CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Client and Server Verifiable Additive Homomorphic Secret Sharing

Enforcing Clients to Act Honestly in Sever Verifiable Additive Homomorphic Secret Sharing by including a Range proof

Master's thesis in Computer science and engineering

Hanna Ek

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2021

Master's thesis 2021

Client and Server Verifiable Additive Homomorphic Secret Sharing

Enforcing Clients to Act Honestly in Sever Verifiable Additive Homomorphic Secret Sharing by including a Range proof

Hanna Ek

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2021

A Chalmers University of Technology Master's thesis template for LaTeX
Enforcing Clients to Act Honestly in Sever Verifiable Additive Homomorphic Secret Sharing by including a Range proof
Hanna Ek

Supervisor: Georgia Tsaloli and Katerina Mitrokotsa, Department of Computer Science and Engineering
Examiner: Katerina Mitrokotsa, Department of Computer Science and Engineering

Master's Thesis 2021
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Gothenburg, Sweden 2021

A Chalmers University of Technology Master's thesis template for LaTeX
Enforcing Clients to Act Honestly in Sever Verifiable Additive Homomorphic Secret Sharing by including a Range proof
Hanna Ek
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Abstract text about your project in Computer Science and Engineering.

Keywords: VAHSS, Range Proofs, honest data providers, cryptography, thesis.

Acknowledgements

Here, you can say thank you to your supervisor(s), company advisors and other people that supported you during your project.

Hanna Ek, Gothenburg, May 2021

Contents

List of Figures

List of Tables

1 Introduction

Problem

This paper will investigate the possibility to ensure honest clients in the VAHSS construction where the servers computations are verified using a homomorphic hash function.

Limitations

To concretize and limit the work two major restrictions has been made. First only one construction for VAHSS is considered. Three different constructions for verifying servers computations in AHSS is presented in [16], based on homophobic hash functions, linear homophobic signatures and threshold signature sharing. However only the construction based on homomorphic hash functions will be considered in this paper.

The second limitation is that only range proofs and set membership proofs has been considered as possible methods for verifying the the clients input. Other potential methods could be to consider zero knowledge proofs such as zkSNARK and zkSTARK [2] [?]. Although not investigated here these protocols are seen as potentials methods for verifying the clients in a AHSS construction.

Contribution

Given the problem formulated above and the lets here state contributions and results obtained in this paper:

- Showing that (and how) range proofs and set membership proofs can be combined with a VAHSS construction to verify clients honest.
- Presenting an aggregation of multiple set membership proof, that reduces the verification time when multiple set membership proofs are verified at once.
- Implementing and benchmarking client and server verifiable AHSS, for two different range proofs and one set membership proof and additionally implementing and benchmarking the proposed aggregated version of set membership proof.

Organisation

In chapter 2 the theoretical background is presented, first general cryptographic principles is treated then a more detailed description of the vahss protocol is given followed by a section where different range proofs are explained. In the next chapter both a theoretical and practical evaluation of range proofs is given, then a combination of range proofs and vahss is presented, i.e a server and client verifiable additive homomorphic secret sharing construction, and finally an implementation of this construction in Go is discussed. In the following chapter, chapter 4, runtime results from implementing the presented construction is given. Runtime impact of different parameters such as number clients and range size is also given. Finally in chapter 5 the result obtained is discussed and some conclusions and still remaining questions are given.

2
Theory

This chapter will present the theory used in chapter 3 for constructing a client and server verifiable additive homomorphic secret sharing scheme. This chapter will be structured as follows, first notation, theorems, definitions, assumptions and cryptographic concepts is defined and explained. Then the VAHSS construction [16], that this paper aim to extend such that the construction ensures honest clients, is described. In section 2.3 different construction of methods for verifying that clients are honest is given. More precisely two different constructions for range proofs and one for set membership proofs is presented.

2.1 Preliminaries

This section will introduce cryptographic assumptions, definitions and concepts that will be relevant for the following sections. The purpose of this section is to state information that will be refereed to and used as building blocks for theory presented later in the report.

Notation and setup

To make the text more comprehensible notation that is used throughout the paper is introduced and defined here. Let $\mathbb{F} = \mathbb{Z}_p$ denote a finite field, where p is a large prime and let $\mathbb{G}$ denote a unique subgroup of order q. Define $g \in \mathbb{G}$ to be a group generator and $h \in \mathbb{G}$ a group element such that $log_g\, h$ is unknown and h co-prime to p. The notation $y \in_R \mathbb{Y}$, means that an element y in chosen at random from the set $\mathbb{Y}$.

Definitions, Theorems and Assumptions

In this section definitions, theorems and assumption that will be refereed to and used throughout the thesis is stated. The assumptions given here are the assumptions that all cryptographic constructions in this paper will rely on. The discrete logarithm assumption and the q-strong Diffie Hellman assumption define below does not hold in the presence of quantum computers. Thus the cryptographic constructions presented in this paper are not guaranteed to be secure post quantum.

Definition 1 (Pseudorandom Function (PRF)). *Let S be a distribution over $\{0,1\}^l$ and $F_s : \{0,1\}^m \rightarrow \{0,1\}^n$ a family of functions indexed by a string s in the support S. It is defined that $\{F_s\}$ is a pseudo random function family if, for every*

PPT (probabilistic polynomial time) adversary $\mathcal{A}$, there exists a negligable function ε such that:

$$|Pr[\mathcal{A}^{F_s}(\cdot) = 1] - Pr[\mathcal{A}^{R}(\cdot) = 1]| \leq \varepsilon,$$

where s is distributed according to S and R is a function sampled uniformly at random from the set of all functions mapping from $\{0,1\}^n$ to $\{0,1\}^m$.

Definition 2 (**Euler's totient function**). *The function $\Phi(n)$ is defined as the counter of the number of integers that are relative primes to n in the set $\{1,...,n\}$. Note if n is a prime number $\phi(n) = n - 1$.*

Theorem 1 (**Euler's Theorem**). *For all integers x and n that are co-prime it holds that: $x^{\Phi(n)} = 1\,(mod\ n)$, where $\Phi(n)$ is Euler's totient function.*

From Theorem 1 it follows that for arbitrary y it holds that $x^{y\Phi(n)} = 1\,(\text{mod n})$.

