathered} log_{(CM_{\textsf{AUX},i}-SA_i)}(B_{\textsf{Token,AUX},i} - SB_i) = \\ log_{(\sum v_{i,j}G+r_i'H-\sum v_{i,j}G-\sum r_{i,j}H)}((r_i' \cdot sk_i)H - (\sum r_{i,j} \cdot sk_i)H) = \\ log_{((r_i'-\sum r_{i,j})H)}(sk_i \cdot (r_i' - \sum r_{i,j})H) = sk_i \end{gathered}$$

- If $CM_{\textsf{AUX},i}$ is a recommitment of CM_i, then:

$$\begin{gathered} log_H(CM_{\textsf{AUX},i} - CM_i) = \\ log_H(v_iG + r_i'H - v_iG - r_iH) = \\ log_H(r_i' - r_i)H = r_i' - r_i \end{gathered}$$

Note that if an evil bank E creating the transaction decides to prove that they know the difference between r_i' and r_i for their own entry, then each entry in the transaction must be a commitment to 0 since the sum of all commitments have to add up to 0 and E cannot steal funds from another bank. Suppose that E tried to steal funds by putting a commitment to a negative amount in an entry that is not their own and tries to avoid getting caught. Since we have a range proof on

CM_{AUX}, they would have to prove they have enough assets since $CM_{\mathsf{AUX},i}$ cannot be a commitment to the negative value. In order to prove they have enough assets, they need to know the secret key of the bank they are attempting to steal funds from, so unless they comprise an entire bank, they cannot spend as that bank. Since all other entries in a transaction must be positive, theirs must be negative to balance out the entries, else the transaction will not be valid because the commitments do not sum to 0. The only way the sending bank can decide to prove $r' - r$ in the disjunctive proof is for every entry to be a commitment to 0.

Since each transaction relies on the sum of the previous $n - 1$ rows for each column, a malicious bank cannot perform a replay attack using a previously constructed transaction pTX_{Original} because the proof of assets will not correctly incorporate the addition of the value commitment CM_{Original} from the replayed row. A replayed transaction pTX_{Replay} would attempt to be added at some point $n + \delta$ in the ledger, and a verifier calculating SA would notice SA_{Replay} does not include the extra δ transactions that precede it and reject it from the ledger.

Consistency Proofs With $B_{\mathsf{Token},i}$ and $B_{\mathsf{Token,AUX},i}$, we need to ensure that a bank is creating the tokens with the correct r_i and r'_i values used in the commitments and not producing a corrupt entry that other banks cannot open to an auditor or cannot be used to transact. To address this issue, the sending bank must produce a consistency proof that shows the r_i value used in CM_i is the same as the one in the $B_{\mathsf{Token},i} = r(sk_i)H$ and that the r'_i value used in $CM_{\mathsf{AUX},i}$ is the same as the one in $B_{\mathsf{Token,AUX},i} = r'_i(sk_i)H$. This consistency proof prevents a bank from lying in the $B_{\mathsf{Token},i}$ for each entry.

The proof follows from Figure 4-4. What gets written to the ledger is A_1, A_2, c, s_1, and s_2 for both CM_i and $CM_{\mathsf{AUX},i}$ so that banks and auditors can verify the consistency with the associated B_{Token}.

Overall, transfer transactions consist of:

- commitment CM_i and $B_{\mathsf{Token},i}$ of the original value,

- recommitment $CM_{\mathsf{AUX},i}$ of either the total assets or the original value along with the associated $B_{\mathsf{Token,AUX},i}$,

- consistency proofs between the re/commitments and B_{Token}'s,
- and a range proof on the recommitment, along with a disjunctive proof stating what the recommitment is to.

5.2.3 Ledger Functions

From the API description, we go into detail of what each function does on the ledger.

Setup At system start, either a trusted third party setting up the system or banks and financial auditors choose cryptographic public parameters from a standard set of parameters. These parameters include a cryptographic hash function $\mathcal{H}$, an elliptic curve group $\mathbb{E}(\mathbb{Z}_p)$, group generators G and H, and scalar group modulus p. Once parameters are chosen, each bank creates their own public/private key pair and shares their public key with each participant, allowing for authenticated channels and the capability to verify proofs.

Transfer A transfer transaction occurs between banks, and assets are only transferred when a sending bank produces a proof of sufficient assets. If all disjunctions in the proof of assets prove a recommitment of the original value, then each value must be 0 and no assets will be transferred. Note that these two types of transactions are indistinguishable because banks choose different commitment randomness values in each entry.

