Verifiable Value Added Tax

Victor Sint Nicolaas[1] and Sascha Jafari[2]

[1] Provable
victor@provable.com
[2] summitto
sascha@summitto.com

Abstract. Value Added Tax (VAT) is a cornerstone of government revenue systems worldwide, yet its self-reported nature has historically been vulnerable to fraud. While transaction-level reporting requirements may tackle fraud, they raise concerns regarding data security and over-reliance on tax authorities as fully trusted intermediaries. To address these issues, we propose Verifiable VAT, a protocol that enables confidential and verifiable VAT reporting. Our system allows companies to confidentially report VAT as a homomorphic commitment in a centrally managed permissioned ledger, using zero-knowledge proofs to provide integrity guarantees. We demonstrate that the scheme strictly limits the amount of fraud possible due to misreporting. Additionally, we introduce a scheme so companies can (dis)prove exchange of VAT with fraudulent companies. The proposed protocol is flexible with regards to real-world jurisdictions' requirements, and underscores the potential of cryptographic methods to enhance the integrity and confidentiality of tax systems.

Keywords: Tax · Zero-Knowledge Proofs · Incrementally Verifiable Computation · Accumulators.

Table of Contents

1 Introduction

Value Added Tax (VAT) has taken the world by storm. Since its introduction in the 1950s, it is now one of the largest sources of government tax revenue [ROO23].

VAT works as follows. Suppliers are obligated to charge a percentage of tax on top of their selling price. The supplier has to remit it to the tax authority, whereas the buyer can ask it back, but only if they are also an economic entity. As a result, at every step of the value chain, the difference between a firm's purchases and sales, the value added, is taxed [3].

Unfortunately, when the tax is self-reported, as it has been in most countries for most of VAT's history, there is a risk of malicious or accidental differences between the supplier and buyer's reported amounts. Companies are effectively given a theoretically unlimited "VAT credit" because collected VAT is not returned to the tax authority immediately, only by the next VAT return. Billions are lost each year due to such misreporting and fraud [Mur19] [Com18]. As a concrete example : Alice sells apples worth 100 EUR (+ 20 EUR output VAT) to Bob, who in turn sells bottled apple juice worth 200 EUR (+ 40 EUR output VAT) to consumer Charlie. Both companies charge the local VAT rate of 20%. Given that Bob is allowed to deduct his input VAT, the tax authority ultimately gains 40 EUR in VAT. If however, Bob reports higher input VAT, the tax authority will gain less, or can even *lose* money.

In the fight against fraud, many tax authorities have mandated various forms of reporting requirements [con23], revealing invoice details to the tax authority and linking the report of the supplier and buyer. This resolves the aforementioned fraud, as there can be no discrepancy between Alice and Bob's reporting. The major downside is that all transaction details and company relations become visible to the tax authority. In this setup, tax authorities are fully trusted intermediaries, responsible for the integrity and privacy of the VAT reporting. To overcome these issues, we propose a scheme for confidential and verifiable reporting of VAT.

Contributions. We design a confidential VAT reporting system, which offers key fraud prevention benefits of a transparent invoice reporting system without revealing invoice details to the tax authority by default. We formally analyze the security and guarantees which the system can offer, namely that the maximum amount which fraudulent companies can extract is bounded by the amounts reported by honest companies.

[3] A useful property of VAT is its economic "neutrality" - it aims to directly tax "value added" and to minimize distortions to firms' production decisions. A closely related tax system which also aims to have similar neutrality is the US Sales Tax. In a Sale Tax, the full burden of tax collection lies on only the few companies at the end of the supply chain, creating a distortionary impact on economic activity.

2 Overview

2.1 Goals

While reporting requirements can help to reduce VAT fraud, transparent reporting of invoice data to tax authorities has two major downsides:

1. A large amount of **confidential** business information is shared with tax authorities, which makes it possible that the data is leaked to unintended parties.
2. The tax authority is assumed to be a fully trusted party, responsible for the **integrity** of the system.

Design goals to overcome the above two obstacles include:

1. Company transaction details should be committed to, but they should not be revealed by default to the tax authority. Only the end balance at the end of each VAT period should, which is the minimum requirement for VAT to function.
2. Companies and third parties should be able to verify that reported commitments are being tracked correctly.

Making VAT verifiable has the potential to bring many exciting additional benefits: 1. Improved government revenue-based finance 2. More accurate national economic statistics 3. More accurate bookkeeping and easier audits with regards to other taxes.

In reality, VAT systems are more complicated than outlined in the model in this paper. There are reporting requirements which can change depending on the sector, type of product and jurisdiction. Still, the basic system outlined in this paper should be implementable in any jurisdiction and leaves enough flexibility to enforce additional policies.

2.2 System Model & Trust Assumptions

We build upon the model introduced by Platypus [WKDC22]. We consider the setting in which a tax authority wants to issue and maintain a public transaction log, as shown in Figure 1. For simplicity, we assume the tax authority is responsible for enforcing regulatory requirements, but note that arbitrary tasks can theoretically be delegated to a regulator like in Platypus.

Companies reporting VAT are considered untrusted, i.e., they may behave arbitrarily. Since tax authorities are responsible for VAT collection, we assume that the tax authority is trusted for the integrity of company enrollment. Given that commitments to any state updates are publicly recorded, it is possible for any external auditor to verify that regulatory requirements were indeed upheld.

We assume companies have access to secure messaging channels. For censorship resistance and enhanced privacy, transactions can be submitted to a distributed ledger or using onion routing like Tor.

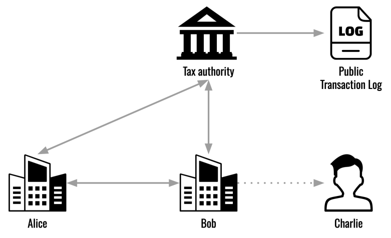


Fig. 1. The Verifiable VAT reporting model. The tax authority is responsible for transaction validation and publishes a log of all transactions.

We present and analyse our VAT reporting system with a number of simplifying assumptions, each of which can easily be loosened up to complement more complex real-world VAT systems:

- All companies adhere to the same VAT period.
- There is a fixed set of companies and they are all operational during the full VAT period.

Our Verifiable VAT scheme does not eliminate all fraud. For example, companies can lie about how much VAT they received from consumers. Companies may also disappear and not pay the VAT deficit which they owe the tax authority. However, the maximum fraud due to misreporting is limited by the reporting of honest firms (formally argued in section 6.3), and we can discover direct trading partners of disappearing companies (formally argued in section 6.4).

2.3 Verifiable VAT Design

The key insight underlying this work, is that **VAT credit can be represented as a privacy preserving homomorphic commitment**. Instead of requiring companies to transparently report the amount of VAT in a transaction to the tax authority, tokenized VAT credit can be transferred from suppliers to buyers in a confidential manner.

