Privacy-preserving tax-case processing

Espen Kjellstadli Lund
Computer Science Department
NTNU, Gjøvik, Norway

Mariusz Nowostawski
Computer Science Department
NTNU, Gjøvik, Norway
mariusz.nowostawski@ntnu.no

Abylay Satybaldy
Computer Science Department
NTNU, Gjøvik, Norway
abylay.satybaldy@ntnu.no

Nader Aeinehchi
Tax Administration
Oslo, Norway

***Abstract*—Tax-case processing is based on a classification of common taxable conditions, e.g. retirement benefits, social security benefits, income brackets. Based on those, groups of people fall under certain categorizations and should be treated equally. However, to maintain privacy, people from the same group might be handled anonymously by different case handlers. Consequently, tax-case processing for different individuals from the same group categorization may end up with different results, depending on the region in which the case is dealt with and which case manager treats the given case. This situation maintain privacy but violates fairness. On the other hand, some of the tax calculation is personal since there is no isolation between the taxable conditions data and the individual data. Often, taxable data provides personally identifiable queues. Furthermore, tax processing cannot take advantage of previous tax-case processing that has been already conducted; thus, reducing the system's overall efficiency for cases that has been already processed. To improve efficiency and transparency of the taxation process, we have investigated and explored architectural designs for privacy-preserving tax-case processing prototypes with the use of blockchain technology. In this article, we provide an overview of the available technologies, classify the architecture, and discuss a proof-of-concept implementation that can facilitate those goals. Based on the findings, a number of frameworks is considered and a proof-of-concept is made to showcase the design intent. We established, that in some cases it is possible to use blockchain technology for tax-case processing to enhance the privacy and improving the transparency at the same time.**

***Index Terms*—tax-case processing, blockchain, privacy, cryptography, anonymisation, zero-knowledge proofs**

I. Introduction

In this work, we modify tax processing such that we can decouple personal information from the taxable data needed for the tax calculation. The framework that we propose uses blockchain technology as the foundation. Secure distributed ledger facilitates data integrity and transparency through its built-in triple accounting, audit and validation mechanisms. The transaction's origin, destination and content are recorded, such that changes in the system can be observed and verified later. Thus, it can be observed and audited, whether tax-cases were treated equally. Moreover, given that the GDPR regulation was enforced on 25 May 2018, it is of interest to develop a system which could meet compliance with the new regulation; because it intends to give citizens the control of their personal data and decoupling the tax processing from the personal data attached to the individuals.

Tax-case processing with blockchain technology introduces a number of challenges. One of these challenges is the distribution of data such that privacy is preserved while allowing the data to be processed. When simple anonymisation techniques are used and data is not adequately protected, it can lead to individuals being re-identified and sensitive information being disclosed. Moreover, GDPR rules introduce challenges in the context of blockchain technology. For instance, the right to be forgotten require that personal data could be deleted. This right is difficult to enforce because the public blockchain's rely on the immutability of the records. Consequently, support for traditional deletion of data is not possible in typical blockchain systems.

We have developed a taxonomy for discussing storage, anonymity, ways to model data, and a number of different mechanisms to isolate personal data from computational data, such that data can be shared between tax-cases. As a result, we have gained a much better understanding of the limitations of privacy-preserving schemes, and what it means to be within or outside certain privacy guarantees under particular trust models. Moreover, various prototype implementations have been tried on a hyper-ledger and Ethereum blockchains with the contracts set up for the required logic. The example implementations have addressed the problem of anonymity for taxpayers, by using cryptography and anonymisation. Finally, it is important to notice that although this work is about tax-relevant data, it can be easily adapted for other types of data e.g. healthcare data. The proposed solutions are inspired, but not limited to, work only for the Norwegian tax-case processing. The applicable data transformations can vary dependent on the required utility and privacy, as well as the actual data processing requirements.

II. Framework Development

Let us consider two core models: centralised data centre and de-centralised one, in which user is put at the center. In the case of a centralized data centre, we assume that a *tax administration* has accumulated the data for tax-case processing through some means. At this point, data cannot be directly published to the blockchain as it is personally identifiable. Therefore, some data anonymisation must take place to enable publishing. The output of this procedure, allows data to remain useful for taxation and maintain the privacy to the taxpayers. *Tax processor* is responsible for the actual taxation calculations. It is a logical machine that implements taxation logic that can respond to requests. In other words, the tax administration has all the data, including personally identifiable information; but does not execute the taxation processing on a privately hosted infrastructure. The actual processing is offloaded to the *tax processor*.

In the de-centralised model, a *tax payer* takes the responsibility of computing their own taxes. The *institution* (tax administration) provides the computational data to the *tax payer*. Once the taxpayer has accumulated the data and computed their taxes, the output is published to the blockchain for the *tax administration* to validate the actual calculation results. It is the *tax payer* that stores and processes the data. The taxpayer works as the *tax processor*. The *tax administration* knows who is who and how much they should be charged based on the ledger entries.

There are many considerations that need to be dealt with, such as the balance between utility and privacy, how to support a claims based system, the viable interactions between multiple stakeholders, who is responsible for the data privacy, how to guarantee a stakeholder's authenticity, which data representation to chose, and measures to enable attribute modification or expansion. These were some of the factors that played in, and which the frameworks dealt with under different parameters.

A. *Publishing of anonymized datasets*

Tax administration gathers data from multiple institutions and aggregates the information per *national ID* (or tax ID given to individual). This dataset is considered personal and cannot be published. Therefore, the dataset must be treated through data anonymization before it can be published onto a public blockchain.

a sequence diagram on Figure 1 depicts how objects are to interact with each other. It consists primarily of three entities: the *tax administration* submitting datasets to a *data publisher*, which, in turn, broadcasts information about the datasets to clients of the application. At last, the tax administration must be able to request taxation. To satisfy this, a *tax processor* is designed. It contains the application logic required to execute tax-case processing for taxpayers. After such a request is processed, the taxation output is broadcast to everyone. As a result, a framework to notify peers about the taxation phase is proposed; where accumulation of the data and corrections from pre-taxation and self declaration was done beforehand. In other words, this design is concerned with transforming personal data into computational data for the taxation phase; and adopts the idea of a centralized data distribution, where the *tax administration* holds access to the all the data, including financial records as well as personal data.