Assumption 1 (**Discrete logarithmic assumption**). *Let $\mathbb{G}$ be a group of prime order q, a generator $g \in \mathbb{G}$ and an arbitrary element $y \in \mathbb{G}$, it is infeasible to find $x \in \mathbb{Z}_q$, such that $y = g^x$*

Assumption 2 (**q-strong Diffie Hellman Assumption**). *Given a group $\mathbb{G}$, a random generator $g \in \mathbb{G}$ and powers $g^x, ..., g^{x_q}$, for $x \in_R \mathbb{F}$ and $q = |\mathbb{G}|$. It is then infeasible for an adversary to find $(c, g^{\frac{1}{x+c}})$, where $c \in \mathbb{F}$.*

Homomorphic Secret Sharing

Homomorphic secret sharing (HSS), first mentioned in [12], hides a secret x by splitting it into shares, such that any subset $\mathcal{S}$ of shares smaller than a threshold τ, i.e $|\mathcal{S}| < \tau$, reviles no information about the original value of x. Let a secret x be split into m shares denoted x_i s.t $i \in \{1,...,m\}$, then to reconstruct the value x at least τ shares has to be combined, this is called a (τ, m)-threshold scheme. In this paper the threshold will be equal to the number of shares, $\tau = m$. A certain type of HSS called *additive* homomorphic secret sharing (AHSS) will be considered, then the secret is to reconstruct by computing the sum of at least τ shares, i.e $x = \sum_{i=1}^{\tau} x_i$.

Homomorphic hash functions

Let $\mathcal{H}$ be a cryptographic hash function, $\mathcal{H} : \mathbb{F} \mapsto \mathbb{G}$. Any such function should satisfy the following two properties:

- **Collision-resistant** It should be hard to find $x, x' \in \mathbb{F}$ such that $x \neq x'$ and $\mathcal{H}(x) = \mathcal{H}(x')$.
- **One-Way** It should be computationally hard to find $\mathcal{H}^{-1}(x)$.

A homomorphic hash function should also satisfy the following property:

- **Homomorphism** For any $x, x' \in \mathbb{F}$ it should hold that $\mathcal{H}(x \circ x') = \mathcal{H}(x) \circ \mathcal{H}(x')$, where $\circ$ is either "$+$" or "$*$".

A function satisfying the thee properties is $\mathcal{H}_1(x) : \mathbb{F} \mapsto \mathbb{G}$ and $\mathcal{H}_1(x) = g^x$ [17].

Pedersen Commitment scheme

The *Pedersen commitment scheme* is a commitment to a secret $x \in \mathbb{F}$, defined as $\mathbb{E}(x, R) = g^x h^R$, where $R \in_R \mathbb{F}$, [11]. The Pedersen commitment satisfies the following theorem;

Theorem 2. *For any $x \in \mathbb{F}$ and for $R \in_R \mathbb{F}$, it follows that $\mathbb{E}(x, R)$ is uniformly distributed in $\mathbb{G}$. If two commits satisfys $\mathbb{E}(x, R) = \mathbb{E}(x', R')$ $x \neq x'$ then it must hold that $R \neq R'$ mod q and*

$$log_g(h) = \frac{x - x'}{R' - R} \; mod \, p \quad \textit{Where } p \textit{ is the prime underlying the field.} \tag{2.1}$$

Proof. The statements of the theorem follows from solving for $log_g(h)$ in $\mathbb{E}(x, R) = \mathbb{E}(x', R')$ □

Theorem 2 implies that if someone knows the discrete logarithm of h with respect to g, i.e $log_g(h)$, this person is able to provide two equal commits, $\mathbb{E}(x, R) = \mathbb{E}(x', R')$ such that $x \neq x'$. Since the $log_g h$ is assumed to be unknown it is impossible to construct two equal commits hiding different secrets. This means that the Pedersen commitment scheme is computational binding under the discrete logarithm assumption, it is also perfectly hiding of the secret x [11].

The Pedersen commitment is homomorphic. Hence for arbitrary messages $x_1, x_2 \in \mathbb{F}$, random values $R_1, R_2 \in_R \mathbb{F}$ and the commits $C_i = \mathbb{E}(x_i, R_i)$, $i \in \{1, 2\}$, it holds that $C_1 \cdot C_2 = \mathbb{E}(x_1 + x_2, R_1 + R_2)$.

Note that the Pedersen commitment is similarity to the hash function $\mathcal{H}_1$ and the hash function can be seen as a generalisation of the Pedersen commitment.

A Pedersen commitment scheme can also be defined for vectors and is then called *Pedersen vector commitment*. Consider a n dimensional vector $\mathbf{x} \in \mathbf{F}^n$, let $\mathbf{g} = (g_1, ..., g_n) \in \mathbb{G}^n$ and $h \in \mathbb{G}$ where $\mathbb{G}$ is a group of order p as above. A commitment to the vector $\mathbf{x} = (x_1, ..., x_n)$ with the random value $R \in_R \mathbb{F}$ is then defined as $\mathbb{E}(\mathbf{x}, R) = \mathbf{g}^{\mathbf{x}} h^R = h^R \prod_{i=1}^{n} g_i^{x_i}$ and the commitment is a value in the one-dimensional group $\mathbb{G}$.

Bilinear mapping

Bilinear mapping (also commonly called bilinear pairing) maps two group elements from one groups to an element in another group. In this paper admissible bilinear mapping fulfilling Definition 3 will be used. In the definition given here two elements from the same group are mapped. Generally in the definition of admissible bilinear mapping, two elements from different groups are mapped to a third group, i.e $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, but in this paper it will always hold that $\mathbb{G}_1 = \mathbb{G}_2$ and hence the definition is given on this form.

Definition 3 (**Admissible Bilinear Map**)**.** *Let $\mathbb{G}_1, \mathbb{G}_T$ be two multiplicative cyclic groups of prime order p such that there exist an admissible bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$. Let $\mathbb{G}_1^* = \mathbb{G}_1 \backslash \{1\}$. Then the bilinear map e fulfils:*

- *Bilinear: for any group element $g \in \mathbb{G}_1^*$ and $a, b \in \mathbb{Z}_p$,*

$$e(g^a, g^b) = e(g, g)^{ab}$$

- *Non-degenerated:* $e(g, g) \neq 1$
- *The bilinear map is efficiently computable*

The bilinear property of the mapping e will later be used to create digital signatures using Bohen-Boyen signatures scheme. Bohen-Boyen presented a signature scheme that exploits the bilinear property of the mapping e to verify the signatures [3]. In short the scheme works as, the signer knows the secret key x and distributes the public key g^x. To sign a message m the signer uses the secret key, x, and computes $\sigma = g^{1/(x+m)}$. This signature is $q-secure$ to forgery under the q-Strong Diffie Hellman Assumption. Verification is done by checking that $e(\sigma, y \cdot g^m) = e(g, g)$, which holds due to the bilinearity of e.