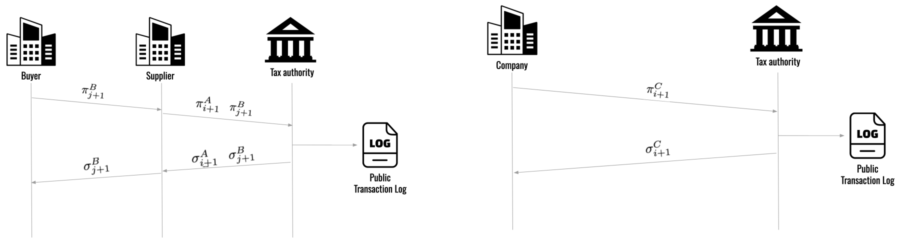
In order to capitalize on this insight, we formalize a VAT reporting system by building on the design introduced in Platypus [WKDC22]. Platypus is a privacy-preserving Central Bank Digital Currency system in which users create privacy preserving currency transfers which are finalized and ordered by the central bank. In contrast, in our VAT reporting system, privacy preserving VAT credit transfers are finalized and ordered by the tax authority.

In more detail, in our reporting system companies can perform 4 different actions in a given VAT period:

1. Enroll accounts with the tax authority.
2. Request VAT credit from the tax authority.
3. Transfer VAT credit to another company.
4. Return VAT credit to the tax authority.

We now explain the intuition for how this flow enables tax authorities to limit fraud by looking at the example presented in section 1. After Alice and Bob enroll, let's assume they both request an initial 100 EUR of VAT credit. When Alice sells 100 EUR worth of apples to Bob, Bob will demand that Alice transfers 20 VAT credit. When Bob sells 200 EUR worth of bottles to Charlie, no VAT credit is sent, as Charlie is not a registered company. When the period closes, all VAT credit must be returned to the tax authority. Compared to the amounts requested:

– Alice has a **deficit** of 20 VAT credit, meaning she must remit 20 EUR to the tax authority. We do not rely on Alice's honesty because **it is in Bob's rational interest to obtain VAT credit**.
– If Bob is honest, he will have a **deficit** of 20 VAT credit, meaning he has to remit 20 EUR to the tax authority. If Bob is dishonest, he might also return the 40 VAT credit which was supposed to be linked to Charlie's purchase, and even end up with a **surplus**. We describe additional mechanisms to enforce Bob to be honest in section 4.1.



(a) VAT credit transfer flow

(b) Enrollment, VAT credit request and VAT credit return flow

Fig. 2. Flow of proofs and signatures for all transaction types in the protocol.

3 Protocol Description

We now give a high-level description of the protocol. We present a more formal fully rolled out protocol in section 6.2 and a description of how much this scheme can limit fraud in section 6.3. For more background and security definitions, we also refer the reader to Platypus [WKDC22].

In order for a company to perform enrollment, VAT credit requests, VAT credit transfers and VAT credit returns without revealing confidential information, the following general recipe is used: 1. create a hiding and binding commitment to private information 2. create a zero-knowledge proof which privately attests that the committed data adheres to the desired properties. 3. after checking the proof, the tax authority signs and publishes the commitment.

There are three types of commitments which companies create:

- **account state commitment** $state_i^C$, which commits to a company's current valid VAT credit balance, as well as a private unique serial number $serial_i^C$.
- **request commitment** req_i^C, which commits to a VAT credit amount to request.
- **transaction commitment** $comm_{tx}$, which commits to a VAT credit amount to transfer between two companies.

3.1 Company enrollment

During enrollment, companies create an initial account state commitment and request commitment, and prove their initial committed balance and requested amount are set to 0.

3.2 VAT credit requests

Companies can ask the tax authority to 'mint' VAT credit for them. A company will want to request enough VAT credit to fulfill all of their VAT credit transfer needs in a given period. However, the total requested amount must stay below a designated maximum req_{max}^C. The enables the tax authority to limit fraud, see section 6.3 for a discussion.

A company can request $\mathbf{X}$ amount of VAT credit by sending a previous account serial number $serial_i^C$, a new account state commitment $state_{i+1}^C$ and a zero knowledge proof to the tax authority. The proof must attest to:

- The correct creation of a new account state commitment. Meaning, $serial_i^C$ was present in a previously signed account state commitment $state_i^C$ with private balance $\mathbf{B_i}$ and $state_{i+1}^C$ commits to private balance $\mathbf{B_i} + \mathbf{X}$.
- The correct creation of a new request commitment. Meaning, the existing $comm_{req}^C$ commits to value $\mathbf{R_i}$, the total requested amount $\mathbf{R_i} + \mathbf{X}$ is less than req_{max}^C and is committed to in the new request commitment $comm_{req}^C$.

The tax authority checks the proof and uniqueness of the serial number, and returns a signature over the company's new state and request commitment.

3.3 VAT credit transfers

If Bob's company buys a product with VAT, they will be entitled to receive VAT credit from seller Alice. Recall it is in Bob's rational interest to obtain the VAT credit, he can return any surplus to the tax authority in exchange for fiat currency.

$\mathbf{X}$ amount of VAT credit can be transferred by *both* companies sending a previous account serial number $\{serial_i^C\}_{C\in\{A,B\}}$, a new account state commitment $\{state_{i+1}^C\}_{C\in\{A,B\}}$, zero knowledge proofs and a transaction commitment $comm_{tx}$ to the tax authority. The proofs must attest to:

– The correct creation of the new account state commitments. Meaning, the numbers $\{serial_i^C\}_{C\in\{A,B\}}$ were present in previously signed account state commitments$\{state_{i+1}^C\}_{C\in\{A,B\}}$ with private balances $\mathbf{A_i}$ and $\mathbf{B_i}$ respectively. $comm_{tx}$ commits to private amount $\mathbf{X}$, $state_{i+1}^B$ commits to private balance $\mathbf{B_i}+\mathbf{X}$ and $state_{i+1}^B$ commits to private balance $\mathbf{A_i}-\mathbf{X}$.

The tax authority checks the proof and uniqueness of the serial numbers, and sends back signatures over the new state commitments.

We model these transfers as being buyer-initiated instead of supplier-initiated, because only the buyer of a product or service will always know who the counterparty is. Think of the case where a buyer's receipt is processed by a corresponding finance department far past the date of purchase.

3.4 VAT credit returns

When a VAT period ends, companies return any remaining VAT credit to the tax authority. As mentioned in the example presented in section 1, when more VAT credit is returned than was requested, the surplus is awarded to the company in fiat currency. When less VAT credit is returned than was requested, the deficit must be paid for in fiat currency.