At first, the *tax administration* is responsible for aggregating data per national ID; in accordance to institutions' reports. As the volume of data increases, the *tax administration* must take into consideration whether data publishing should take place. The considerations varies dependent on the planned technique to acquire anonymity. At some point this decision is made, causing a transaction to be broadcast with its destination being a smart contract in the blockchain. The data field's content of this transaction is an anonymized data or a data reference. Where the smart contract is the *data publisher*, whose implementation differs dependent on the data representation.

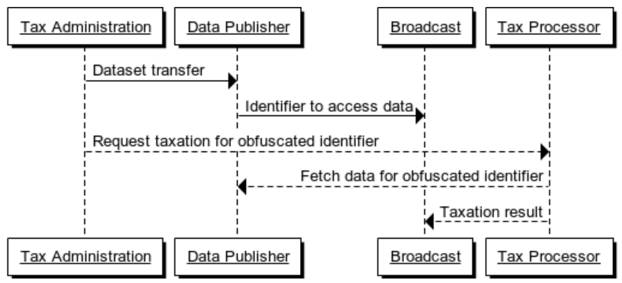


Fig. 1. Sequence diagram for datasets publishing

Next, the *data publisher* is in charge of the management of datasets. It keeps track of datasets and provides notifications to clients of the system about any state updates (e.g. update to the existing dataset or new datasets via *broadcast* in Figure 1). Storing data directly onto the blockchain is not possible due to high costs [1]. Although it appears expensive, it should be considered that the cost is a one-time cost for permanent data storage [1]. Note that the cost is primarily a concern for situations where public/permissionless blockchains are used, as private/permissioned blockchains can adjust the parameters affecting cost. However, when considering the performance and flexibility of the data, it is preferable to store data off-chain [1]. Moreover, if the data is stored at a centralized off-chain platform, the *tax administration* can remain in control of the data. In regard to GDPR, this can allow easier compliance as data can be withdrawn. In distributed storage platforms, none has implemented this feature yet, but some researchers are exploring it. BigchainDB[1] is examining the functionality that would restrict access to data to a time limit [2], [3]. IPFS[2] community appeared to be discussing the matter [4]. Swarm[3] had yet to discuss it but proposed that it is possible to withdraw data once demand and incentive diminish to preserve accessibility [5].

At last, the *tax processor* is accountable for taxpayers receiving their final taxation. In order to fulfill this duty, there are multiple stages that shall take place. First, the *tax administration* must release their dataset; which is required for the taxation. Second, after the release notification is given; it describes how one may interact with the dataset. For instance, it may provide a uniform resource identifier(URI) such that it could be downloaded. The URI is composed of a string of characters used to identify a resource and allows the content to be accessible to authorized parties. However, dependent on this second stage, the design of this particular component can greatly differ. There are three plausible design choices here:

- If data is stored on-chain, and made accessible to other smart contracts; then the tax processor can simply request to fetch this data for further processing. In other words, there are two smart contracts in the picture. The data publisher for storing data, and the taxation for tax-case

[1]Blockchain-centric Database management systems.
[2]Decentralised and Distributed peer-to-peer storage system.
[3]This is another example of a Peer-to-Peer storage system.

processing.

- If the data is stored off-chain, and served through some Web API on the Internet. Then we have two plausible design choices:
  - Use an oracle to fetch data outside the walled garden a blockchain impose. Where an oracle is a smart contract, which have implemented the required functions to pull content off the Internet. Utilizing it would allow the taxation logic to be implemented as a smart contract; especially when the data publisher only stores references to the data.
  - Implement the taxation logic as an off-chain application and publish the results of the taxation on-chain. In this scenario, the data publisher is the only smart contract. Therefore, to support taxation broadcasts it must be extended to support such functionality.

Regardless of the design choice, the attributes required to perform taxation shall be available; such that taxation can proceed with the calculations. Once finished, the output can be published to the blockchain either through the data publisher or tax processor. Where it includes an identifier, such that the tax office is certain about how much they shall charge a certain taxpayer. Finally, the publishing leads to notifications being made, to notify clients about the latest updates.

In regard to errors, we envision that data and taxation errors could be efficiently treated within the system. That needs special attention from the tax administration. The errors must be reported to the tax administration; such that a new and accurate dataset could be generated and published. The data updates shall be dealt with, by publishing another transaction to broadcast another event such that the value stored in the blockchain's event log is updated.

B. Anonymity

No proposed anonymization scheme in academic literature is found appropriate for taxation. This therefore requires an additional research to identify and establish the best anonymisation scheme for tax data. We hypothesize, that tabular data anonymization schemes in many cases are appropriate. For instance, we discovered that t-closeness had addressed how MSA could be dealt with, and could be combined with k-anonymity that can be combined with the quasi-identifiers. Although there exist transformations based on permutation or perturbation, those introduce untruthful data; which does not fit to support taxation. Thus, we constrain ourselves to generalization and suppression. This does, however, make a number of tax payers being excluded from the system, because their privacy cannot be guaranteed for (e.g. due to data uniqueness).

Another important aspect is the *tax office's ability to perform re-identification*. Although the anonymized data does not contain identifiers, the tax administration must be able to acquire the correct taxation for a particular tax payer. In order to fulfill this requirement, another mechanism must be introduced to the system. There are two methods which turns out to be imminent: 1. encrypt the identifier with the tax administration encryption key, or 2. create randomized identifiers to be published, which are mapped to a national ID in a private data store. Regardless which mechanism is used, it is included with the anonymized data on the blockchain. Moreover, considering that both methods can be reversed back to personal data they are subject to GDPR and must comply with the requests to be forgotten. We will look into this in the following section.

C. GDPR requirements

Our framework may allow compliance to GDPR through a combination of anonymization and encryption. The access to the identifiers must be restricted, such that only the tax administration shall be able to identify the data owner. The actual taxable data has been scrubbed for potential identifiers and is, therefore, possible to leave it in a public storage system. In order to perform re-identification, additional information is required such as a decryption key or national ID to randomized identifier mappings. This implies that once a request to erasure of data has been received, a process must instantiate to destroy such additional information.