Zero knowledge proof

Zero-knowledge proofs (ZKP) is a cryptographic primitive that was first presented in [8]. The idea behind a ZKP is that after successfully performing a ZKP a certain statement about a secret x has been verified to be true (or false) without having revealed any other information about the secret x beyond the statement. Here only non-interactive ZKP that ensures proof of knowledge (PoK) is considered. Before closer defining what this means the set up and environment of ZKP protocols are considered. A ZKP consists of two parties a *prover* and a *verifier*, further assume both parties has access to the protocol parameters generated by a set up algorithm and a language $\mathcal{L} \in$ NP, additionally the prover knows a secret $x \in \mathcal{L}$. The prover constructs a proof that x belongs to $\mathcal{L}$, by using a witness w of x. Given the proof the verifier can in polynomial time determine if the proof is valid or not. For a ZKP to be non-interactive means that there is no communication required between the prover and verifier during the construction of the proof. PoK means that the verifier is not only convinced there exist a witness w but also that the prover knows such a witness.

A ZKP should fulfil the thee properties in Definition 4, informally the definitions states that: a correctly constructed proof of an instance $x \in \mathcal{L}$ should be accepted with probability 1, an incorrect proof of an instance $x \notin \mathcal{L}$ should have a negligible probability of being accepted and the verifier should learn nothing about the secret beyond the statement being proved.

Definition 4. *First define the two algorithms;* $\texttt{Prove}(x, w)$*,the algorithm for generating a ZKP of instance* $x \in \mathcal{L}$ *and witness* w*, and* `Verify`*, the verification algorithm of the ZKP. Such a ZKP scheme should fulfil the three properties:*

- ***Completeness*** *Given a witness* w *satisfying the instance* $x \in \mathcal{L}$*, it should hold that* $\texttt{Verify}(\texttt{Prove}(x, w)) = 1$.
- ***Soundness*** *If the witness* w *does not satisfy the instance* $x \notin \mathcal{L}$*, then the probability* $Prob[\texttt{Verify}(\texttt{Prove}(x, w)) = 1] < \varepsilon$*, for a sufficiently small* ε.
- ***Zero-knowledge*** *A proof system is honest verifier zero-knowledge if there exist a PPT algorithm* `Simulator` *having access to the same input as the algorithm* `Verify` *but not the provers input, such that output from the Simulator and* `Prove` *is indistinguishable, i.e have the same distribution given that* $x \in \mathcal{L}$.

This paper will consider zero knowledge range proof (ZKRP) and zero knowledge set membership proofs (ZKSM) where the statement that the prover convinces the verifier of is that the secret belongs to a predetermined range or set.

Fiat-Shamir heuristic

Fiat-Shamir heuristic [1] can be used to convert an interactive protocol into non-interactive, here it will be used to construct non-interactive ZKP. A non interactive ZKP requires no communication between the prover and verifier during the construction of the proof. In interactive constructions the verifier sends a challenge $c \in_R \mathbb{F}$ to the prover that is included in the proof in order to convince the verifier that the prover did not cheat. The Fiat-Shamir heuristic replaces the random challenge sent by the verifier with the output of a hash-function of the partial-proof up to this point. The Fiat-Shamir heuristic converts an interactive ZKP to non-interactive while preserving security and full zero-knowledge relying on the random oracle model (ROM) [1] .

2.2 Verifiable additive homomorphic secret sharing

In this section the construction of the verifiable additive homomorphic secret sharing (VAHSS), that this paper aim to extend is presented. The goal is to extend the VAHSS such that it in addition to the verification of servers computations client are also verified to be honest. This will be done by verifying that they do not provide false data. Different methods for verifying the data provided by the clients is explained in the next section.

Before further description of the VAHSS construction, general set up and properties of AHSS and VAHSS is given. It will be assumed that n clients and m servers participated in the construction. To simplify notation the two sets $\mathcal{N} = \{1, ..., n\}$ and $\mathcal{M} = \{1, ..., m\}$ are introduced. The clients are denoted c_i for $i \in \mathcal{N}$ and their respective data x_i. The servers in the construction will be refered to $s_j,\ j \in \mathcal{M}$.

For AHSS constructions, the idea is that each client splits their secret, x_i, into m shares, denoted x_{ij}. The clients then sends one share to each server. The servers receives shares from all n clients and computes the partial sum $y_j = \sum_{i=1}^{n} x_{ij}$ and publishes the result. The final sum can then be computed by any party by summing the public partial sums, this gives $y = \sum_{j=1}^{m} y_j = \sum_{i \in \mathcal{S}} x_i$. For a VHASS construction, a proof σ that verifies that $y = \sum_{j=1}^{n} y_j = \sum_{j=1}^{m} \left(\sum_{i=1}^{n} x_{ij} \right) = \sum_{i=1}^{n} \left(\sum_{j=1}^{m} x_{ij} \right) = \sum_{i=1}^{n} x_i$ is generated and published. This allows any party to verify the correctness of the servers computations.

Construction

The considered VAHSS, uses homomorphic hash functions to verify the servers computations, it was presented in [16] and then implemented and evaluated in [15].

The shares that hides the secrets are constructed by making use of a certain polynomial, p_i. For each client, c_i, let $\theta_{i1}, ..., \theta_{im} \in \mathbb{F} \backslash \{0\}$ and $\lambda_{i1}, ..., \lambda_{im} \in \mathbb{F}$ be chosen such that the following property for polynomial p_i holds,

$$p_i(0) = \sum_{j=1}^{m} \lambda_{ij} p_i(\theta_{ij}). \tag{2.2}$$

Then if the the shares for client c_i are put to $x_{ij} = \lambda_{ij} p_i(\theta_{ij})$ and the polynomial $p_i(X)$ is a t-degree polynomial defined as $p_i(X) = x_i + \sum_{k=1}^{t} a_k X^k$. The sum of all shares is , $\sum_{j=1}^{m} x_{ij} = \sum_{j=1}^{m} \lambda_{ij} p_i(\theta_{ij}) = p_i(0) = x_i$, thus the shares x_{ij} adds up to the secret x_i and the secret is perfeclt hidden is not all shares used in the summation.

This idea is used in the VAHSS construction 1. The construction consists of the six PPT algorithms: **ShareSecret**, **PartialEval**, **PartialProof**, **FinalEval**, **FinalProof** and **Verify**. The clients providers executed the step **ShareSecret**, the servers **PartialEval** and **PartialProof** and the last three steps can be ran by anyone. A full description of the construction and all six algorithms is seen in Construction 1.