A company can return VAT credit by sending a previous account serial number $serial_i^C$, values $\mathbf{X}$ and $\mathbf{U}$, and a zero knowledge proof to the tax authority. The proof must attest to:

– $serial_i^C$ was present in a previously signed account state commitment $state_i^C$ with public balance $\mathbf{X}+\mathbf{U}$.

Additionally, the company must prove how much VAT credit $\mathbf{R}$ they requested, by revealing the latest $comm_{req}^C$'s blinding value.

The tax authority checks the proof and uniqueness of the serial number, and sends back a signature over returned amounts $\mathbf{X}$ and $\mathbf{U}$. The tax authority can determine the size of a company's VAT credit surplus or deficit by comparing their requested VAT credit $\mathbf{R}$ and their returned VAT credit $\mathbf{X}$. In section 6.3 we show this scheme is sufficient to ensure that the VAT credit returns of honest companies limit the amount of fraud which any dishonest companies can misappropriate.

Note how the company also returns unclaimed amount $\mathbf{U}$. This voluntarily indicates the total amount of VAT credit which was technically claimable by counterparties, but never was. A common scenario is when this VAT credit was part of a B2C sale. See section 4.1 for a discussion on enforcing honest reporting of VAT credit.

4 Discussion

4.1 Real world deployment concerns

Interactivity and concurrency. The transaction model used in this paper to transfer VAT credit requires 1 round of interaction between buyer and supplier. We now discuss whether this is feasible for a real-world deployment.

One concern one might have is that being always online is a hard guarantee to make, even when making use of intermediaries. But this is not a major concern in the context of VAT, as there is no need for VAT credit transfers to finalize in real-time - as long as they happen before the end of the VAT period.

Another concern is that the interactivity requirement may limit throughput, as an account can only start to process a new transaction when the previous one has finalized. To see why this is an issue, consider the edge case where Alice and Bob send VAT credit to *each other* concurrently. They will both be waiting for a tax authority signature before completing the other's transaction. Moreover, all of Alice and Bob's current buyers will *also* be waiting for the transaction to complete. In theory such deadlocks can be resolved by letting outstanding transactions expire. A faster resolution would entail companies having at least two accounts: one for sending and one for receiving VAT credit. Not only does this prevent such deadlocks, if Alice and Bob's buyers are waiting for them to respond, at least they can still send VAT credit to others. To further increase throughput, one can increase the number of sending or receiving accounts.

Approaches to limit fraud Although Verifiable VAT limits the maximum amount of fraud (see section 6.3), companies can still disappear and leave their VAT credit deficit unpaid. A tax authority can limit companies from disappearing with too much requested VAT credit by:

1. Limiting the total amount of VAT credit given (for free).
2. Not paying for surplus unless the full or partial deficit of company's direct suppliers has been paid for.
3. Checking that companies correctly indicate when VAT credit is unclaimed (implying a transaction occurred at the end of the value chain). This can be done with B2C spot checks by the tax authority, enforced use of secure hardware, or even incentivising consumers to check that receipts are registered using lotteries. These approaches are common practice in various countries around the world today.

Key management. The cryptographic primitives used in Verifiable VAT require companies to do some form of key management, a seemingly high requirement. The burden can be alleviated by making use of existing software and hardware which companies already entrust with their data. More specifically, hosted bookkeeping software could offer the service to maintain keys and interact with other parties. Some form of insurance may also additionally be possible to protect against the risk of losing ones VAT credit keys.

Scalability. Platypus describes how their registration system's overhead is small enough to plausibly be used for all payments [WKDC22]. An important part of their reasoning is that their system is horizontally scalable. The tax authority can split workloads by company id or the most significant bits of the serial number. The transaction types introduced in this paper are similar in size and complexity as Platypus, and the number of transactions with VAT is a strict subset of all transactions. We therefore conclude that scalability should not be a concern.

4.2 Future work

We leave the reader with several directions for future work.

– Can the maximum VAT credit deficit be bounded by a lower amount than indicated in section 6.3? One additional tool which can be used is to only allow VAT credit to be spent when it has been received directly from the tax authority.
– Can Verifiable VAT be implemented (efficiently) in a UTXO-based transaction model, whereby there are no account state commitments and there is no interactivity requirement?
– Can the Proof of Interaction described in section 6.4 be implemented (efficiently) in such a way that the companies which are "blacklisted" are not revealed?
– How much can prover time, proof size and verifier time of the designed scheme be reduced? Some strategies include: making use of better proving systems, combining (commit-and-prove) proving systems, using more efficient arithmetization and arithmetization-friendly primitives.

4.3 Related Work

CDBC / e-cash. Central Bank Digital Currency literature comes closest to the Verifiable VAT design, as this literature also aims to efficiently realize private permissioned e-cash [WKDC22]. An interesting avenue some of these systems take is to use multi-party computation to allow for conditionally revealing data to authenticated stakeholders [KKS22] [TBA+22].

Tokenized VAT. Several papers have been published on tokenizing VAT [AAC16] [AAC20], which have not presented rigorous arguments about how fraud is prevented or how transaction data is kept private. Critically, these proposals fundamentally alter how VAT works by introducing a new non-tradeable currency to pay VAT in.

5 Acknowledgments

We thank Kari Kostiainen, Sheila Zingg, Nicolas Mohnblatt, Ben Wilson, Georg Wiese, Hamish Ivey-Law and Andrew Zitek-Estrada for helpful discussions.

References

AAC16. Richard Thompson Ainsworth, Musaad Alwohaibi, and Mike Cheetham. Vatcoin: the gcc's cryptotaxcurrency. *Boston Univ. School of Law, Law and Economics Research Paper*, (17-04), 2016.

AAC20. Richard Thompson Ainsworth, Musaad Alwohaibi, and Mike Cheetham. Uk & ksa vats: A cutting-edge proposal–mini-blockchain and vatcoin. *Available at SSRN 3574381*, 2020.

AR20. Shashank Agrawal and Srinivasan Raghuraman. Kvac: key-value commitments for blockchains and beyond. In *Advances in Cryptology–ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part III 26*, pages 839–869. Springer, 2020.

ark. URL: `https://github.com/arkworks-rs/groth16/tree/8e5c347bd8776645e046ca7ec1e4b9ff4b97c054`.

CCDW20. Weikeng Chen, Alessandro Chiesa, Emma Dauterman, and Nicholas P Ward. Reducing participation costs via incremental verification for ledger systems. *Cryptology ePrint Archive*, 2020.

Com18. European Commission. Vat: Eu member states still losing almost €150 billion in revenues according to new figures, 2018. URL: `https://ec.europa.eu/commission/presscorner/detail/en/IP_18_5787`.

con23. Wikipedia contributors. Transaction-based reporting. in wikipedia, the free encyclopedia., 2023. URL: `https://en.wikipedia.org/w/index.php?title=Transaction-Based_Reporting&oldid=1178411174`.