Alternatively, a scheme similar to BCDiploma could be used. It has been marketed as a GDPR compliant solution for publishing diplomas. To obtain this compliance, the data is encrypted with three separate keys on the blockchain [6]. Two of the keys, namely the *graduate key* and *persistence key* are required to create a intermediate key. In which the graduate key is integrated into a URL for the diploma to be accessed. The intermediate key is combined with the school's permanent key, to derive the final key. To satisfy the right to be forgotten, the school's graduate can request the persistence key to be destroyed to render the data inaccessible; as all keys must take part in the decryption process. In other words, the blockchain is used as a storage platform for encrypted data. The data needs to undergo a decryption procedure off-chain at the school responsible for publishing diplomas. The procedure is initiated when a reader application transmit a request to the diploma's URL that consists of diploma number and graduate key.

Finally, note that cryptographicly obfuscated data can also fulfill the right to be forgotten by deleting the data. However, this requires that data is stored in a storage platform, that allows to delete objects. As of now that implies that data is stored off-chain. There are some private blockchain proposals in the permissioned ledgers space, such e.g. Accenture's proposal of a editable blockchain through chameleon hashes [7], that might enable deletion of the data on chain.

D. Advantages and disadvantages

The advantages associated with this framework are:

- Freedom of flexibility in terms of anonymization
- Simple design, similar to existing architectures
- Allows storage to be done either on-chain or off-chain

The disadvantages associated with this framework are:

- Requires off-chain software to anonymize datasets

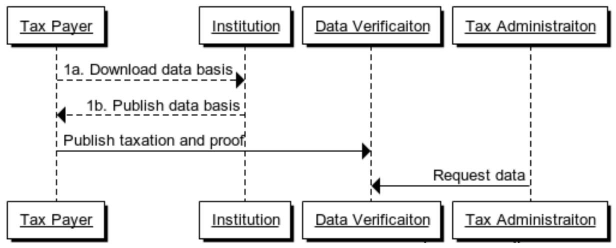


Fig. 2. Publish self-taxation

- Institutions cannot publish information directly to the blockchain, it needs to be obfuscated, encrypted and anonymized
- Tax administration is responsible for collecting and anonymizing data
- Does not allow accumulation of reported data to be done on-chain

III. Self-taxation

To provide an enhanced privacy model, let us try to turn the roles around. It would be the taxpayers that compute their taxes themselves. As such, the tax administration only needs to verify the taxes to determine if they can be considered authentic and valid. As a result, much of the taxation process can be offloaded to the taxpayers. To make taxpayers participate in this scheme, we deem it necessary to provide some incentives.

In this framework, the taxpayers are responsible for computing their taxes themselves. Considering that taxpayers may not have all the required data, the institutions are obliged to provide the data export mechanism; which both, the tax administration and taxpayers, can utilize. When the taxpayer has reached a conclusion about data to include in taxation, it is processed by a number of different functions. The functions serve the purpose of calculating tax and creation of proofs that the function has been executed correctly. They are based upon the idea of cryptographic proofs, where the user acts as a prover; and the data verification as a verifier. The verification serves the purpose of filtering out, inaccurate or untruthful taxations; while also acting as an interface for the tax administration to retrieve taxation data. Figure 2 depicts the interactions, that can occur in this framework.

With the emphasis on self-taxation, we achieve a shift in entities' responsibilities. The tax payers now take a more participatory role, because taxation has typically been dealt with by the tax administration. This involves a variety of tasks to be fulfilled. There is the need to gather the data for the individual's national ID. As depicted in Figure 2, there are two suggested approaches to achieve this:

- 1: Interaction with institutions' own infrastructure, which are capable of authentication, authorization and data export. A tax payer may therefore need to perform numerous login procedures, to verify their identity to different data providers; such that data export can start. There is no connection to the blockchain, and is thus referred to as a off-chain approach. Consequently, this is considered an active approach to obtain data because of the number of institutions that must be interacted with.
- 2: Institutions publishes data, to the tax payers' address in the blockchain. This make the process less interactive, and demand less effort from the tax payer; because the institutions do the effort of sending data. This approach is considered passive, due to the reduced number of interactions required. Moreover, since it makes use of the blockchain network, it is referred to as an on-chain approach.

We capture the interactive property in the following definition:

Definition 1 (Passive/active gathering): The process of harvesting data is passive, if the user does not have to interact with institutions to export the data. It is considered active, if such interactions must take place to receive the data.

If we assume that data is gathered, the tax payer can continue by computing their taxes. This involves downloading a set of functions that are to be executed, either from the blockchain or the tax administration itself. The place where such functions are stored does not affect the privacy model. Once executed, a taxation and proof from running the functions are returned. This data is to be passed to a smart contract on a blockchain for data verification.

Next, the data verification entity is a smart contract responsible for the verifying the authenticity of the generated proofs and taxation that it receives. That is to say, it contains the necessary logic required to perform verification of the proofs. When done verifying a given proof, the proof and taxation can be stored within the entity if this is desired. If it ends up following that route, the entity is also responsible for the management of taxation and proofs. On the other hand, if the data is not stored in the entity, but rather persisted onto the blockchain through events; then the entity only broadcasts the result of the verification and taxation, such that clients can search for the events.

This brings us to the last remaining piece: the *tax administration*. They are responsible of creating the data verification entity, publishing functions and the additional data required to generate proofs (e.g. cryptographic keys). Furthermore, they must also retain some mapping from a published taxation to its respected national ID. This could be done by registering a blockchain account to the tax administration, and providing digitally signed proof about being a citizen within the Norwegian state. Describe details of such registration schemes are out of scope for this paper.