For each bank i in the ledger, the following is included for a transfer transaction n:

- Commitment
  - A commitment CM_i of v_i with randomness r_i: $CM_i = v_iG + r_iH$
  - The multiplication of the randomness r_i with their public key pk_i: $B_{\mathsf{Token},i} = r_ipk_i$
  - A consistency proof that the randomness used in the commitment is the same as the one multiplied with their public key

- Commitment$_{\mathsf{AUX}}$
  - A commitment $CM_{\mathsf{AUX},i}$ to either the same value v_i using a new randomness r'_i or the bank's entire balance $\sum_j v_{i,j}$ which is the sum of all entries in their column including the current transaction
  - The multiplication of the new randomness r'_i with their public key pk_i: $B_{\mathsf{Token},\mathsf{AUX},i} = r'_i pk_i$
  - A consistency proof that r'_i is the same in the above two operations
- Proof of Assets
  - A range proof on $CM_{\mathsf{AUX},i}$ proving that it is in the range $[0, 2^{64})$.
  - A disjunctive proof that either $CM_{\mathsf{AUX},i}$ is a commitment to the their total balance or a recommitment of the originally transacted value.

The commitment random values are chosen such that the sum of all CM_i in the transaction add up to the identity of the elliptic curve group, ensuring that no new value is created in this transaction. This requirement is secure as long as the bank cannot solve the ECDLP on the two generators for our group, G and H.

In order to add an entry to the ledger, all banks must verify each proof in the transaction, as well as verify that the sum of the commitments in the transaction add up to the identity of the elliptic curve group.

Issuance/Withdrawal An issuance/withdrawal transaction is produced by a bank that aims to add/remove an amount of an asset to the system and is completely public. This comes with a proof of knowledge of the bank's secret key to assure others are not forging this transaction.

Audit Auditing is covered in depth in the next chapter.

Chapter 6

Auditing Protocols

In this section, we describe two protocols for running an audit on the APL system: private sums and Herfindahl Index calculation. For auditing we have two goals: (1) to run functions that can provide effective measures of systemic risk, and (2) to ensure that banks are verifiably honest in their responses. The protocols are built off of linear functions on the Pedersen commitments stored on the ledger. These functions let auditors learn aggregate information while hiding fine-grained details about transactions.

Auditing is an interactive protocol between the bank being audited and the auditor. The auditor requires access to a ledger and may be online connected to a ledger verifying the transactions being added in real time, or may come online at a later time to verify an active ledger.

6.1 Private Sums

Auditing sums is an interactive protocol between a bank A and the auditor. The bank A sends the total value of the asset they have to an auditor as well as an equivalence proof that this value matches what is on the ledger. Since the auditor has access to the ledger it knows the sum of A's column, so they can verify that the answer provided by the bank is consistent with what the auditor has seen.

The zero-knowledge proof of equivalence assures the auditor the sent value is the

Equivalence Σ Protocol

Prover
knows sk_A

Verifier
knows $SA - \Sigma v_{A,i}G = \Sigma r_{A,i}H$
knows $SB = \Sigma B_{\mathsf{Token},A,i} = \Sigma r_{A,i} sk_A H$
knows $PK_A = sk_A H$

$k \xleftarrow{\$} \mathbb{Z}_p$
$T_1 = k(SA - \Sigma v_{A,i}G)$
$T_2 = kH$
$c = \mathcal{H}(H, SA - \Sigma v_{A,i}, PK_A, SB, T_1, T_2)$
$s = k + c \cdot sk_A$

$\xrightarrow{T_1, T_2, s, c}$

check $c \overset{?}{=} \mathcal{H}(H, SA - \Sigma v_{A,i}G, PK_A, SB, T_1, T_2)$
check $s(SA - \Sigma v_{A,i}G) \overset{?}{=} T_1 + cSB$
check $sH \overset{?}{=} T_2 + cPK_A$

Figure 6-1: Equivalence proof for sum of assets

same as what is on the ledger without the bank knowing the randomness used in each commitment. The proof of equivalence consists of the bank A proving knowledge that their secret key sk_A is the solution to two different discrete logarithms. Figure 6-1 shows the auditing protocol with the sum of the commitments on the ledger $SA = \sum_{i=0}^{n} CM_{A,i} = \sum_{i=0}^{n} v_{A,i}G + \sum_{i=0}^{n} r_{A,i}H$ and the sum of the supplemental information $SB = \sum_{i=0}^{n} B_{\mathsf{Token},A,i} = \sum_{i=0}^{n} r_{A,i} sk_A H$. The auditor computes $\Sigma v_{A,i}G$ with the value given by the prover (bank A), and subtracts it from SA to get $\Sigma r_{A,i}H$. The prover uses $\Sigma r_{A,i}H$ as a base to prove that $SB = \Sigma r_{A,i} sk_A H$ equals $\Sigma r_{A,i}H$ times sk_A and that that same value is also the solution to $\log_H PK_A = \log_H sk_A H$.

Through this protocol, the auditor has an assurance that the total sum of an asset of a bank is correct without the bank having to reveal their individual transactions. If a bank attempted to provide a different total number of assets than those stored on the ledger, then $SA - \Sigma v_i G \neq \Sigma r_i H$ since the Σv_i would not cancel out, and the bank's attempt to cheat would be detected.