Construction 1 : Verifiable additive homomorphic secret sharing

Goal: Construct and share the sum $\sum_{i=1}^{n} x_i$, where x_i is a secret value known by client c_i, where $i \in \mathcal{N}$ without any client needing to revealing their individual secret. The servers, used to sharing the secrets, computations are verified so they must be honest.

- **ShareSecret** $(1^\lambda, i, x_i) \to (\tau_i, \{x_{ij}\}_{j \in \mathcal{M}})$
  Pick uniformly at random $\{a_i\}_{i \in \{1,..,t\}} \in \mathbb{F}$ and a t-degree polynomial p_i on the form $p_i(X) = x_i + a_1 X + ... + a_t X^t$. Let $\mathcal{H} : x \mapsto g^x$, be a collision-resistant homomorphic hash function. Let $R_i \in \mathbb{F}$ be the output of a PRF. Where it is required that $R_n \in \mathbb{F}$ satisfies $R_n = \phi(N) \lceil \frac{\sum_{i=1}^{n-1} R_i}{\phi(N)} \rceil - \sum_{i=1}^{n-1} R_i$. Compute $\tau_i = \mathcal{H}(x_i + R_i)$, and put $x_{ij} = \lambda_{i,j} p_i(\theta_{ij})$. Output τ_i and $x_{i,j}$ for $j \in \mathcal{M}$.
- **PartialEval** $(j, \{x_{ij}\}_{i \in \mathcal{N}}) \to y_j$
  Compute and output $y_j = \sum_{i=1}^{n} x_{ij}$.
- **PartialProof** $(j, \{x_{ij}\}_{i \in \mathcal{N}}) \to \sigma_j$
  Compute and output $\sigma_j = \prod_{i=1}^{n} g^{x_{ij}} = g^{\sum_{i=1}^{n} x_{ij}} = g^{y_j} = \mathcal{H}(y_j)$.
- **FinalEval** $(\{y_j\}_{j \in \mathcal{M}}) \to y$
  Compute and output $y = \sum_{j=1}^{m} y_j$.
- **FinalProof** $(\{\sigma_j\}_{j \in \mathcal{M}}) \to \sigma$
  Compute and output $\sigma = \prod_{j=1}^{m} \sigma_j = \prod_{j=1}^{m} g^{y_j} = g^{\sum_{j=1}^{m} y_j} = g^y = \mathcal{H}(y)$.
- **Verify** $(\{\tau_i\}_{i \in \mathcal{N}}, \sigma, y) \to \{0, 1\}$
  Compute and output $\sigma = \prod_{i=1}^{n} \tau_i \wedge \prod_{i=1}^{n} \tau_i = \mathcal{H}(y)$.

A HSS and a AHSS construction should satisfy two requirements: *Correctness* and *Security.* A VAHSS should also satisfy *Verifiability.* The exact definition of the three requirements for Construction 1 is given in [16] and Theorem 3 states that Construction 1 fulfils these requirements.

Theorem 3. *Construction 1 satisfies the correctness, security and verifiability requirements defined in [16].*

Proof. See section 4.1 in [16]. □

2.3 Constructions for verifying clients input

Range proofs allows a prover to convince a verifier that the value of a secret is in an allowed range. Zero knowledge range proofs (ZKRP) does this without revealing any other information about the secret beyond the fact that the secret belongs to the range. Formally the ZKRP presented in this paper are constructed to prove the following statement about a secret x:

$$\{(g, h \in \mathbb{G}, C; x, R \in \mathbb{F}) \; : \; C = g^x h^R \wedge x \in \{\text{"\textit{predetermined allowed range}"}\}. \tag{2.3}$$

Zero knowledge set memberships proof (ZKSM) will also be considered, they prove that a secret belongs to a set, instead of a range, which does not need to be continuous. Formally (ZKSM) prove the following statement:

$$\{(g, h \in \mathbb{G}, C; x, R \in \mathbb{F}) \; : \; C = g^x h^R \wedge x \in \Phi\}, \tag{2.4}$$

where Φ is some known set.

Note that in the above statements it is assumed that x is the secret hidden in a Pedersen commitment, which is not a general requirement for range proofs and set membership proofs however only such proofs will be studied in this paper. The range which x is proved to belong to in ZKRPs may vary between different constructions and will be more precisely defined below for the separate constructions.

There exists several constructions for range proofs and set membership proofs however this paper will investigate two different range proof constructions and one set membership construction. These constructions will then in the next chapter be investigated if they can be combined with the VAHSS-construction described above to ensure clients honesty in the VHASS construction.

In the two subsections below the theory and construction of *Set membership proofs & Signature based range proofs* and *Bulletproofs* are presented. All proofs satisfies the three conditions *completeness*, *soundness* and *zero-knowledge* stated in Definition 4 and proves a statement on the form given in equation (2.3) or (2.4).

2.3.1 Set membership proof and Signature-based range proof

First a ZKSM is described and then it will be extended to a ZKRP refereed to as *signature-based range proof*, the idea of these two construction was originally presented in [6]. Both the ZKSM and ZKRP constructions are modified compared to the original construction according to the Fiat-Shamir heuristic to be non-interactive.

Set membership proof

The idea behind the ZKSM (and also the later derived ZKRP) is that for each element in the allowed set Φ there exist a public commitments, denoted A_i, $\forall i \in \Phi$. These commitments are made public in the set-up phase by the verifier or some other party (not the prover). The prover aims to prove that the secret hidden by a pre-published Pedersen commitment, denoted C, is in the allowed set Φ. To prove this the public commitment representing the secret x, i.e A_x is selected and used to compute the value $V = A_x^\tau$, where $\tau \in_R \mathbb{F}$. The prover has to convince the verifier that 1) the published value V is indeed equal to A_x^τ where A_x is a signature of an element from the allowed set 2) the secret in the Pedersen commitment C is the same as the secret hidden by V. In Construction 2 a detailed description of the PPT algorithms constructing the ZKSM described above is explained. The notation $e(\cdot,\cdot)$ in Construction 2 and 3 refers to an admissible bilinear mapping as defined previously in section 2.1.

Construction 2 : Non interactive set membership proof

Goal: Given a Pedersen commitment $C = g^x h^R$ and a set Φ, prove that the secret x in the commitment belongs to the set Φ without revealing anything else about x.