A. Anonymity

Anonymity in this scheme is provided through zero-knowledge proofs. Suppose the tax administration are given a taxation x of some value, and wish to have some proof that a tax payer knows the value w that resulted in taxation x. Normally, this would involve giving w to the tax administration; which could compute the taxation to check if it equals to taxation x. However, the tax payer does not want to reveal the value w to the tax administration; only that he or she knows the value. In essence, we want to obtain the functionality described in the following function (without conveying w):

```
function C(x, w) {
    return ( taxation(w) == x );
}
```

The function takes the secret value w and public taxation x as input, and returns *true* if the taxation of w equals x. Unfortunately, it requires that the secret value w is revealed, which is not a desirable feature. Therefore, it would be interesting to translate this function, such that only a proof of knowing w would be enough. This would result in privacy, because the value w is never disclosed.

In other words, we need the ability to convince a verifier that the prover possess knowledge of a witness, which satisfies some relation without revealing the secret. In cryptography, there is a method referred to as zero-knowledge proofs which satisfies this ability. There exists a variety of implementations, often categorized into non-interactive or interactive zero-knowledge proofs. For this framework we are interested in the non-interactive case; because it enables greater efficiency, by requiring minimal interaction from the prover to verify the proof. This type of proof construction could be done by *zero-knowledge non-interactive argument of knowledge*, also referred to as zk-SNARK. It is beyond the scope of this article to discuss the use of zero knowledge proofs in detail. We have used existing implementations in our proof of concept, and identified some of the limitations of the existing implementations.

B. GDPR requirements

Our framework fulfills privacy requirements and the right to be forgotten through zero-knowledge proofs and encryption. Both techniques are categorized as cryptographic obfuscation, but each has different purpose. The former aims to provide proof for a certain taxation without revealing the actual financial or personal data; while the latter aims to enable such proofs, by transmitting securely the personal data to the tax payer. If institutions published the actual data on-chain, it is encrypted with the taxpayer's encryption keys. Thus, to enable right to be forgotten the taxpayer only needs to destroy their key.

Moreover, the design also entails that institutions and tax administration need some sort of mapping between national IDs and their blockchain identifiers. This is because institutions shall publish data to the right recipient, and for the tax administration to retrieve the correct taxation. As a result, tax payers must register themselves, to verify the identity behind a certain blockchain identifier. Unfortunately, this means that when a right to be forgotten request arrives; such mappings must be deleted from the private data store of the tax administration. To summarize, the following shall happen in case of a right to be forgotten:

- Tax payer shall destroy their decryption key
- Tax administration shall remove their mapping

As a result, the data is rendered inaccessible, because the decryption keys to the data is no longer available. Furthermore, no one shall know that a national ID is mapped to a certain blockchain identifier.

C. Advantages and disadvantages

The advantages associated with this framework are:

- Core data can be transferred off-chain/on-chain
- Taxation process off-loaded to the taxpayers
- Privacy through cryptographic obfuscation and end-user data ownership
- Smart contract only needs to verify proofs
- Data transparency on how taxation is verified and applied

The disadvantages associated with this framework are:

- Does not support a claims-based system
- Mapping between blockchain identifier and national ID is required
- Associations made by various institutions may erode privacy
- Impossible to proof that mappings between identifiers are deleted
- Requires tax payers to verify their identity
- Taxpayers need to be coerced to participate and conduct the calculations. This is equivalent to filling out the tax returns.

IV. Synthetic data

In order to evaluate the proposal we needed to generate a synthetic dataset. This allowed us to experiment with different anonymization schemes, to evaluate of how generalization and suppression could be applied to guarantee privacy.

The distribution parameters used were the following:

1) *Age (Gaussian)*: Avg: 28 [8], Std: 3
2) *Municipality (Uniform)*: Range: [0101-5054] [9]
3) *Income (Gaussian)*: Avg: 16000, Std: 3000
4) *Debt (Gaussian)*: Avg: 280826 [10], Std: 10000
5) *Wealth (Gaussian)*: Avg: 10000, Std: 1000
6) *BSU (Gaussian)*: Avg: 12500, Std: 1000
7) *Married (Bernouilli)*:
   P(married|<30): 7.3% [11], P(¬married|<30): 92.7% [11]

Although numerous of the attributes had information about their averages, no measures of their standard deviations were found for their Gaussian distribution. Therefore, it was required to generate some random values that likely do not represent the true distribution. Moreover, a couple of assumptions were laid for the averages of Wealth, BSU, and Income attributes such that data could be generated. These assumptions were:

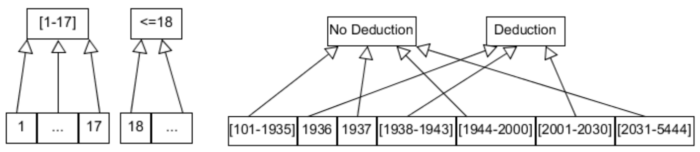
- We assumed students had acquired part-time jobs, and have a minimum wage of 150,- NOK per hour. The average work hours were estimated to be ≈ 8 hours per week; and only one third of the Norwegian students were estimated to have part-time work [12]. This resulted in the following equation to acquire the income average: $150 * 8 * 4 * 10/3 = 16$k

- Next, we assumed students had been saving in average 100k, for students between 18-26 years old [13]. Thus, we divided $100k/8$k to obtain the estimate that students insert roughly 12.5k each year into BSU.
- Finally, that students had acquired a cheap car with an average value of 10k. Its value is accounted for when measuring how much wealth a certain student had in a particular year.