6.2 Herfindahl Index

The Herfindahl Index builds on the sum protocol and is used by auditors to compute the contagion of a particular asset [12]. The Herfindahl Index calculation consists of getting the sums of all participants, summing this value to get the total number of assets across the system, then doing a sum-of-squares on the market share of each participant.

The Herfindahl Index protocol goes as so: for each participant i up to all n participants the auditor gets the sum a_i via running the private sum protocol above in parallel, then the auditor computes the total T when all protocols complete, divides each a_i by the total T to get market share and finally does a sum-of-squares on these values:

$$H = \sum_{i=1}^{N} \left(\frac{a_i}{T}\right)^2 .$$

Note that the banks do not have to cooperate, and that the auditor can contact each bank in parallel.

Security of the Herfindahl Index protocol is based the security of the private sum protocol. As long as all sum protocols execute correctly, then the Herfindahl Index protocol will also execute correctly.

Chapter 7

Implementation

We implement our system in Go and use a modified version of the *btcec* library [9] that contains the parameters and methods to compute with the elliptic curve secp256k1. We also take advantage of Go's built-in SHA-256 implementation for use as our cryptographic hash function. In total we have over 3200 lines of code, with 40% of it cryptographic operations.

7.1 Structure

We follow the client/server paradigm in the development of APL. Servers provide an interface to APL that uses the API outlined in Definition 9, and clients take advantage of this interface for different roles.

As APL is a distributed system with no central trusted party, each participant runs an APL server and associated client based on their role. Participants must agree on a set of cryptographic elliptic curve public key parameters that may be chosen by a trusted party setting up the system, or by participants themselves. Public keys are distributed to all participants before starting the system.

7.1.1 Servers

Servers contain code for networking, cryptography, and interacting with the permissioned blockchain. Servers provide the foundation for APL and handle interactivity with servers running on other machines.

The server code interfaces with the ledger used. For our initial implementation, we built an in-memory append-only ledger that all participants interface with as a third-party service. This service appends valid transactions and broadcasts the appends to all banks. Each bank and the auditor keep a copy of the ledger to respond to/confirm auditing queries. Banks keep plaintext of their total assets so that they can quickly produce proof of assets in transactions and answer auditor queries.

My largest contribution was implementing the cryptographic components of APL, consisting of Pedersen Commitments, and the zero-knowledge proofs in our system. We opted to re-implement Blockstream's range proofs [39] in Go for understandability and ease of referencing in our code.

7.1.2 Clients

Clients are the gateway to the APL API, and provide methods depending on their role. There are three types of clients:

- ledger clients that abstract the ledger technology used;
- bank clients that let banks manage keys and send transactions to the ledger; and
- auditor clients that let auditors query banks and verify transactions as they get appended to the ledger.

The ledger client lets banks and auditors append items to the ledger. In our implementation, there is one ledger client that all banks and auditors talk with for ease of consensus and message passing.

Bank clients initiate transactions on the ledger and respond to auditor queries. For queries, bank clients currently can only respond to private sum queries.

Auditor clients initiate queries to each bank and perform client-side calculations with bank responses to calculate systemic risk.

7.2 Optimizations

Caching on the bank and auditor clients improves computation speed. Currently each client stores a rolling sum of commitments and supplemental information (SA and SB respectively) so that they can quickly produce proof of assets and answer queries from auditors. For the private sum protocol, instead of having to sum up a subset of ranges, the ledger could include the sum up to a particular point so that computation costs are replaced with storage costs. This would only include another Pedersen commitment on the ledger, and would provide quick verification of subsets of the ledger for queries that measure transaction volume for a particular duration.

Transaction throughput is sped up by parallelization of the range proof generation and keeping a repository of range proofs for the value 0 for transactions that do not include every participant.

Chapter 8

Evaluation

In this chapter we measure the performance of APL's two features: appending to a ledger, and auditing a ledger.

8.1 Setup and Tests

We evaluate our system on a machine with two 6-core Intel i7 x980 3.33 GHz 12 MB cache processors and 24 GB of RAM running 64-bit Ubuntu 16.04.2. All services run on separate processes on this single machine. A copy of the ledger is stored in the memory of each process, and none of the processes use the disk. The processes communicate via Go's RPC framework.

We performed a test to measure the performance of appending to a ledger using APL, and a test to measure the performance of auditing ledger content.

8.2 Appending to Ledger

Transaction throughput (appends to the ledger per second) scales in an exponentially decaying manner with the number of participants in APL. Figure 8-1 shows an exponential decay in the throughput as the number of banks increases in APL. This figure represents all banks sending transactions at the same time to the ledger.