KKS22. Aggelos Kiayias, Markulf Kohlweiss, and Amirreza Sarencheh. Peredi: Privacy-enhanced, regulated and distributed central bank digital currencies. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 1739–1752, 2022.

KL14. Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Chapman and Hall/CRC, 2nd edition, 2014.

KST22. Abhiram Kothapalli, Srinath Setty, and Ioanna Tzialla. Nova: Recursive zero-knowledge arguments from folding schemes. In *Annual International Cryptology Conference*, pages 359–388. Springer, 2022.

Mur19. R Murphey. The european tax gap. commissioned by the socialists and democrats party, 2019. URL: `https://www.socialistsanddemocrats.eu/sites/default/files/2019-01/`.

Nov. URL: `https://github.com/microsoft/Nova/tree/79de586b1e61caabbc7ad1854d6dec41f56313d7`.

ROO23. Max Roser and Esteban Ortiz-Ospina. Taxation, 2023. URL: `https://ourworldindata.org/taxation`.

TBA$^+$22. Alin Tomescu, Adithya Bhat, Benny Applebaum, Ittai Abraham, Guy Gueta, Benny Pinkas, and Avishay Yanai. UTT: Decentralized ecash with accountable privacy. Cryptology ePrint Archive, Paper 2022/452, 2022. `https://eprint.iacr.org/2022/452`. URL: `https://eprint.iacr.org/2022/452`.

TFZ$^+$22. Nirvan Tyagi, Ben Fisch, Andrew Zitek, Joseph Bonneau, and Stefano Tessaro. Versa: Verifiable registries with efficient client audits from rsa authenticated dictionaries. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 2793–2807, 2022.

WKDC22. Karl Wüst, Kari Kostiainen, Noah Delius, and Srdjan Capkun. Platypus: A central bank digital currency with unlinkable transactions and privacy-preserving regulation. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, page 2947–2960, New York, NY, USA, 2022. Association for Computing Machinery. `doi:10.1145/3548606.3560617`.

6 Appendix

6.1 Algorithms

To make VAT verifiable, we make black-box use of the following cryptographic algorithms. We informally describe their interface, and refer the interested reader to the respective sources for more in-depth definitions and background. We assume all algorithms are instantiated with a security parameter λ. Similarly, any generated public parameters are included implicitly in all algorithms.

Commitment Scheme (CS)
A hiding and binding **CS** is a tuple of algorithms (**Setup**, **Commit**) adopted from [KL14].

- **Setup** outputs public parameters params.
- **Commit** takes as input a message m and randomness r and outputs a commitment $comm$.

Pseudo-Random Function (PRF)
A **PRF** is a tuple of algorithms (**Init**, **Gen**) adopted from [KL14].

- **Init** takes as input a domain separator d and randomness r and outputs an initial state st_0.
- **Gen** takes as input state information st_i, and outputs bits y and updated state st_{i+1}.

Digital Signature Scheme (DS)
A **DS** is a tuple of algorithms (**Generate**, **Sign**, **Verify**) adopted from [KL14].

- **Generate** outputs a pair of keys (pk, sk). We refer to the first of these as the public key and the second as the secret key.
- **Sign** takes as input a secret key sk and a message m, and outputs a signature σ.
- **Verify** takes as input a public key pk, a message m, and a signature σ. It outputs a bit b indicating whether or not σ is valid.

Authenticated Dictionary (AD) Scheme
An **AD** is a tuple of algorithms (**Setup**, **Init**, **Update**, **Lookup**, and **Verify-Lookup**), adopted with slight modifications from [TFZ$^+$22].

- **Setup** outputs public parameters.
- **Init** takes as input a set of keys and values $[(k_j, v_j)]$ and outputs an initial digest d_0.
- **Update** takes as input a digest d_i, a key k, and a delta δ. It outputs an updated digest d_{i+1}. In our construction, this *increments* the stored value by δ, i.e., $d_{i+1}[k] = d_i[k] + \delta$.

- **Lookup** takes as input a key k. It outputs a value v and proof Λ_k.
- **Aggregate** takes as input a set of proofs Λ_k. It outputs a single proof Λ.
- **VerifyLookup** takes as input a digest d, a set of keys and values $[key, value]_k$ and proof Λ. It outputs a bit b indicating whether or not Λ is valid.

Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (zk-SNARK) Scheme
A **zk-SNARK** is a tuple of algorithms (**Generate**, **Prove**, **Verify**) adopted with slight modifications from [KST22].

- **Generate** takes as input a structure s, and outputs a proving and verifying key (pk, vk).
- **Prove** takes as input the proving key pk, an instance x, and a witness w, and outputs a proof π.
- **Verify** takes as input the verifying key vk, an instance x, and a proof π. It outputs a bit b indicating whether or not π is valid.

Incrementally-Verifiable Computation Scheme (IVC)
An **IVC** is a tuple of algorithms (**Generate**, **Prove**, **Verify**) adopted from [KST22].

- **Generate** takes as input a statement s, and outputs a proving and verifying key (pk, vk).
- **Prove** takes as input the proving key pk, index i, public inputs z_0 and z_i, witness w_i and a proof Π_i. It outputs a new proof Π_{i+1}.
- **Verify** takes as input the verifying key vk, index i, public inputs z_0 and z_i and a proof Π_i. It outputs a bit b indicating whether or not Π_i is valid.

6.2 Verifiable VAT protocol

We now present a rolled-out protocol of the full scheme, including the Proof of Interaction.

System Setup To set up the system, the tax authority:

- runs **DS.Generate** to create a secret/public key pair (sk_T, pk_T) that is used for signing account state commitments and publishes its public key pk_T.
- sets bal_{max} which is a maximum limit on account balances to prevent value overflows (i.e. since all values are finite field elements, it ensures that balances cannot be negative) and can be set to a value larger than all realistic values for account balances.
- sets req^C_{max} which is the maximum amount of VAT credit companies are allowed to request.