V. Implementation

A. Publishing of tabular data

To realize the *Publishing of anonymized datasets* framework, we chose to first address the underlying assumption that the data (base data plus corrections from pre-taxation and self-declaration) had been established. This was done through the synthetic data generation explained in Section IV. With this data in possession, we needed some anonymization toolbox that had implemented a variety of anonymization techniques. This toolbox must support k-anonymity, t-closeness, and the creation of value generalization hierarchies. Based on a comparison of different anonymization tools [14], it was discovered that only ARX fulfilled these requirements. As such, we sought to use this tool to generalize and suppress the data; to achieve certain objectives set in the k-anonymity and t-closeness privacy models. As mentioned earlier, these models has certain parameters. In the case of k-anonymity, it was common to: (1) get a data custodian to select a value of k commensurate with the re-identification probability they are willing to tolerate (a threshold risk) [15]; or (2) determine the optimal value for k based on some utility vs privacy curve [16]. However, for t-closeness there is no well established method to determine a value for t [17]. Roy and Jena [17] did therefore propose to use some method based on how sensitive attributes were partitioned in equivalence classes [17]. What these methods tend to share, was the use the information loss imposed by the applied generalization. Such a metric was not viable for our scenario, when we sought to achieve the specified privacy goals. More specifically, to limit reuse of data for other purposes than just tax calculation; we were required to apply the maximum generalization, to such a point that it was almost only viable for taxation. In other words, generalization was static and not a variable. What was a variable though, was how many tax payers could be suppressed from the dataset due to set privacy-guarantees. Therefore, we came to the decision that the number of suppressed records was a more appropriate metric. Unfortunately, support to find optimal privacy parameters based on this metric was not implemented in ARX. Therefore, we leave this as further work. If we were to implement some algorithm for the optimal privacy parameters, we would make use of optimization algorithm to minimize a cost function(e.g. gradient descent); in which the privacy-guarantees and suppressed records were taken into account. Finally, some algorithm would finalize the choice by looking for the elbow point and evaluate the resulting risks.



(a) Attribute Age (b) Attribute Municipality

Fig. 3. A figure of two value generalization hierarchies

However, this wass out of scope for this paper. We restricted the problem and chose a single parameter set instead, which is appropriate for testing purposes. Once established, ARX was to generalize and suppress the following attribute setup: Quasi-Identifiers: $\{Age, Municipality, Married\}$, Sensitive Attributes: $\{Income, Debt, Wealth, BSU\}$. The VGHs used were shown in Figure 3.

For the data publisher, we pursued a centralized design pattern; in terms of how the blockchain logic is distributed. It appeared that such a design pattern was preferable because it imposed less cost, limit replication of identical code, and unified logic. This involved a single smart contract being responsible for taking upon the *data publisher* role. As discussed earlier, it was generally preferable to store data off-chain in terms of efficiency. However, in scenarios where data was to be processed on-chain to generate some output; the data must exist on-chain or be imported through the oraclize service. Since the transaction size limit was close to roughly 0.1MB on Ethereum, it was hard to publish the data in one swoop; taken into consideration that the synthetic data of 30k records was roughly 2MB. If we were to scale the application to 1 million records, the data would need to be split into numerous transactions. It was therefore decided that data was better stored off-chain. Given our limited access to numerous storage platforms, it was decided that the distributed file system *Inter Planetary File System* (IPFS) shall be used. It turned out to be the more established and mature decentralised file storage protocol, that offers content-addressable access. Consequently, the data publisher's design was reduced to act as a registrar for external URIs. To store these, it was deemed that events would be emitted rather than storing them within the contract. The reasoning behind this was their cheaper form of storage; and the ability to easily scan the history of events, to get a project of the state of the taxation and published datasets.

The following code-snippet depicts how we developed a smart contract capable of emitting events. It was developed with the programming language Solidity version 0.4.16, and has defined two types of events: DatabaseReceipt and TaxationReceipt. The two events had defined certain attributes as indexed, such that they are parameters that can be searched for. To enable publishing of such events, the client needs to call either the depositData() or depositTax() functions.

```
pragma solidity ^0.4.16;

contract TaxReceipt {
    // Event definitions for databases
    event DatabaseReceipt(
        // Indexed attributes allow the event
        to be searched for by the attribute
        address indexed _owner,
        // Non-indexed attributes
        string _name,
        bytes32 _hashURI,
        string _comment
    );

    // Event definition for taxations
    event TaxationReceipt(
        // Indexed attributes allow the event
        to be searched for by the attribute
        bytes32 indexed _hashURI,
        address indexed _owner,
        uint indexed _encryptedID,
        // Non-indexed attributes
        uint _taxation
    );

    // Function definitions for tax administration
    to publish data or taxes
    function depositData(string _name, bytes32 _hashURI,
    string _comment)
    public {emit DatabaseReceipt(msg.sender, _name,
    _hashURI, _comment); }

    function depositTax(uint _encryptedID, bytes32 _hashURI
        ,
    uint _taxation)
    public {emit TaxationReceipt(_hashURI, msg.sender,
    _encryptedID, _taxation);}
}
```

To interact with the smart contract, a decentralized application(DApp) was made. We use the web3js library, to enable communication with Ethereum in the web browser. The library behave as an *application programming interface*(API), which forwards its requests to a Ethereum full node (e.g, geth). In other words, it allowed users to publish external URIs for datasets and taxations in a web-browser. However, in order to obtain the external URIs; the multihash from IPFS needed to be retrieved. As such, the js-ipfs-api library was imported to enable uploads to IPFS and obtain the multihash of uploaded datasets.

To summarize, the proof-of-concept addressed the privacy goals in the following manner: The risk of unintended reuse of data is limited through a strict domain specific generalization. Moreover, the generalization was done such that the granularity is good enough for taxation purposes. Given that the data was transformed through anonymization techniques, to protect against identity and attribute disclosures through k-anonymity and t-closeness; we could potentially allow data to be made publicly accessible (but to make sure, an evaluation of the re-identification risk should be done until such a decision can be made). Finally, we believe that GDPR compliance could be achieved by requesting the encryption key tied to an encrypted identifier to be deleted upon request. Unfortunately, we did not implement or draft a procedure to how this encryption key can be deleted in an auditable procedure. The source code is released to the public domain.[4]

[4]https://github.com/neeps/taxreceipts

B. *Self-taxation with zero-knowledge techniques*

This was an ambitious attempt intended to realize the *self-taxation* framework, where the tax payers are in charge of computing their taxes themselves; while preserving the integrity and privacy of the process. If the process is not honest, the tax administration cannot trust the published values. In such a scenario, the value of the system is greatly reduced. Although, such verifiable computing can already be obtained in the blockchain, because all computations must be replicated on the network's nodes; it does not achieve the desirable privacy features, that we sought towards eventual GDPR compliance. In particular, we were interested in the ability to keep information secret and provide proof about knowing such information. To obtain the desired privacy, zero-knowledge proofs were combined with blockchain technology. Furthermore, to ensure that tax payers cannot make deceptive inputs; there must be a function to verify the authenticity of the data. This may involve the verification of a signature attached to the retrieved base data.