An entry in a transaction is around 29 KB, with 26 KB, or around 90%, of this

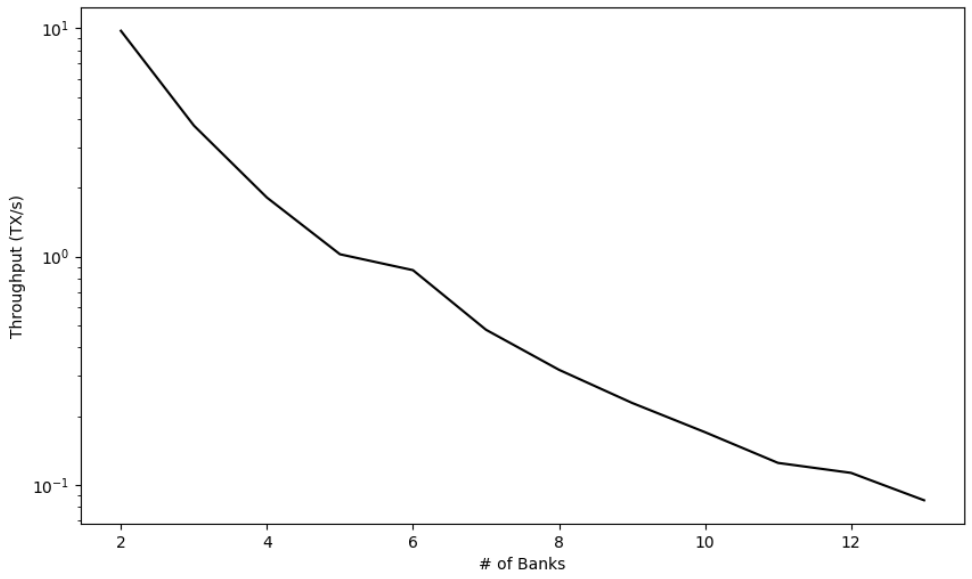


Figure 8-1: pTX throughput with one to thirteen simultaneous transacting banks

space dedicated to range proofs alone. This is without point compression [9], which can halve the representation of elliptic curves.

At two banks sending transactions in parallel, our throughput is around 10 transactions per second, while at thirteen banks sending simultaneous transactions our throughput is around 5 transactions per minute. Another way to look at this data is looking at the latency to append a transaction to the ledger. Figure 8-2 is another look at the same data, showing us with thirteen banks transacting simultaneously it takes 10 seconds to add a transaction to the ledger.

Proofs for transactions takes around 12 ms to produce per entry, and take 10 ms to verify per entry, although proof generation and verification can be done in parallel for all entries. Proofs in transactions are only created at one point, but are verified many times by each actor on the ledger.

As more banks simultaneously transact, transactions end up getting queued since transactions need to be added in an ordered fashion. We run a test looking at the throughput of transactions when only a subset of banks are simultaneously transacting. Figure 8-3 shows a similar decay in throughput when only a subset of banks are

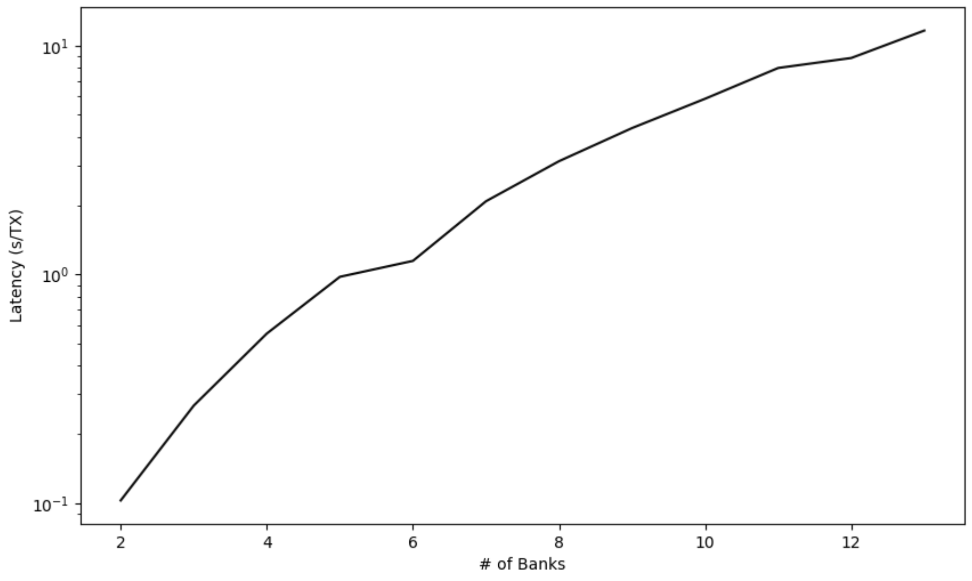


Figure 8-2: pTX latency with one to thirteen simultaneous transacting banks

transacting as the number of banks increases. For each line, the change in throughput as the number of banks increases is the cost to produce a new entry with the associated proofs for each bank. The difference between lines is the cost of queuing for having multiple banks simultaneously transact. Note that each line does not stretch all the way to the left since it does not make sense for three banks to simultaneously transact when there are only two banks.

Most of the time spent producing these transactions is in generating the proofs for each entry. As new banks are added to the ledger, our throughput roughly gets halved as we generate more zero-knowledge proofs to be added to the ledger.