- **SetUp** $(g, h, \Phi) \to (y, \{A_i\}_{i\in\Phi})$
  Pick uniformly at random $\chi \in_R \mathbb{F}$. Define $y = g^\chi$ and $A_i = g^{\frac{1}{\chi+i}}\ \forall i \in \Phi$, output y and $\{A_i\}_{i\in\Phi}$.
- **Prove** $(g, h, C, \Phi) \to$ $\mathit{proof}_{SM} = (V, a, D, z_x, z_\tau, z_R)$
  Pick uniformly at random $\tau \in_R \mathbb{F}$, choose from the set $\{A_i\}$ the element A_x and calculate $V = A_x^\tau$. Pick uniformly random three values $s, t, m \in_R \mathbb{F}$. Put $a = e(V, g)^{-s} e(g, g)^t$ and $D = g^s h^m$. Then use these two values to compute the challenge $c = \text{Hash}(a, D)$. Then given this challenge compute $z_x = s - xc$, $z_R = m - Rc$ and $z_\tau = t - \tau c$ then construct and publish the proof, $\mathit{proof}_{SM} = (V, a, D, z_x, z_t au, z_R)$.
- **Verify** $(g, h, C, \mathit{proof}) \to \{0, 1\}$ Check if $D \overset{?}{=} C^c h^{z_R} g^{z_x} \wedge a \overset{?}{=} e(V, y)^c e(V, g)^{-z_x} e(g, g)^{z_\tau}$. If the equality holds the prover has convinced the verifier that the secret $x \in \Phi$ thus return 1 otherwise return 0.

Signature-based range proofs

The ZKSM construction can be turned into a efficient zero knowledge range proof by rewriting the secret x in base u such that,

$$x = \sum_{j=0}^{l-1} x_j u^j.$$

Optimal choice of the two parameters u, l is discussed in [6]. Using this notation it follows that if $x_j \in [0, u)\ \forall j \in \mathbb{Z}_l$, then $x \in [0, u^l)$. A remark is that the subscript j goes between the number $[0, l-1]$ and not $[0, l]$. Construction 3 is a modification of construction 2 into a non interactive zero knowledge range proof using the above decomposition of the secret x.

This ZKRP construction can be generalised to prove membership to an arbitrary interval $[a, b]$ where $a > 0$ and $b > a$, by showing that $x \in [a, a+u^l)$ and $x \in [b-u^l, b)$, since then the secret belong to the range $[a, b]$. Figure 2.1 illustrates the intuition and correctness of the statement. Proving $x \in [a, a+u^l)$ and $x \in [b-u^l, b)$ can easily be transferred into proving $x - a \in [0, u^l)$ and $x - b + u^l \in [0, u^l)$, since both a, b are public. Therefore to prove a secret is in an arbitrary interval the steps **Prove** and **Verify** in construction 3 will have to be executed twice. Plus the **Verify** algorithm has to be modified to include an `AND` operation to verify that

Construction 3 : Non interactive range proof

Goal: Given a Pedersen commitment $C = g^x h^R$ and two parameters u, l, prove that the secret $x = \sum_{j=0}^{l} x_j u^j$ belongs to the interval $[0, u^l)$ without revealing anything else about x.

- **SetUp** $(g, h, u, l) \rightarrow (y, \{A_i\}_{i \in \mathbb{Z}_u})$
  Pick uniformly at random $\chi \in_R \mathbb{F}$. Define $y = g^\chi$ and $A_i = g^{\frac{1}{\chi + i}}\ \forall i \in \mathbb{Z}_u$, output y and $\{A_i\}$.
- **Prove** $(g, h, C, u, l) \rightarrow\ proof_{RP} = (\{V_j\}, \{a_j\}, D, \{z_{x_j}\}, \{z_{\tau_j}\}, z_R)^1$
  For every $j \in \mathbb{Z}_l$: pick uniformly at random $\tau_j \in_R \mathbb{F}$ and compute $V_j = A_{x_j}^{\tau_j}$. Then pick uniformly at random three more values $s_j, t_j, m_j \in_R \mathbb{F}$ and compute $a_j = e(V_j, g)^{-s_j} e(g, g)^{t_j}$ for all $j \in \mathbb{Z}_l$ and $D = \prod_{j \in \mathbb{Z}_l} (g^{u^j s_j}) h^{m_j}$ Given this let $c = \text{Hash}(\{a_j\}, D)$. Then for all $j \in \mathbb{Z}_l$ compute $z_{x_j} = s_j - x_j c$,$z_{\tau_j} = t_j - \tau_j c$ and $z_R = m - Rc$, where $m = \sum_{j \in \mathbb{Z}_l} m_j$. Finally output the proof: $proof_{RP} = (\{V_j\}, \{a_j\}, D, \{z_{x_j}\}, \{z_{\tau_j}\}, z_R)$
- **Verify** $(g, h, C, proof) \rightarrow \{0, 1\}$
  Check if $D \overset{?}{=} C^c h^{z_R} \prod_{j \in \mathbb{Z}_l} (g^{u^j z_{x_j}}) \wedge a_j \overset{?}{=} e(V_j, y)^c e(V_j, g)^{-z_{x_j}} e(g, g)^{z_{\tau_j}}$ for all $j \in \mathbb{Z}_l$. If the equality holds the prover has convinced the verifier that $x \in [0, u^l)$ return 1 otherwise return 0.

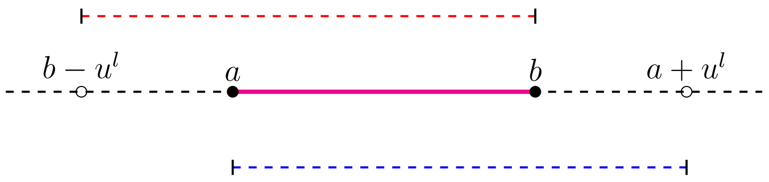


Figure 2.1: Illustration of generalisation to arbitrary intervals $[a, b]$ for range proofs

$x - a \in [0, u^l) \wedge x - b + u^l \in [0, u^l)$. In [7] an optimised implementation is presented reducing the complexity with a factor 2. This rather small reduction is important when a verifier is required to check the range of multiple secret simultaneously.

2.3.2 Bulletproofs

Bulletproofs is a state of the art range proof that is integrated in many real-world protocols, for example they are often used in in cryptocurrencies. In the following sections the concepts behind Bulletproofs is presented.

Notation and Set Up

Before describing the construction of Bulletproofs, some additional notation is introduces. Note that here the same notation that is used in original Bulletproof paper, will be used, hence more details of the notations can bee seen in [5] .

In this section n will denote the dimension of the room not the number of clients as earlier. Further remark that, the dimension of the room is the length of the bit representation of the secret in the Pedersen vector commitment, potentially padded with zeros.