ZoKrates is a toolbox to generate keys, proofs, and smart contracts; and it is suitable to create and verify zero-knowledge proofs in Ethereum, without needing to know about the underlying details. To enable this, it supported a high-level language and a compiler to transform programs to provable constraint systems [18]. To get started with the toolbox, the sequence of commands to generate proofs and a verification contract are specified. The first prerequisite is that the logic has already been written in the mentioned high-level language. Once this has been established, it can be fed into the compile command. It outputs an arithmetic circuit (*program C*. Next, the output is passed to the setup command, to generate the proving key and verification key. It implements the key generator function. Next, a set of parameters called witnesses must be created for use in generation of a proof. This is because the proof is dependent on specific values of public and private parameters. To deal with this, the compute-witness command generates a *witness* file. The proof can now be made, by executing the generate-proof command which feeds the public and private witnesses, and public key into the prover function. At last, the export-verifier command can be used to generate a solidity verification contract; which contains the verification key and a verifier function.

Unfortunately, the high-level language was limited in its functionality. First, it only supported prime field elements (unsigned integers, subject to modulo). This prevented real numbers from being obtainable in the software's current implementation. As such, one could not represent the applicable tax rates for salaries, wealth, and more. To understand whether this was a limitation with ZoKrates or zero-knowledge proofs in general, we discussed with Jacob Eberhardt, the author of ZoKrates and quote him: "It is possible to simulate binary arithmetic and with that floats in an arithmetic circuit, though, so that way it could be supported. However, we currently do not expose types for that, as it leads to a huge number of constraints and with that proofs that are expensive to

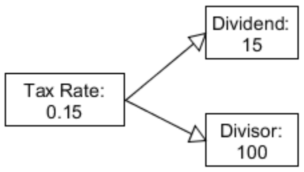


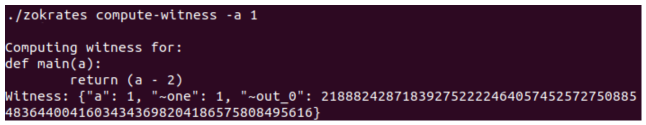
Fig. 4. Workaround for float-point arithmetic

generate”. Support for floats were also mentioned in the libsnark’s vnTinyRam preprocessor [18], [19], which is another circuit generator that shares part of its pipeline to create zero-knowledge proofs. It mentioned that floating-point arithmetic are not directly supported, but could be implemented in software [19]. Second, there was no support for arrays; as such all the input parameters must be explicitly defined in the function declaration. As a result, it was not possible to create a single program capable of tackling every taxation scenario. Therefore, it may be necessary to create separate programs for each respected group of tax payers. Third, a function to verify signatures would be needed to prevent deceptive inputs. There does currently not exist any libraries implemented in this high-level language, to allow for such functionality.

To get around the issue regarding lack of rational numbers arithmetic, it was first thought of splitting up the computation into two operations: by multiplying the dividend with the input variable, and then divide the result on the specified divisor. This would allow the tax rate, to be applied and be represented as a pair of integers.

Furthermore, given that the system was built on top of prime field elements, it meant that any value going beneath 0 was subject to modulo arithmetic. This is similar to how a clock would work, in that once it reach 24:00 it goes back to 00:00. Except, in this example we may go in the opposite direction: 0 to the relevant modulus M. To showcase this, an application resulting in -1 was made. The result of this is shown in Figure 5, where the modulus M is revealed in $\sim out\_0$. In other words, precautions must be made to ensure that potential negative numbers are handled correctly. This could either be done, by imposing if-else conditions checking for such occurrences (e.g, if a < b then 0 else (a-b)).

Before we ventured into more complex arithmetic circuits and evaluation of which tax rates were appropriate for certain attribute values. The prototype beneath was made for a simple taxation scenario, where a number of parameters had been pre-defined. For instance, assume that Bob was a single college student making him subject to tax class 2. Second, he had more debt than wealth such that he was not subject to wealth taxes. Third, he lived in the municipality and county of Oslo; so he was not qualified to receive tax deductions which individuals living in Finnmark and Troms county can claim. Fourth, his income for 2016 was 70k. This resulted in a minimum standard deduction of 31.8k, because 43% of the income was 30.1k which was less than the lower limit of 31.8k; therefore the



```
./zokrates compute-witness -a 1

Computing witness for:
def main(a):
        return (a - 2)
Witness: {"a": 1, "~one": 1, "~out_0": 21888242871839275222246405745257275088548364400416034343698204186575808495616}
```

Fig. 5. Example of modulus arithmetic

lower limit was set. Moreover, this deductions was subtracted from the gross income to produce the general income of 38.2k. Then, we subtract the personal allowance from the general income, to obtain the basis for general income tax. Provided that there are no additional income based upon interests, a bracket tax rate of 0.0% applies because the income was less than 159.8k. At last, a national insurance contribution was computed as well. In other words, Bob was to be represented by the following data:

$$Bob = (taxClass : 2, income : 70k, minStandardDeduction : 31.8k) \quad (1)$$

```
def main(private taxClass, private income, private
    mStandardDeduction):
  // Subtraction, where evaluation for negative number is
      required
  generalIncome = if income < mStandardDeduction then 0
      else (income - mStandardDeduction) fi
  // Another deduction based on taxClass
  personalAllowance = if taxClass == 1 then 51750 else
      76250 fi
  // Subtraction, where evaluation for negative number is
      required
  generalIncome = if generalIncome < personalAllowance then
       0 else (generalIncome - personalAllowance) fi

  // Compute taxes
  gTax = (generalIncome * 25)/100
  nTax = (income * 82)/1000
  tax = gTax + nTax

  // Return taxes
  return tax, nTax, gTax
```

Before we expanded our prototype application to support tax rate evaluation, it was discovered that the initially planned workaround for division did not work as intended. Since the ZoKrates toolkit was built upon prime field elements, the division operator provides some challenges. For instance if we tried to do x = 6333/1000, the system would try to determine which integers x would be such that $x * 1000 = 6333$ (mod p) was satisfied. Since real numbers were not supported, the value x obtained would be a number close to modulus; which when multiplied with 1000 would go beyond modulus and result in 6333 when (mod p) was applied. In other words, the system was not capable of simulating real numbers. As such, ZoKrates use case for taxation at its current iteration was limited. The proof-of-concept was therefore not expanded to support tax rate evaluation.