Both the tax authority and companies run:

- $\mathbf{IVC_{POI}.Generate}$
- $\textbf{zk-SNARK}_{enrollment}\textbf{.Generate}$
- $\textbf{zk-SNARK}_{request}\textbf{.Generate}$
- $\textbf{zk-SNARK}_{transfer,send}\textbf{.Generate}$
- $\textbf{zk-SNARK}_{transfer,receive}\textbf{.Generate}$
- $\textbf{zk-SNARK}_{return}\textbf{.Generate}$
- **CS.Setup**

Company enrollment When a company C enrolls in the system, they create a secret key $sk_C = (sk_{C1}, sk_{C2})$, consisting of two randomly chosen keys. These keys can later be used to pseudorandomly derive serial numbers and blinding values for their account states using pseudorandom functions $\mathbf{PRF}_{serial,C} = \mathbf{PRF.Init}('serial', sk_{C1})$ and $\mathbf{PRF}_{blind,C} = \mathbf{PRF.Init}('blind', sk_{C1})$. The company also needs a unique identifier id_C, which could be for example their VAT number. To create an enrollment transaction, C derives the pseudorandom values $serial_1^C = \mathbf{PRF}_{serial,C}.\mathbf{Gen}(0)$, $blind_1^C = \mathbf{PRF}_{blind,C}.\mathbf{Gen}(0)$ and uses them to create a new state commitment. C then runs $\textbf{zk-SNARK}_{enrollment}\textbf{.Prove}$ to create $\pi_{enrollment}$ which proves the following statement:

Given public values $\mathbb{X} = (state_1^C,\ req_1^C,\ id_C)$,
I know secret values $\mathbb{W} = (sk_1^C, serial_1^C, blind_1^C)$, such that
$state_1^C == \mathbf{CS}.\mathbf{Commit}(serial_1^C, 0, id_C, blind_1^C)$ $req_1^C == \mathbf{CS}.\mathbf{Commit}(0, blind_{req}^C)$ $serial_1^C == \mathbf{PRF}_{serial,C}.\mathbf{Gen}(0)$

C then sends $\mathbb{X}$ and $\pi_{enrollment}^C$ to the tax authority. The tax authority checks that the check $\textbf{zk-SNARK}_{enrollment}\textbf{.Verify}(\pi_{i+1}^C, \mathbb{X})$ passes. If the proof is correct, the tax authority runs $\mathbf{DS}.\mathbf{Sign}(sk_T, state_1^C, req_1^C)$ and returns the signature back to C.

VAT credit requests If companies want to request additional credits, they can create a one-sided transaction as follows.

1. Transaction Initiation

They runs $\textbf{zk-SNARK}_{request}\textbf{.Prove}$ to create π_{i+1}^C which proves the following statement:

Given public values $\mathbb{X} = (serial_i^C, comm_{Tx}, state_{i+1}^C, bal_{max}, pk_T, comm_{req,in}, id_{receiver}, comm_{req,out})$,
I know secret values $\mathbb{W} = (sk_{C1}, bal_i^C, bal_{i+1}^C, blind_i^C, \sigma_i^C, req_{in}, req_{out}, v_{Tx}, blind_{Tx}, serial_{i+1}^C, blind_{i+1}^C, i)$, such that
$1 == \textbf{DS}.\textbf{Verify}(pk_T, state_i^C, comm_{Tx,i-1}^C, \sigma_i^C)$ $comm_{Tx} == \textbf{CS}.\textbf{Commit}(v_{Tx}, 0, id_{receiver}, i, blind_{Tx})$ $state_i^C == \textbf{CS}.\textbf{Commit}(serial_i^C, bal_i^C, id_{receiver}, i, blind_i^C)$ $state_{i+1}^C == \textbf{CS}.\textbf{Commit}(serial_{i+1}^C, bal_{i+1}^C, id_{receiver}, i+1, blind_{i+1}^C)$ $bal_{max} \geq bal_{i+1}^C$ $bal_{i+1}^C := bal_i^C + v_{Tx}$ $serial_{i+1}^C == \textbf{PRF}_{serial,C}.\textbf{Gen}(serial_i^C)$ $comm_{req,in} == \textbf{CS}.\textbf{Commit}(req_{in}, blind_{req})$ $req_{out} == req_{in} + v_{Tx}$ $req_{max} \geq req_{out}$ $comm_{req,out} == \textbf{CS}.\textbf{Commit}(req_{out}, blind_{req})$

The company then sends $comm_{Tx}, serial_i^C, state_{i+1}^C, \pi_{i+1}^C, comm_{req,out}$ to the tax authority.

2. Transaction Execution

The tax authority checks that the serial number is unique and that the check $\textbf{zk-SNARK}_{request}.\textbf{Verify}(\pi_{i+1}^C, \mathbb{X})$ passes. If so, it publishes the transaction and signs and returns the new state and transaction commitment: $\sigma_{i+1}^C = \textbf{DS}.\textbf{sign}(sk_T, comm_{Tx}, state_{i+1}^C)$. It also replaces the company's request commitment $comm_{req,in}$ by $comm_{req,out}$.

3. Acceptance

The company checks that the signature is valid, and if so, stores it.

VAT credit transfers

Let's look at the example from section 1, Alice transfers VAT credit to Bob.
1. Transaction Initiation: Bob runs $\textbf{zk-SNARK}_{transfer,receive}.\textbf{Prove}$ to create π_{i+1}^B which proves the following statement:

Given public values $\mathbb{X}_B = (serial_i^B, comm_{Tx}, state_{i+1}^B, bal_{max}, pk_T)$,
I know secret values $\mathbb{W}_B = (sk_{B1}, bal_i^B, bal_{i+1}^B, blind_i^B, \sigma_i^B, v_{Tx}, blind_{Tx}, serial_{i+1}^B, blind_{i+1}^B, id_{sender}, id_{receiver}, i)$, such that
$1 == \textbf{DS}.\textbf{Verify}(pk_T, state_i^B, comm_{Tx,i-1}^B, \sigma_i^B)$ $comm_{Tx} == \textbf{CS}.\textbf{Commit}(v_{Tx}, id_{sender}, id_{receiver}, i, blind_{Tx})$ $state_i^B == \textbf{CS}.\textbf{Commit}(serial_i^B, bal_i^B, id_{receiver}, i, blind_i^B)$ $state_{i+1}^B == \textbf{CS}.\textbf{Commit}(serial_{i+1}^B, bal_{i+1}^B, id_{receiver}, i+1, blind_{i+1}^B)$ $bal_{max} \geq bal_{i+1}^B$ $bal_{i+1}^B == bal_i^B + v_{Tx}$ $serial_{i+1}^B == f_{sk_{A1}}(serial_i^B)$

Bob then sends $v_{Tx}, blind_{Tx}, serial_i^B, comm_{Tx}, state_{i+1}^B$ and π_{i+1}^B to Alice.