8.3 Auditing

We evaluate auditing by running the private sum auditing protocol varying the number of transactions in the system.

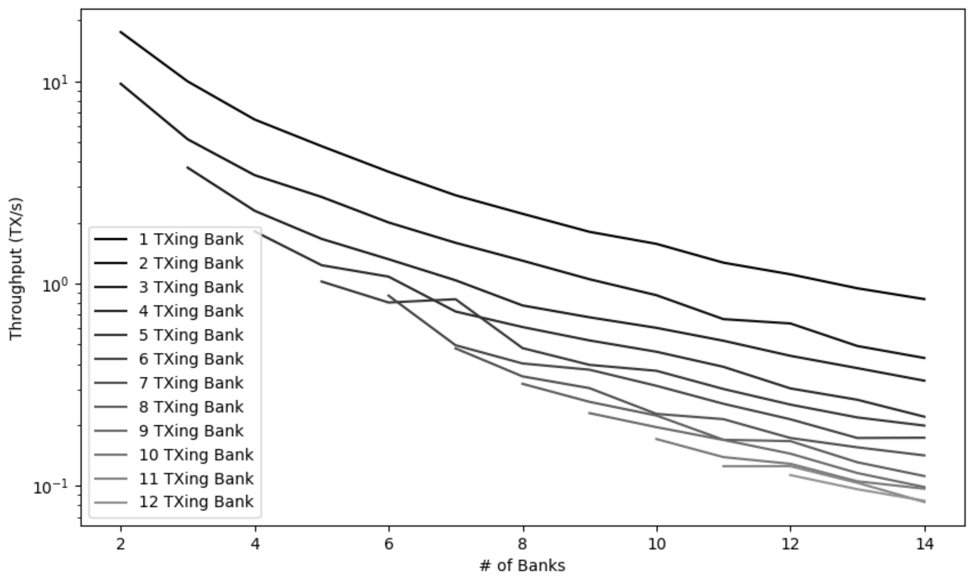


Figure 8-3: pTX throughput with two to fourteen banks, varying simultaneous banks transacting from 1 to 12

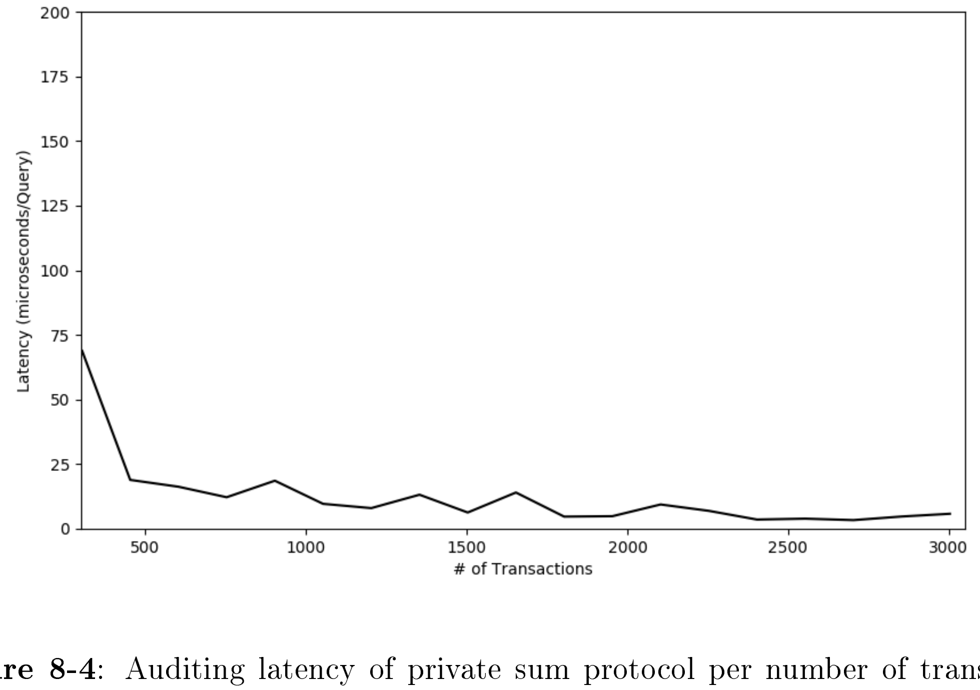


Figure 8-4: Auditing latency of private sum protocol per number of transactions stored on the ledger