As a last resort, a possibility was to reduce the implementation to range proofs instead. This meant that the tax payer would no longer compute taxes, but provide information such that taxation can proceed. For instance, one may use range proofs to reveal information about a certain attribute, e.g. that a salary is within the salary bracket: [400k - 500k]. Or another option, would be to use range proofs to support a certain tax rate being applied. This would help the tax administration manage and apply certain taxation templates, dependent on the claims about tax rates. In terms of anonymization techniques, the range-proofs could be regarded an alternative approach to achieve generalization. One implementation of such range proofs was zkrangeproof [20]. It requires the Ethereum clients to include a precompiled smart contract; such that validation can be done, because it was regarded too expensive to be run in the Ethereum virtual machine(EVM). Unfortunately, the authors have specified that due to security vulnerabilities in *An Efficient Range Proof Scheme* [21], the proofs generated from this system were no longer zero-knowledge. Although ZoKrates could be used to generate such proofs, there was no support for more than '>' operator; and the less than '<' operator was experimental and had bugs reporting about strange behaviours [22], [23]. Thus, we chose to not pursue this approach either.

VI. Results

Our project focused on architectural consideration and experimentation with existing technology to evaluate how to implement tax-case processing with enhanced transparency and privacy features. We learned that:

- Storing data on-chain was expensive on Ethereum, as roughly 2 MB had a cost of $48.57 to publish (at the time of research, mid 2018). In comparison, a reference to the same 2 MB on IPFS had a cost of $0.0092. Thus, if data must be stored on a public blockchain; it was preferable to store data off-chain due to costs and preferable in a centralized storage platform such that ability to delete data upon request can be obtained.
- The privacy models proposed in the field of anonymization techniques tend to be implemented for tabular data representations. We have not found tools to obtain e.g. t-closeness for graph data.
- The search for k-anonymity and t-closeness's optimal parameters were either based on information loss or a data custodian specifying a privacy guarantee. Considering that the generalization we applied was maximized to limit reuse, and there were no specified privacy guarantee to obtain; we proposed to rather use the amount suppressed records, as a mean to find the optimal parameters. But support to find optimal parameters based on this metric, was not to be found in the data anonymization tool ARX.
- To determine if the value of k-anonymity and t-closeness were adequate to protect privacy, a risk analysis should be done to evaluate re-identification risks.

Also, for the *self-taxation* framework, we attempted to create an implementation specialized in *self-taxation with zk-SNARKs*. In this architecture, tax payers were in charge of their data, and control whom to share it with and how it would be processed. It allowed us to identify that: the tools available to do zero-knowledge proofs for blockchains are scarce, and those available have limited capabilities, often reduced to simple integer arithmetic. Thus, until support for either integer division or floating point arithmetic is implemented, zero-knowledge proofs libraries for taxation cannot be expected to work.

Moreover, due to the issue of transactional privacy we were limited to privacy-preserving schemes on the data field in a transaction for the implementation. Which unfortunately, had to be done through off-chain/third-party applications because the schemes were not available in Ethereum's blockchain client. In addition, the lack of code obfuscation made us rely on peers to do confidential processing off-chain; as we cannot process cryptographicly obfuscated data without leaking personal data. Although some headway was made by allowing zero-knowledge proofs to be verified on-chain, we cannot do computations on homomorphic encrypted data, SMPC, or encryption/decryption in a smart contracts on Ethereum, as all the data keys would leak through the public/transparent computational model. Thus, if privacy-preserving tax-case processing was to be done on-chain, the personal data had to be made computational either through anonymization-based schemes or zero-knowledge proofs. As a result, we have identified a number of limitations of blockchain technology in the context of implementing privacy-preserving tax-case processing.

VII. Discussion

In this article, two frameworks were proposed with attention to two different data processing models. Those were: (1) a centralized data centre where the tax administration had accumulated the data, and (2) a decentralized data centre where institutions and organizations publish the data to the public and the taxpayers compute taxes themselves. To address the privacy requirements and right to be forgotten set by the GDPR regulation, the mentioned frameworks suggested the following measures: (a) Use of anonymization and cryptographic obfuscation or pseudonymization to make the data anonymous; (b) Storing data off-chain, to bypass the problem with blockchains' immutability. Thus, data can be deleted upon request or when vulnerabilities with the applied privacy arise. Note that compliance to the right to be forgotten appears to also be satisfied if, for example, the decryption key is deleted; despite data being persisted onto a blockchain.

With these measures we sought to accomplish privacy where: data reuse was limited to certain scenarios, the data granularity preserved ability to perform tax calculations, transparency of the taxation process was granted, and that data would be removed if the right to be forgotten was activated. However, due to the public nature and privacy limitations of blockchain technology; it was recognized that the lack of transactional privacy and code obfuscation made the model reliant on off-chain data anonymization. Thus, the convenience of blockchain technology was diminished. It makes it difficult to encourage the use of blockchains for privacy-preserving tax-case processing. Nevertheless, we have identified the methods and metrics available to apply and evaluate anonymity; whilst preserving data transparency and ability to process data on-chain. We found that anonymization techniques allowed data to remain in a public and interpretable state for tax-case processing; zero-knowledge proofs could be verified on-chain with Ethereum. But until anonymization has been addressed in regard to re-identification risks with a high-dimensional dataset, the tools for zero-knowledge proofs provide functionality for integer division, or floating-point arithmetics and additional advances are made in either transactional privacy of transactions (e.g. through ring signatures like in the case of Monero, or through zero-knowledge proofs like in ZCash) we conclude that the blockchain technology is not mature enough to be recommended for tax case processing. Other alternatives should be considered.