2. Transaction Completion
Alice runs **zk-SNARK**$_{transfer,sender}$.**Prove** to create π_{i+1}^A which proves the following statement:

Given public values $\mathbb{X}_A = (serial_j^A, comm_{Tx}, state_{j+1}^A, bal_{max}, pk_T)$,
I know secret values $\mathbb{W}_A = (sk_{A1},\ bal_j^A,\ bal_{j+1}^A,\ blind_j^A,\ \sigma_j^A,\ v_{Tx},\ blind_{Tx},\ serial_{j+1}^A,$ $blind_{j+1}^A,\ id_{sender},\ id_{receiver},\ j)$, such that
$1 == \mathbf{DS}.\mathbf{Verify}(pk_T, state_j^A, comm_{Tx,j-1}^A, \sigma_j^A)$ $comm_{Tx} == \mathbf{CS}.\mathbf{Commit}(v_{Tx}, id_{sender}, id_{receiver}, j, blind_{Tx})$ $state_j^A == \mathbf{CS}.\mathbf{Commit}(serial_j^A, bal_j^A, id_{sender}, j, blind_j^A)$ $state_{j+1}^A == \mathbf{CS}.\mathbf{Commit}(serial_{j+1}^A, bal_{j+1}^A, id_{sender}, j+1, blind_{i+1}^A)$ $bal_j^A \geq v_{Tx}$ $bal_{j+1}^A == bal_j^A - v_{Tx}$ $serial_{i+j}^A == f_{sk_{A1}}(serial_j^A)$

Alice then sends $comm_{Tx}, serial_i^A, state_{i+1}^A, \pi_{i+1}^A, serial_j^B, state_{j+1}^B, \pi_{j+1}^B$ to the tax authority.

3. Transaction Execution
The tax authority checks that the serial numbers are unique and that the check for Alice **zk-SNARK**$_{transfer,sender}$.**Verify**$(\pi_{j+1}^A, \mathbb{X}_A)$ as well as the check for Bob **zk-SNARK**$_{transfer,receiver}$.**Verify**$(\pi_{i+1}^B, \mathbb{X}_B)$ both pass. If so, it publishes the transaction and signs and returns the new state commitments and transaction commitment:

- $\sigma_{i+1}^A = \mathbf{DS}.\mathbf{sign}(sk_T, comm_{Tx}, state_{i+1}^A)$.
- $\sigma_{j+1}^B = \mathbf{DS}.\mathbf{sign}(sk_T, comm_{Tx}, state_{j+1}^B)$.

4. Payment Acceptance
Alice checks that the signatures are valid, and if so, stores their own signature and forwards Bob's signature back.

5. Payment Completion
Bob checks that the signature is valid, and if so, stores it. If they have not received a signature yet after a time-out, Bob scans the public transaction log for the signature. If it is not present, Bob creates a transaction to request 0 VAT credit, in order to invalidate the previous incomplete transaction.

VAT credit returns If companies want to return their VAT credit, they can create a one-sided transaction.
1. Transaction Initiation

They runs $\textbf{zk-SNARK}_{return}\textbf{.Prove}$ to create π^C_{i+1} which proves the following statement.

Given public values $\mathbb{X} = (serial^C_i,\ comm_{Tx},\ state^C_{i+1},\ bal_{max},\ pk_T,\ comm_{req,in},\ i,\ bal^C_{i+1},\ X,\ U,\ comm_{req,out},\ id_C)$,
I know secret values $\mathbb{W} = (sk_{C1},\ bal^C_i,\ blind^C_i,\ \sigma^C_i,\ blind_{Tx},\ serial^C_{i+1},\ blind^C_{i+1})$, such that
$1 == \textbf{DS}.\textbf{Verify}(pk_T, state^C_i, comm^C_{Tx,i-1}, \sigma^C_i)$ $comm_{Tx} == \textbf{CS}.\textbf{Commit}(X + U, id_C, 0, i, blind_{Tx})$ $state^C_i == \textbf{CS}.\textbf{Commit}(serial^C_i, bal^C_i, id_C, i, blind^C_i)$ $state^C_{i+1} == \textbf{CS}.\textbf{Commit}(serial^C_{i+1}, bal^C_{i+1}, id_C, i+1, blind^C_{i+1})$ $bal^C_i \geq (X + U)$ $bal^C_{i+1} == bal^C_i - (X + U)$ $serial^C_{i+1} == \textbf{PRF}_{serial,C}.\textbf{Gen}(serial^C_i)$

The company then sends $comm_{Tx}, serial^C_i, state^C_{i+1}, \pi^C_{i+1}, i, bal^C_{i+1}, X, U$ to the tax authority.

2. Transaction Execution
The tax authority checks that the serial number is unique and that the check $\textbf{zk-SNARK}_{return}\textbf{.Verify}(\pi^C_{i+1}, \mathbb{X})$ passes. If so, it publishes the transaction and signs and returns the new state and transaction commitment: $\sigma^C_{i+1} = \textbf{DS}.\textbf{sign}(sk_T, comm_{Tx}, state^C_{i+1})$.

3. Acceptance
The company checks that the signature is valid, and if so, stores it.

6.3 Verifiable VAT Security Analysis

Given the similarity of our model with Platypus, most of their security analysis holds also for our model. We introduce a couple of extensions and enhancements where we introduce significant changes.

Transaction Integrity Claim (Balance Invariance) *No computationally bounded adversary without access to the simulation trapdoor of the zero-knowledge proof system can create a transaction that increases the available funds in the system or spends funds more than once.*

In contrast to Platypus, the proof sketch must also take into account that companies can formally request up to req_{max} from the tax authority.

Proof extension. Consider the case where an adversary attempts to create a credit request transaction that increases the account balance of the sender by more than the global parameter req_{max}. Since the value requested is committed to using the transaction commitment $comm_{Tx}$, which is created using a hiding and binding commitment scheme, no computationally bounded party can open the commitment to a transaction value other than what was committed to originally. Since the proof of the transaction sender proves that their account balance was increased by exactly the committed value, any adversary that could increase the sender's balance by a different value could be used to either break soundness of the zk-SNARK proof system or to break the binding property of the commitment scheme. Moreover, the same proof system allows a user to attest that the sum of requested values thus far (committed to in $comm_{req}$), stays below req_{max}. Because we assume the use of a secure authenticated broadcast channel, the tax authority is ensured a company only uses and updates their own request commitment. □

Bounds on maximum amount of fraud

Claim *Total VAT credit surplus is bounded by total VAT credit deficit and total unclaimed VAT credit*

We can distinguish between VAT credit returns which result in a surplus S_i, or a deficit D_i of VAT credit. The maximum deficit D_i of each company equals the maximum VAT credit they can request from the tax authority: req_{max}. Recall that the deficit and surplus respectively signal that a company has to pay VAT to the tax authority or is eligible to receive back VAT. Call the total surplus $S = \sum_i S_i$ and the total deficit $D = \sum_i D_i$. Tax authority revenue is non-zero if $S < D$.

Recall also that there is VAT credit $U = \sum_i U_i$ which companies report to be unclaimed - these were claimable in a real-world transaction by a counterparty, but never were, potentially because the counterparty was a consumer.

We can prove that $S \leq D - U$ as follows:

- First assume a non-zero U and a single firm $i = 1$. There is no surplus S and the VAT deficit, D_1 must equal U_1.
- Next, if any new firm sends a non-zero amount A of VAT credit to an existing firm, e.g. firm $i = 1$, then either:
  - A is less than or equal to D_1, total S, D and U remain constant
  - A is bigger than D_1, total S and D will grow equally.
- If any firm has additional unclaimed VAT credit, this must either increase total D or decrease total S.