Both in the construction of the inner product argument and the Bulletproof the parameters $g, h, \mathbf{g}, \mathbf{h}, u$ are assumed to be pre-shared and known by both verifier and prover. The group elements g, h are as before. Further the two vectors $\mathbf{g}, \mathbf{h} \in \mathbb{G}^n$ are assumed to be independent generators of the space $\mathbb{G}^n$. The variable $u \in \mathbb{G}$ is such that there is no known discrete logarithmic relation between $\mathbf{g}, \mathbf{h}$. In order to ensure the fairness and correctness of the parameters $g, h, \mathbf{g}, \mathbf{h}, u$ they can be assumed to be chosen by some trusted third party. Another possibility that drops the assumption of a trusted setup is to use the *Nothing Up My Sleeve* (NUMS) strategy, [9].

Inner product argument

The Bulletproof construction is based on the inner product argument which will be closer presented in this section. The inner product argument is an argument of knowledge that $\mathbf{s}, \mathbf{q}$ in a Pedersen vector commitment $P_v = \mathbf{g}^{\mathbf{s}}\mathbf{h}^{\mathbf{q}}$ satisfies a given inner product denoted c. More formally the argument is a proof system of the statement,

$$\{(\mathbf{g}, \mathbf{h} \in \mathbb{G}^n,\ P_v \in \mathbb{G},\ c \in \mathbb{F};\ \mathbf{s}, \mathbf{q} \in \mathbb{F}^n) :\ P_v = \mathbf{g}^{\mathbf{s}}\mathbf{h}^{\mathbf{q}} \wedge\ c = \langle \mathbf{s}, \mathbf{q} \rangle\}$$

Which can be shown to be equivalent to a proof of the statement,

$$\{(\mathbf{g}, \mathbf{h} \in \mathbb{G}^n,\ u, P_v \in \mathbb{G};\ \mathbf{s}, \mathbf{q} \in \mathbb{F}^n) :\ P_v = \mathbf{g}^{\mathbf{s}}\mathbf{h}^{\mathbf{q}}u^{\langle \mathbf{s}, \mathbf{q} \rangle}\}.$$

A logarithmic sized proof of the above inner product statement is presented in Construction 8 in Appendix B. The presented construction is modified compared to the one presented in [5] to be non-interactive using the Fiat-Shamir heuristic.

Remark that the inner product argument is sound but not zero-knowledge since information about two exponentials $\mathbf{s}, \mathbf{q}$ is reviled.

Bulletproofs construction

Here the logarithmic sized range proof called *Bulletproof*, based on the inner product argument, is presented. This construction allows a prover, given a Pedersen commitment $C = g^x h^R$, of the secret x, to convince a verifier that the secret belongs to the interval $[0, 2^n)$. This is achieved by convincing the verifier that $\boldsymbol{x} \in \{0,1\}^n$ is the binary representation of the secret x, or equivalently that $x = \langle \boldsymbol{x}, \mathbf{2}^n \rangle$ and that the prover knows $\boldsymbol{x}$. This can be shown to be done by proving the following statement;

$$\left\langle \boldsymbol{x} - z \cdot \mathbf{1}^n, \boldsymbol{y}^n \circ (\bar{\boldsymbol{x}} + z \cdot \mathbf{1}^n) + z^2 \cdot \mathbf{2}^n \right\rangle = z^2 \cdot x + \delta(y, z), \tag{2.5}$$

where $\bar{\boldsymbol{x}}$ is the component-wise complement of $\boldsymbol{x}$ and $\delta(y, z) = (z - z^2) \cdot \langle \mathbf{1}^n, \boldsymbol{y}^n \rangle - z^3 \langle \mathbf{1}^n, \mathbf{2}^n \rangle \in \mathbb{F}$. The values z and y are either chosen at random from the set $\mathbb{F}$ by the verifier in an interactive construction or is the output of a hash function in a non-interactive construction. Here a non-interactive construction will be considered.

Directly using an inner product argument presented in Construction 8 to prove the statement in equation (2.5) would leak information about x, since information about the two vectors $\boldsymbol{x}, \bar{\boldsymbol{x}}$ is revealed, i.e the binary representation of x. Hence two new vectors $\boldsymbol{s}_1, \boldsymbol{s}_2$ are introduced and will serve as blinding vectors and help construct a zero-knowledge range proof. Note that this results in that the bulletproof is zero-knowledge despite that the inner product argument is not a zero knowledge construction. Given this idea, the inner product in (2.5) is tweaked to include the two blinding vectors and the new statement is to prove the inner product of is,

$$\begin{aligned} t(X) &= \langle l(X), r(X) \rangle = t_0 + t_1 \cdot X + t_2 \cdot X^2 \\ l(X) &= \boldsymbol{x} - z \cdot \mathbf{1}^n + \boldsymbol{s}_1 \cdot X \\ r(X) &= \boldsymbol{y}^n \circ (\bar{\boldsymbol{x}} + z \cdot \mathbf{1}^n + \boldsymbol{s}_2 \cdot X) + z^2 \cdot \mathbf{2}^n, \end{aligned}$$

Note that $t_0 = z^2 \cdot x + \delta(y, z)$ which is equal to the right hand side of equation (2.5). Further it holds that $t_1 = \langle \boldsymbol{x} - z \cdot \mathbf{1}^n, \boldsymbol{y}^n \circ \boldsymbol{s_R} \rangle + \langle \boldsymbol{s_L}, \boldsymbol{y}^n \circ (\boldsymbol{a_R} + z \cdot \mathbf{1}^n) + z \cdot \mathbf{2}^n \rangle$ and $t_2 = \langle \boldsymbol{s_L}, \boldsymbol{y}^n \circ \boldsymbol{s_R} \rangle$. Given these vectors, Construction 4 gives a non interactive zero knowledge range proof that the secret x belongs to the interval $[0, 2^n]$.

Construction 4 : Bulletproof

Goal: Given a Pedersen commitment $C = g^x h^R$ and a number n, prove that the secret x in the commitment belongs to the range $[0, 2^n)$ without revealing anything else about x.