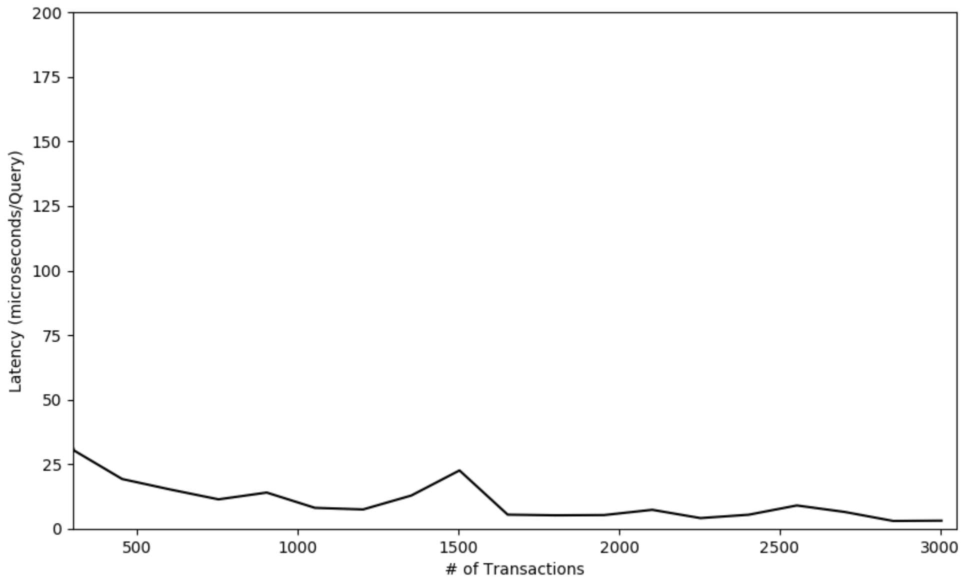


Figure 8-5: Auditing latency of Herfindahl Index protocol per number of transactions stored on the ledger

8.3.1 Private Sums

We implement and test the private sum protocol from Figure 6-1 and measure query latency versus number of transactions. Figure 8-4 shows the latency for answering and verifying queries as we increase the number of transactions on the ledger. The results show that the query is roughly constant taking less than 25 μs for most queries as the number of transactions increases. A small querying time is possible because of the caching and real-time updating by auditors and banks described in Section 7.2.

8.3.2 Herfindahl Index

The Herfindahl Index calculation follows from the private sum calculation with an added requirement to sum the square of the market shares of each bank. Figure 8-5 shows the latency for performing the Herfindahl Index protocol, which roughly resembles the private sum protocol results with most queries under 25 μs and a constant slope. Figure 8-6 compares the differences in computation time between the Herfindahl Index and private sums, with negative values meaning the Herfindahl Index

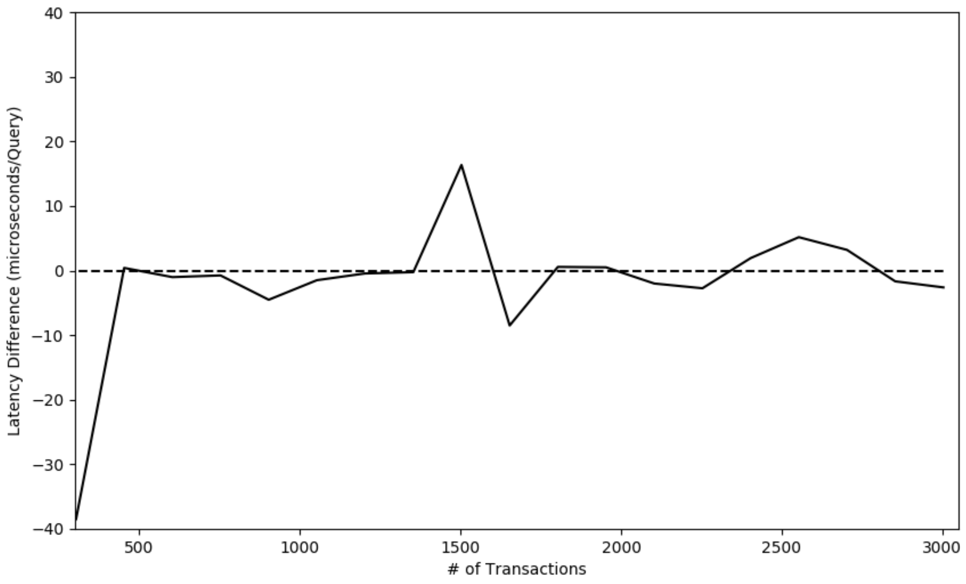


Figure 8-6: Difference in performance between Herfindahl Index protocol and private sum protocol

protocol outperformed the private sum protocol and positive values the opposite. This balance around the x-axis shows that the extra calculations are negligible in calculating the Herfindahl Index.

Chapter 9

Conclusion

We have shown a new way to provide auditability to private data on permissioned blockchains that differs from existing solutions. Our scheme hides transaction amounts, and the transaction graph of participants, while allowing authorized third parties to perform computations in coordination with participants. We have also applied our techniques to financial auditing, providing the capability to perform real-time systemic risk analysis to ensure provably correct results. The fact that the assets stored on the ledger are the true record, and not a copy of the record, guarantees that banks cannot falsify information and that auditors run their functions on true holdings.