We discuss methods to further bound fraud in section 4.1.

6.4 Proof of Interaction Protocol

We demonstrate an extension to Verifiable VAT which is of unique use to tackle VAT fraud. We first highlight the intuitions, after which we demonstrate the rolled out protocol additions.

Companies might not pay their VAT credit deficit and try to disappear. One response to this is for the tax authority to put the disappeared companies on a blacklist, so companies can prove whether or not they interacted with them. It's very useful information for the tax authority to know who the buyer is, because the VAT credit receiver might be collaborating with a fraudulent supplier in order to siphon VAT from the tax authority. Proof of interaction with blacklisted parties can provide grounds for audits or even reduction of VAT to get in return for a VAT credit surplus.

A first naïve attempt to achieve this could involve each company maintaining an accumulator which encodes a mapping of suppliers and buyers to the amount of VAT credit received and sent respectively. At the end of the period, Companies share the proof that a certain (key, value) pair was included with the tax authority. However, this construction has two issues:

1. The tax authority has no guarantee that the accumulator has actually been updated with all or even any of the company's confirmed VAT credit transfers.
2. Accumulator constructions are typically not guaranteed to be hiding [TFZ$^+$22], [AR20], [CCDW20], so a company risks leaking the amount of VAT exchanged with non-blacklisted trading partners.

To overcome these issues:

1. Companies should generate a zk-SNARK which attests to the fact that updates to the accumulator were made using all of a company's signed transaction commitments.
2. Companies should not share the final accumulator directly, but rather prove the value that it opens to for given blacklisted companies, keeping the accumulator as private witness inputs.

Ideally, such a proof should incur minimal proving or communication overhead. A natural candidate to prove and verify this efficiently is to leverage Incrementally Verifiable Computation (IVC) using e.g. folding schemes [KST22]. This allows a prover to produce a succinct proof of repeated invocation of the same function. Note that folding schemes by themselves reveal the public function input and output to the verifier [KST22]. Therefore, to avoid data leakage we must ensure that the input and output of each IVC step is a hiding commitment to the accumulator.

To make the scheme secure, we require that both state and transaction commitments need to include:

– A transaction counter, which increments with each new transaction. This will allow for tracking that all transactions are included in the Proof of Interaction.
– Company ids referring to the sending and receiving company (in the case of the tx commitment) or holding company (in the case of the state commitment). This will allow for tracking which counterparties companies transacted with.

We make the following adjustments to the Verifiable VAT protocol.

– **System Setup**: the tax authority generates a public keypair with which they can receive encrypted data.
– **Company Enrollment**: During enrollment, companies must prove they included the correct company id into their initial state commitment. After all companies are enrolled, all companies and the tax authority must initialize the accumulator with the value 0 for each enrolled company.
– **VAT credit transfers and requests**: Companies must prove the same company id is used across state commitments, and that each new state commitment includes an incremented counter. Moreover, the tax authority must sign the transaction commitment. After receiving back a signed transaction, each company proves in one IVC step that a committed accumulator is updated with a unique *signed* transaction commitment's value.
– **VAT credit returns** The final counter, value, remaining balance and company id must be proven and revealed to the tax authority.

After the tax authority shares a blacklist of company ids, companies must create a proof of how much VAT credit was traded with respective blacklisted companies in the final accumulator. For this a company will need to prove knowledge of the fact that:

1. n IVC steps were performed correctly and output a commitment acc_n, where n equals the index revealed during the VAT credit return.
2. There exists a valid proof such that acc_n is an accumulator commitment to a set of (sent and received) values (to and from) the blacklisted company ids.

Rolled out protocol: Accumulating interactions Both the tax authority and companies need to run **AD.Setup**. After all companies are enrolled, companies and the tax authority run AD_{init} = **AD.init**, inserting for all registered company identifiers the value 0.

Whenever a company requested or transferred VAT credit, they call **IVC.Prove** for the following statement:

Given public values $\mathbb{X}_{IVC,i} = (i, pk_T, Comm_{AD_i}, Comm_{AD_{i+1}}, id_C)$,
I know secret values $\mathbb{W}_{IVC,i} = (\sigma, state_i, comm_{Tx}, blind_{tx}, blind_{AD}, v_{Tx}, id_{sender}, id_{receiver}, upd_i)$, such that

$\textbf{DS.Verify}(state_i, comm_{Tx}) == true$
$comm_{Tx} == \textbf{CS.Commit}(v_{Tx}, id_{sender}, id_{receiver}, index, blind_{Tx})$
$index == i$
$is\_sender := id_C == id_{sender}$
$is\_receiver := id_C == id_{receiver}$
$is\_sender \vee is\_receiver$
$counterparty_i := is\_sender ? id_{receiver} : id_{sender}$
$Comm_{AD_i} == \textbf{CS.Commit}(AD_i, blind_{AD})$
$(i == 1 \wedge AD_i == AD_{init}) \vee true$
$(AD_{i+1}, upd_i) = \textbf{AD.Update}(AD_i, (counterparty_i, is\_sender), v_{Tx})$
$Comm_{AD_{i+1}} == \textbf{CS.Commit}(AD_{i+1}, blind_{AD})$

Rolled out protocol: Proving interactions In theory, a Proof of Interaction could be submitted with every new transaction. For simplicity, we describe a scenario where a Proof of Interaction is required just once. If the tax authority determines that a Proof of Interaction is required after n transactions regarding a certain m possible counterparties, companies and tax authority run:

– **zk-SNARK$_{\textbf{IVC-n}}$.Generate**
– **zk-SNARK$_{\textbf{POI-m}}$.Generate**

Companies run for each txtype and for each company K in the blacklist **B**, $\Lambda_{K,txtype} =$ **AD.Lookup**$(key_{K,txtype}, val_{K,txtype})$. These can be aggregated into a single proof $\Lambda =$ **AD.Aggregate**$([\Lambda_{C,txtype}]_{K\in\mathbf{B},txtype\in\{send,receive\}})$. If the company itself is in the blacklist, they can exclude this entry.

Companies run **zk-SNARK**$_{IVC-n}$**.Prove** to create π_{IVC} which proves n steps of **IVC$_{\textbf{POI}}$** were run correctly, and have as final output $Comm_{AD_n}$. Companies also run **zk-SNARK**$_{POI-m}$**.Prove** to create π_{POI} which proves the following statement.