- **Prove** $(g, h, \mathbf{g}, \mathbf{h}, P, n, x, R, u) \rightarrow$ *proof*$_{RP}$
  Let $\boldsymbol{x}$ denote the binary representation of the secret x in the commitment P and $\bar{\boldsymbol{x}}$ the component-wise complement such that $\boldsymbol{x} \circ \bar{\boldsymbol{x}} = 0$. Construct the commitment $A = h^{\alpha} \boldsymbol{g}^{\boldsymbol{x}} \boldsymbol{h}^{\bar{\boldsymbol{x}}}$, where $\alpha \in_R \mathbb{F}$. Then chose the two blinding vectors $\boldsymbol{s_R}, \boldsymbol{s_L} \in_R \mathbb{F}^n$ and the value $\rho \in_R \mathbb{F}$ and compute the commitment $S = h^{\rho} \boldsymbol{g}^{\boldsymbol{s_L}} \boldsymbol{h}^{\boldsymbol{s_R}}$. Let $y = \text{Hash}(A, S)$, $z = \text{Hash}(A, S, y)$ and $\tau_1, \tau_2 \in_R \mathbb{F}$. Now the it is possible to construct t_1, t_2 defined above. Given this let $T_1 = g^{t_1} h^{\tau_1}$ and $T_2 = g^{t_2} h^{\tau_2}$, next let $X = \text{Hash}(T_1, T_2)$. Now construct the two vectors for the inner product argument: $\boldsymbol{l} = \boldsymbol{x} - z \cdot \mathbf{1}^n - \boldsymbol{s_L} \cdot X$, $\boldsymbol{r} = \boldsymbol{y}^n \circ (\bar{\boldsymbol{x}} + z \cdot \mathbf{1}^n + \boldsymbol{s_R} \cdot X) + z^2\ X$ and calculate the inner product $\hat{t} = \langle \boldsymbol{l}, \boldsymbol{r} \rangle$. Finally compute $\tau_X = \tau_2 x^2 + \tau_1 X + z^2 R$ and $\mu = \alpha + R\,X$. Now use the inner product argument to prove that $\hat{t}$ is indeed the inner product of the two vectors $\mathbf{l}, \mathbf{r}$ using the commitment $P_v = \boldsymbol{g}^{\boldsymbol{l}} \boldsymbol{h}^{\boldsymbol{r}}$, run the algorithm **Prove** defined Construction 8 with the input $(\boldsymbol{g}, \boldsymbol{h}, u, P_v, \hat{t}, \boldsymbol{l}, \boldsymbol{r})$ to construct such a proof denoted *proof*$_{IP}$. Combine and publish the proof: *proof*$_{RP} = (\tau_X, \mu, \hat{t}, P, A, S, T_1, T_2, P_v, \textit{proof}_{IP})$.
- **Verify** $(g, h, C, \textit{proof}_{RP}) \rightarrow \{0, 1\}$
  Compute the three hash functions $y = \text{Hash}(A, S)$, $z = \text{Hash}(A, S, y)$ and $X = \text{Hash}(T_1, T_2)$. Then given y, z, X compute $h'_i = h_i^{y^{-i+1}}$ for all $i \in \{1, ..., n\}$, $P_l = P \cdot h\mu$ and $P_r = A \cdot S^x \boldsymbol{g}^{-z} \boldsymbol{h'}^{z\boldsymbol{y}^n + z^2 \cdot \mathbf{2}^n}$. Then check if the following equalities hold: $P_l \overset{?}{=} P_r \wedge g^{\hat{t}} h^{\tau_X} \overset{?}{=} P^{z^2}\ g^{\delta(y,z)}\ T_1^x\ T_2^{x^2}$ and if the output of **Verify** in Construction 8 on the input (proof_{IP}) is 1. If all three criterion is fulfilled then the secret x in the commitment P is in the range $[0, 2^n]$.

3
Methods

The purpose of this chapter is, based on the theory given in the previous chapter present an extended VAHSS construction that ensures honest clients by verifying that their inputs is from an allowed range or set. This will be done by first evaluating the performance of the range proofs and set membership proofs discussed the previous chapter to get an understanding of their advantages and disadvantages. Then determine which are more suitable to use in a extended VAHSS construction. In the section 3.2 details on how to construct a server and client verifiable AHSS scheme is discussed. Section 3.3 investigates the possibility of improving of the combined construction by aggregating the clients individual range proofs into one combined range proof. Then the final section discuss details about how to implement the construction presented in section 3.2.

3.1 Comparison of constructions for verify clients honesty

In this section the different constructions for verifying clients honesty presented in section 2.3 will be compared and discussed based on the purpose of including them the VAHSS Construction 1 . This is to obtain an understanding of their suitability to be combined with the VAHSS scheme to verify clients honesty. The aspects that is of interest in when evaluating the compatibility with the VAHSS construction are:

- Computation complexity
- Proof size (communication complexity)
- Flexibility of range

The computational complexity affects the runtime, i.e it is important since it gives information about the runtime and what parameters that affects the runtime. Remark that the computational complexity in the verification will become important when used in a VAHSS construction. There are usually many clients participating and thus the verification will grow linear with the number of clients. This is further discussed later in section 3.3. The size of the proof affects how many bits that needs to be sent by the prover to the verifier, this is not the most crucial aspects here, but it is still desired to keep this as low as possible. Finally the flexibility of the range determines the application areas, if for example a range proof can only be used to prove that a value is between $[0, 10]$, and no other ranges. The application areas for such a range proof is highly limited. Further discussion will be made in the next sections where the considered range proofs and set membership proofs are compared.

Remark that all of the range proofs and set membership proofs considered aims to prove that the secret in a Pedersen commitment is in an allowed range (or set). Thus to combine any such proof with the VAHSS construction, the clients needs to publish a Pedersen commitment of their secret x_i. This is investigated further in section 3.2 and it will be shown that the adaptation of the VAHSS construction to include a range proof is the same independent of the range proof used, hence the adaptability to VAHSS in not considers in the following discussion and evaluation.

3.1.1 Theoretical analysis

In this section the proofs will be discussed theoretically and then in the following section runtime for prototype implementations is discussed.

A disadvantage for set membership proofs is the time consuming Setup phase, this phase has computational complexity of $\mathcal{O}(n)$, where $n = |\Phi|$. This comes from that the set membership proof requires digital signatures to be known by both prover and verifier, one signature for each elements in Φ. The signatures are usually shared by the verifier in the Setup phase. Sharing the digital signatures of the elements in the set Φ becomes intractable when the set is large. If it can be assumed that these signatures has been pre-shared, for example in applications where the same set of signatures are used many times, then the set membership can still be competitive to range proofs for applications requiring large sets. Large here refers to sets of size > 100.

The computational complexity will not be compared further theoretically since the Bulletproofs are considerably different compared to the set membership proof and signature based range proof. The latter requires bilinear mappings unlike Bulletproofs and bilinear mappings are comparatively expensive operations compared to group exponentials which are the operation dominating the computational complexity for Bulletproofs. Instead of comparing the theoretical computational complexity the runtime for prototype implementation of the different proof is considered in the next section.