Even if the frameworks had addressed the privacy goals with off-chain data anonymization, it would be at the cost of transparency; because a portion of the system would be hidden from the public. One of the advantages of blockchain technology is its transparency. With it being limited to maintain privacy, we suggest that other ICT models should be considered. For instance, one could use a traditional database system with secure signing and encryption mechanisms combined with PKI to achieve similar if not better privacy metrics with most of the blockchain problems not-applicable. Therefore, we advocate further work to focus on evaluating alternative technologies, as well as making research into privacy techniques in blockchain technology.

We conclude that blockchain technology could potentially be feasible with anonymisation-based schemes for privacy-preserving tax-case processing. But until the experiments have been repeated for high-dimensional datasets, and an appropriate re-identification risk has been set it is hard to advocate the use of the researched models, due to the limited scope of the study. To attain the sought efficiency and transparency for privacy-preserving tax-case processing remains an open research question.

References

[1] X. Xu, I. Weber, M. Staples, L. Zhu, J. Bosch, L. Bass, C. Pautasso, and P. Rimba, "A taxonomy of blockchain-based systems for architecture design," in *2017 IEEE International Conference on Software Architecture (ICSA)*, April 2017, pp. 243–252.

[2] S. Schwerin, "Privacy on the blockchain," 2017, iPDB BigchainDB meetup. [Online]. Available: https://www.youtube.com/watch?v=JVsSmSY2qrs

[3] K. Lemoie, "Blockchain and the right to be forgotten," http://badgechain.com/badgechain-newsletter-16-blockchain-and-the-right-to-be-forgotten/, January 2018, (Accessed on 02/17/2018).

[4] "Can i delete my content from the network? - help - discuss.ipfs.io," https://discuss.ipfs.io/t/can-i-delete-my-content-from-the-network/301, May 2016, (Accessed on 02/17/2018).

[5] V. Tron, A. Fischer, D. A. Nagy, Z. Felföldi, and N. Johnson, "Swap, swear and swindle incentive system for swarm," https://ethersphere.github.io/swarm-home/ethersphere/orange-papers/1/sw%5E3.pdf, (Accessed on 02/17/2018).

[6] BCDiploma, "Whitepaper," https://www.bcdiploma.com/img/BCD-WhitePaper_last.pdf, January 2018, (Accessed on 05/15/2018).

[7] R. Lumb, D. Treat, and O. Jelf, "Why distributed ledger technology must adapt to an imperfect world," https://www.accenture.com/t20160927T033514Z__w__/ae-en/_acnmedia/PDF-33/Accenture-Editing-Uneditable-Blockchain.pdf, (Accessed on 05/19/2018).

[8] A. Lindholm and E. Tønnessen, "Flere studenter under 24," https://khrono.no/studentalder-alder-unge/flere-studenter-under-24/138277, July 2017, (Accessed on 04/27/2018).

[9] E. Bolstad and G. Thorsnæs, "kommunenummer – store norske leksikon," https://snl.no/kommunenummer, March 2018, (Accessed on 04/27/2018).

[10] Lånekassen, "Lånekassen i 2016," https://www.lanekassen.no/Global/Arsrapporter/Laanekassen_i_2016.pdf, 2016, (Accessed on 04/27/2018).

[11] Statbank Norway, "Persons 18 years and over in private households living/not living as couples, by age 2005 - 2017," https://www.ssb.no/en/statbank/table/06095?rxid=40c95e65-8aaf-45a2-bc5a-6fab5f493f67&loadedqueryid=10006256&timetype=top&timevalue=1, (Accessed on 04/27/2018).

[12] A.-L. Keute, "For mye betalt arbeid går på bekostning av studietiden - ssb," https://www.ssb.no/utdanning/artikler-og-publikasjoner/for-mye-betalt-arbeid-gar-pa-bekostning-av-studietiden, August 2017, (Accessed on 04/27/2018).

[13] B.-E. Mikalsen, "For mange er bsu sparing til bolig nummer to eller tre," https://www.dn.no/privat/eiendom/2015/10/21/1903/Sparing/-for-mange-er-bsu-sparing-til-bolig-nummer-to-eller-tre, 2015, (Accessed on 04/27/2018).

[14] F. Prasser, F. Kohler, R. Lautenschläger, and K. A. Kuhn, "Arx-a comprehensive tool for anonymizing biomedical data," in *AMIA Annual Symposium Proceedings*, vol. 2014. American Medical Informatics Association, 2014, p. 984.

[15] K. El Emam and F. K. Dankar, "Protecting privacy using k-anonymity," *Journal of the American Medical Informatics Association*, vol. 15, no. 5, pp. 627–637, 2008.

[16] R. Dewri, I. Ray, I. Ray, and D. Whitley, "On the optimal selection of k in the k-anonymity problem," in *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*. IEEE, 2008, pp. 1364–1366.

[17] D. Roy and S. Kumar Jena, "Determining t in t-closeness using multiple sensitive attributes," *International Journal of Computer Applications*, vol. 70, no. 19, pp. 47–51, 2013.

[18] J. Eberhardt, "Zokrates - a toolbox for zksnarks on ethereum," 2017, devcon3. [Online]. Available: https://github.com/JacobEberhardt/documents/raw/master/talks/ZoKrates-EthereumDevcon3.pdf

[19] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, "Succinct non-interactive arguments for a von neumann architecture." *IACR Cryptology ePrint Archive*, vol. 2013, p. 879, 2013.

[20] T. Koens, C. Ramaekers, and C. van Wijk, "Efficient zero-knowledge range proofs in ethereum," https://www.ingwb.com/media/2122048/zero-knowledge-range-proof-whitepaper.pdf, 2017, (Accessed on 05/26/2018).

[21] K. Peng and F. Bao, "An efficient range proof scheme," in *Social Computing (SocialCom), 2010 IEEE Second International Conference on*. IEEE, 2010, pp. 826–833.

[22] "Possible less-than operator issue? · issue #7 · jacobeberhardt/zokrates," https://github.com/JacobEberhardt/ZoKrates/issues/7, November 2017, (Accessed on 05/20/2018).

[23] "panick at 'no entry found for key' · issue #43 · jacobeberhardt/zokrates," https://github.com/JacobEberhardt/ZoKrates/issues/43, April 2018, (Accessed on 05/20/2018).