Given public values $\mathbb{X}_{POI,m} = (Comm_{AD_n}, [(key, val)_{K,txtype}]_{K\in\mathbf{B},txtype\in\{send,receive\}})$, I know secret values $\mathbb{W}_{POI,m} = (\Lambda, blind_{AD})$, such that
$comm_{AD_n} ==$ **CS.Commit**$(AD_n, blind_{AD})$ $1 ==$ **AD.VerifyLookup**$(\Lambda, Comm_{AD_n}, [(key, val)_{K,txtype}]_{K\in\mathbf{B},txtype\in\{send,receive\}}$

The company then sends π_{IVC}, π_{POI}, $\mathbb{X}_{IVC,n}$ and $\mathbb{X}_{POI,m}$ to the tax authority.

Finally, tax authority will accept the proof if both the following pass:

– **zk-SNARK**$_{IVC-n}$**.Verify**$(\pi_{IVC}, \mathbb{X}_{IVC,n})$
– **zk-SNARK**$_{POI-m}$**.Verify**$(\pi_{POI}, \mathbb{X}_{POI,m})$

6.5 Proof of Interaction Benchmark

In the following, we only implement and benchmark the steps unique to Proof of Interaction. Performance of enrollment and VAT credit transfers/requests/returns are similar to the original enrollment and transaction benchmarks in [WKDC22].

We benchmark the Proof of Interaction scheme using two instantiations of authenticated dictionary accumulators, similar to those presented by [TFZ$^+$22].

– We use a RSA accumulator construction with 100-bit security, originally proposed in [AR20].
– We use a Poseidon-based Sparse Merkle Tree construction with 128-bit security.

We instantiate **IVC** and $\mathbf{zk\text{-}SNARK_{IVC}}$ over BN256 and Grumpkin (which form a curve cycle) and the Nova library [Nov]. We instantiate $\mathbf{zk\text{-}SNARK_{POI}}$ over BN256 using the Arkworks Groth16 library [ark]. We instantiate **PRF** and **Commit** using Poseidon, and **DS** using schnorr.

We assume $bal_{max} = 2^{64}$, and using 32-bit transaction counters and company ids.

Given our existing constructions and trust assumptions, our benchmarks show that merkle trees are more efficient than RSA accumulators for our purposes. Only if the tax authority blacklists tens of thousands of companies, do the asymptotic benefits of batched RSA accumulator proof verification kick in. We briefly list avenues for optimizing both constructions in 4.2.

	Algorithm	# constraints	Prover time (s)	Proof size (bytes)	Verifier time (s)
SMT	$\mathbf{IVC_{POI}}$	18314	0.1	n.a.	n.a.
	$\mathbf{zk\text{-}SNARK_{IVC\text{-}1}}$	n.a.	0.1	10264	0.03
	$\mathbf{zk\text{-}SNARK_{IVC\text{-}100}}$	n.a.	0.7	10639	0.03
	$\mathbf{zk\text{-}SNARK_{POI\text{-}1}}$	38853	0.2	200	0.2
	$\mathbf{zk\text{-}SNARK_{POI\text{-}100}}$	753099	2.1	200	0.2
RSA	$\mathbf{IVC_{POI}}$	4143228	2.3	n.a.	n.a.
	$\mathbf{zk\text{-}SNARK_{IVC\text{-}1}}$	n.a.	13.5	12774	0.2s
	$\mathbf{zk\text{-}SNARK_{IVC\text{-}100}}$	n.a.	14.5	13151	0.2s
	$\mathbf{zk\text{-}SNARK_{POI\text{-}1}}$	10484634	44	200	0.2
	$\mathbf{zk\text{-}SNARK_{POI\text{-}100}}$	10484634	44	200	0.2

Table 1. Concrete efficiency for both the Sparse Merkle Tree and RSA Key-Value Accumulator constructions to run various operations. (1) $\mathbf{IVC_{POI}}$ to prove insertion of a company's transaction value into their accumulator (2) $\mathbf{zk\text{-}SNARK_{IVC\text{-}n}}$ to prove and verify compression of n-step $\mathbf{IVC_{POI}}$ (3) $\mathbf{zk\text{-}SNARK_{POI\text{-}m}}$ to prove and verify the sent and received amounts from n random blacklisted counterparties. Tested on M2 Max, 12 cores and 32 GiB RAM.

6.6 Proof of Interaction Security Analysis

Proof of Interaction Integrity and Privacy We now discuss the integrity and privacy of the Proof of Interaction scheme.

Claim: *No company can create a valid Proof of Interaction such that their transacted amounts with blacklisted counterparties are not revealed to the tax authority.*

Working our way backwards, for a succeeding proof, assuming a sound Authenticated Dictionary (AD), commitment scheme and zk-SNARK, companies can only cheat by using a value of AD_n which does not actually reflect their accumulated state. There are several ways in which they can try to achieve this: 1. The companies does not start their AD as AD_{init}, as all parties compute it during enrollment 2. The company does not verifiably update the AD with some or any of their transfers 3. A verified update to the AD does not reflect the state of a confirmed transfer 4. The AD passed into an IVC step is not equal to the AD passed into the previous IVC step.

1. AD_{init} is a public value, checked by the verifier in the first step of IVC (which is proven in $\mathbf{zk\text{-}SNARK_{IVC}}$). 2. The zkp in each credit request and transfer checks that the new state includes a counter one above the counter of the previous state. In turn, each IVC step in the Proof of Interaction checks that the counter equals the IVC step index. Finally the verifier checks that the final IVC step counter equals the counter in the final state during the VAT credit return. So the company cannot exclude any transfer. 3. Each state can be used exactly once, because at each IVC step, the index is increased. In turn, at each IVC step, the AD must be updated with the values of the new state. The company can try to use a different a. transfer amount b. sending company id c. receiving company id, but these are all directly derived from the commitment responsible for the signed updated states, and inextricably linked by a transfer zkp: any increase in transfer amount to a receiving company's id, must be increased on the state commitment with the $receiving_{id}$, which in turn must consume a state commitment with the same id - the same argument holds for the $sending_{id}$. The question remains how a company's first state commitment has the correct id encoded. This is enforced by the tax authority during enrollment, because the id is a public input to the enrollment creation zkp. Because we assume a secure authenticated broadcast channel, the tax authority can enforce the use of the correct company id. 4. This is guaranteed by the underlying IVC scheme.

Claim. *A valid Proof of Interaction does not reveal anything other than a company's transacted amounts with blacklisted counterparties to the tax authority.*

The final $\mathbf{zk\text{-}SNARK_{IVC}}$ and $\mathbf{zk\text{-}SNARK_{POI}}$ verification does not use and reveal any sensitive public inputs related to a company's transacted amounts with parties other than the counterparties. Companies do leak the total number of transactions they created on each of their accounts.

Claim. *A valid Proof of Interaction does not reveal anything to any party other than the tax authority.*

We assume a secure authenticated broadcast channel between companies and the tax authority, ensuring no data leaks when the Proof of Interaction is shared.