In exploring other potential use cases, APL fits nicely in problems that meet the following conditions:

- there are a known set of entities (be it through real identities or pseudonymous via public keys);
- transactions occur between these entities;
- these entities require privacy of the transaction graph and transaction amounts;
- third parties require assurances to the correctness of the system, or require analysis of the transaction information;
- fine-grained analysis of the transactions requires privacy.

9.1 Known Challenges

We do not provide any sort of differential privacy [26]. It is possible for a malicious auditor to query for the total of $n-1$ transactions and then n transactions and determine the individual transaction amount of the nth row. Doing this repeatedly would effectively leak the transaction amounts and graph of all participants. A bank can simply not respond to auditing queries that might reveal too much information. For example, the banks and auditors could agree ahead of time what types of queries are appropriate, and the auditor could attach a digital signature to all queries. It is important to note that providing differential privacy would also limit the ability to get exact systemic risk measurements.

Our ledger requires a strict ordering for new transactions to be added to it. A bank adding a new transaction must use all transactions that appear before it to compute the proofs. This means that a new transaction cannot be created unless it has all previous transaction information included in it. If two or more banks attempt to add a new transaction to the ledger, only one will accept and the rest will have to update their transactions with the new data. This limits our throughput, but updating of the transactions is relatively quick, relying on updating CM_{AUX} and its associated B_{Token} and the proof of assets.

9.2 Future Work

Currently we are working to increase throughput by minimizing cryptographic computation time, efficiently supporting multiple assets on a single ledger, and extending the protocol set for advanced auditing protocols.

Most of the computation time to append to the ledger requires creating and verifying the range proofs. We are working to limit our reliance on range proofs, and/or produce more efficient constructions.

Efficiently supporting multiple assets on a single ledger is an important requirement of our work. Using similar constructions as Confidential Assets [39] [20] to incorporate and hide the asset type requires producing a range proof per asset, which significantly decreases ledger throughput.

With the private sum auditing protocol developed, we want to extend the set of supported auditing protocols to provide more sophisticated types of queries. One area of interest is being able to run machine learning or other artificial intelligence functions over private data [38] [4]. Along with more advanced queries would come UI/UX improvements so that banks and auditors can interact with our system in an intuitive fashion.

While our initial use-case is financial auditing, there are a couple of areas of interest where we think APL could be taken advantage of that we're exploring such as electronic medical records, supply chain management, and Internet of Things.

Bibliography

[1] Financial Stability Monitor | Office of Financial Research. Accessed on August 13, 2017. `https://www.financialresearch.gov/financial-stability-monitor/`.

[2] Hyperledger home. Accessed August 8, 2017. `https://www.hyperledger.org`.

[3] Kadena: Scalable Blockchain | Smarter Contracts. Accessed August 8, 2017. `http://kadena.io/`.

[4] Machine Learning with Financial Time Series Data | Solutions. Accessed on August 14, 2017. `https://cloud.google.com/solutions/machine-learning-with-financial-time-series-data`.

[5] Quorum | J.P. Morgan. Accessed August 8, 2017. `https://www.jpmorgan.com/country/US/EN/Quorum`.

[6] U.S. GAO - The Yellow Book. Accessed on August 12, 2016. `http://www.gao.gov/yellowbook/overview`.

[7] Zcash - All coins are created equal. Accessed December 12, 2016. `https://z.cash/?page=0`.

[8] IC3 - The Initiative For Cryptocurrencies & Contracts, July 2017. `http://www.initc3.org/`.

[9] Package btcec implements support for the elliptic curves needed for bitcoin., July 2017. `https://godoc.org/github.com/btcsuite/btcd/btcec#PrivateKey`.

[10] Crypto StackExchange User: 7sujit. Cryptanalysis - why can't the commitment schemes have both information theoretic hiding and binding properties? - cryptography stack exchange. Accessed June 27, 2017. https://crypto.stackexchange.com/questions/41822/why-cant-the-commitment-schemes-have-both- information-theoretic-hiding-and-bind/41834.

[11] Emmanuel A. Abbe, Amir E. Khandani, and Andrew W. Lo. Privacy-preserving methods for sharing financial risk exposures, 2011.

[12] Emmanuel A. Abbe, Amir E. Khandani, and Andrew W. Lo. Privacy-preserving methods for sharing financial risk exposures. *American Economic Review*, 102(3):65–70, May 2012.

[13] Robleh Ali, Nicholas Vause, and Filip Zikes. Systemic risk in derivatives markets: A pilot study using cds data, July 2016. Available at SSRN: `https://ssrn.com/abstract=2809667` or `http://dx.doi.org/10.2139/ssrn.2809667`.

[14] Digital Asset. The digital asset platform. Accessed on August 3, 2017. `http://hub.digitalasset.com/hubfs/Documents/Digital%20Asset%20Platform%20-%20Non-technical%20White%20Paper.pdf`.

[15] Austin-Williams. Define 'monetary base' and 'audit' in the context of zcash · issue 2289 · zcash/zcash. Accessed July 10, 2017. `https://github.com/zcash/zcash/issues/2289`.

[16] Paul Eric Byrnes, CA Al-Awadhi, Benita Gullvist, Helen Brown-Liburd, CR Teeter, J Donald Warren Jr, and Miklos Vasarhelyi. Evolution of auditing: from the traditional approach to the future audit. *Audit Analytics*, 71, 2015.

[17] Agostino Capponi. *Systemic Risk, Policies, and Data Needs*, chapter 8, pages 185–206.

[18] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, OSDI '99, pages 173–186, Berkeley, CA, USA, 1999. USENIX Association. `http://dl.acm.org/citation.cfm?id=296806.296824`.

[19] Ethan Cecchetti, Fan Zhang, Yan Ji, Ahmed E. Kosba, Ari Juels, and Elaine Shi. Solidus: Confidential distributed ledger transactions via pvorm. *IACR Cryptology ePrint Archive*, 2017:317, 2017.

[20] Chain. Hidden in Plain Sight: Transacting Privately on a Blockchain, February 2017.

[21] Mario Christodoulou. U.k. auditors criticized on bank crisis. *Wall Street Journal*, March 2011. `http://www.wsj.com/articles/SB10001424052748703806304576232231353594682`.

[22] Gaby G. Dagher, Benedikt Bünz, Joseph Bonneau, Jeremy Clark, and Dan Boneh. Provisions: Privacy-preserving proofs of solvency for bitcoin exchanges. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 720–731, New York, NY, USA, 2015. ACM.

[23] Phil Daian, Rafael Pass, and Elaine Shi. Snow white: Provably secure proofs of stake. Cryptology ePrint Archive, Report 2016/919, 2016. `http://eprint.iacr.org/2016/919`.

[24] Ivan Damgård. On σ-protocols. *Cryptologic Protocol Theory 2010, v.2*, 2002. Available at `http://www.cs.au.dk/~ivan/Sigma.pdf`.

[25] Lisa Du. To Understand JPMorgan's Trading Fiasco You Have To Go Back To 2005.

[26] Cynthia Dwork. Differential privacy. In *Proceedings of the 33rd International Conference on Automata, Languages and Programming - Volume Part II*, ICALP'06, pages 1–12, Berlin, Heidelberg, 2006. Springer-Verlag.

[27] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Proceedings on Advances in cryptology—CRYPTO '86*, pages 186–194, London, UK, UK, 1987. Springer-Verlag.

[28] FinCEN. United states department of the treasury financial crimes enforcement network | fincen.gov. Accessed June 27, 2017. `https://www.fincen.gov/`.

[29] Christina Garman, Matthew Green, and Ian Miers. Accountable privacy for decentralized anonymous payments. Cryptology ePrint Archive, Report 2016/061, 2016. `http://eprint.iacr.org/2016/061`.

[30] Mike Hearn. Corda: A distributed ledger. `https://docs.corda.net/_static/corda-technical-whitepaper.pdf`.

[31] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, July 1982. url-http://doi.acm.org/10.1145/357172.357176.

[32] Ueli Maurer. Unifying zero-knowledge proofs of knowledge. In *Proceedings of the 2Nd International Conference on Cryptology in Africa: Progress in Cryptology*, AFRICACRYPT '09, pages 272–286, Berlin, Heidelberg, 2009. Springer-Verlag.

[33] Silvio Micali. ALGORAND: the efficient and democratic ledger. *CoRR*, abs/1607.01341, 2016. `http://dblp.uni-trier.de/rec/bib/journals/corr/Micali16`.

[34] Edward V Murphy. Who regulates whom and how? an overview of us financial regulatory policy for banking and securities markets. 2015.

[35] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2009. `http://bitcoin.org/bitcoin.pdf`.

[36] Sunoo Park, Krzysztof Pietrzak, Albert Kwon, Joël Alwen, Georg Fuchsbauer, and Peter Gaži. Spacemint: A cryptocurrency based on proofs of space. Cryptology ePrint Archive, Report 2015/528, 2015. `http://eprint.iacr.org/2015/528`.

[37] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '91, pages 129–140, London, UK, UK, 1992. Springer-Verlag.

[38] Le Trieu Phong, Yoshinori Aono, Takuya Hayashi, Lihua Wang, and Shiho Moriai. Privacy-preserving deep learning via additively homomorphic encryption. Cryptology ePrint Archive, Report 2017/715, 2017. `http://eprint.iacr.org/2017/715`.

[39] Andrew Poelstra, Adam Back, Mark Friedenbach, Gregory Maxwell, and Pieter Wuille. Confidential assets, 2017. 4th Workshop on Bitcoin and Blockchain Research.

[40] R3. R3. Accessed July 2, 2017. `http://www.r3cev.com/`.

[41] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *Proceedings of the 9th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '89, pages 239–252, London, UK, UK, 1990. Springer-Verlag.

[42] Tendermint. Tendermint. `https://perma.cc/X7XM-7LRY`.