The communicational complexity of the proofs is not the most relevant feature of comparison for the aims of this paper, but it is still an important factor and hence the proof size is briefly discussed for the different proofs. Given that the digital signatures are pre-shared the set membership provides a $\mathcal{O}(1)$-size set membership proof. The proof size for the signature based range proof is $\mathcal{O}\Big(\frac{k}{log\,k - log\,log\,k}\Big)$, where $l = \frac{k}{log\,u}, u = \frac{k}{log\,k}$ and for Bulletproofs $\mathcal{O}(log_2 n)$. Thus it can be realised that the proof size for set membership is asymptotically smallest and signature based range proofs largest.

An advantage of the set membership is that it allows to prove that a secret belongs to a set instead of a range and this is much more general. For example a set can be all prime numbers from $1 - 1000$, all odd numbers or just a random set of numbers. While the signature-based range proof and Bulletproof can only be use to prove belonging to a continuous range. In the paper presenting signature-based range proof is seen that a secret can be proven to belong to arbitrary ranges $[a, b]$,where that $a > 0$ and $b > a$, and not only ranges where the lower bound is 0. Considering arbitrary boundaries leads to a doubling of the computational complexity and communication complexity.

Bulletproofs can also be modified to allow arbitrary range, $[a, b]$, with a similar approach as presented for the signature based range proofs and illustrated in Figure 2.1. A Bulletproof where one prover wishes to verify the range of several commitments can be optimised such that the proof size does not growing multiplicatively in the number of commits but instead grows additive. More precisely, assume a prover wants to prove the range of k commits a naive implementation would lead to a proof size of $k \cdot log_2 n$. An optimised implementation reduces this to $log_2 n + 2log_2 k$. Hence for the case of arbitrary intervals proof size is increased with the additive term $2log_2 2 = 2$.

Both for signature-based range proof and for Bulletproofs it is not the size of interval that a secret is proved to belong to ($[a, b]$) that determines the complexity, it is the size of u^l respective 2^n in the constructions. This is seen by how the bounds a, b are just used for rewriting the secret and does not accurately affect the construction. It is required that $b < u^l$ for the signature based range proof and $b < 2^n$ for the Bulletproofs. Thus the complexity is not dependent on the size of the range but rather the upper limit, this leads to that a small range of large numbers results in longer runtime then a larger range of smaller numbers. Set membership proofs does not have this property since the runtime for the proof construction and verification is independent of the size and values of the elements in the set.

3.1.2 Prototype Analysis

Implementation of Bulletproofs and signature-based range proof has been done and compared between themselves in [6]. That comparison does not include results about the runtime for the set membership proof. In order to obtain a fair comparison between Bulletproofs, signature-based range proofs and set membership proofs the code used by [6] is benchmarked for all three constructions on the same hardware.

he reason for redoing the benchmarking for Bulletproof and signature based range proofs is since hardware differences would not lead to a fair comparison. Table 3.1 shows the time complexity comparison between Bulletproofs, signature-based range proofs and set membership proofs implemented in Golang (Go) with 128- bit security level. The settings for the benchmarking is the same as in [6] and the code obtain at [10] is unchanged. The only difference compared to the comparison in [6] is the hardware used and that set memberships proof are included in the comparison.

The set used in the set membership proofs contains 182 elements, which thereby the same amount of elements in the set as in the range, which is $[18, 200]$, for the two range proofs The computer used has a 1.6 GHz Dual-Core Intel Core $i5 - 5250U$ CPU, 8GB 1600 MHz DDR3 RAM and running macOS 10.15.

In Table 3.1 it is clear that the signature based range proof is much slower than Bulletproofs, especially in the verification algorithm. In applications with many clients the verification runtime is important since a verifier has to verify all clients. Bulletproofs has slightly faster runtime compared to Set memberships proofs in the verification algorithm, although their performance is still comparable and the set membership has he advantage that the runtime is indepenent of the size of the set, unlike Bulletproofs. This concludes that the Bulletproofs and set membership proofs

Table 3.1: Time complexity comparison for Bullerproofs, signature-based range proofs and set membership proofs. The benchmarking is done as are described in section 3.1.2. The runtime are for implementations written in Golang and the code s are not optimized in runtime.

	Generate Proof (ms)	Verification (ms)
Bulletproof s	198	79
Signature-based	240	438
Set Membership	66	90

are the best candidates found for combining with the VAHSS, in order to verify the clients.

3.2 Additive homomorphic secret sharing with verification of both clients and servers

The VAHSS constructions discussed in section 2.2 assumes honest clients and verifiers that the servers computations are correct. The aim of this paper is to extended the VAHSS construction to verify both client and servers, without the clients needing to reveal their secrets.

Two methods for verifying clients honesty without revealing the secret values are zero-knowledge range proofs and set membership proofs. If a range proof or set membership proof is included in the VAHSS construction then potentially malicious clients can only have limited influence on the computed sum. Range proofs and set membership proofs forces malicious input to still be part of the range or set. This leads to that the impact on the sum that a malicious client can have is bounded by the size of the range or the elements in the set.

In this section it will be investigated how to combine the range proofs and set membership proofs, presented i section 2.3, with the VAHSS construction presented in section 2.2. Note that to ensure honest clients it is not sufficient to construct and perform a range proof and VAHSS scheme separately. In such a protocol the verifier cannot be sure that the secret proven to be in the allowed range is the same as the secret hidden by the shares.

All considered range proofs and set membership proofs emanate from a Pedersen commitment hiding a secret and generates a zero knowledge proof that this secret belongs to an pre-specified interval or set. Besides this common feature the constructions of the presented range proofs and set membership proofs differ considerably. Therefore the possibility to exploit the Pedersen commitment in order to link the VAHSS construction with a range proof is investigated. More precisely a link between the shares hiding the secrets generated in the algorithm **ShareSecret** in the VAHSS construction and the secrets hidden in the Pedersen commitments used in the range proofs and set membership proofs is desired. Proving such a connection would convince the verifier that the shares represents a secrets that is in the allowed range.

As discussed above, publishing a Pedersen commitment of the secret itself does not provide any guarantee that it is the same secret that is hidden in the shares. It can also be seen that committing to the shares in Pedersen commitments does not ensure the verifier that the secret hidden in the shares are in the allowed set or range. This is since the individual shares themselves does not reveal information about the secret they are hiding. This leads to that there is not guarantee that proving a share belongs to the allowed range (or set) implies that the secret does and the other way assuming that the secret to a range (or set) does not imply that the shares does. Thus some trick to connect the secret in the shares to the secret in the Pedersen commitment must be derived. Therefore that the aggregation of the partial proofs used in the VAHSS construction to prove the honesty of the servers will be used to connect the range proofs to the VAHSS construction.

Recall that the clients in addition to the shares also publishes the checksum τ_i for the secret x_i, further recall that the definition of the checksum is $\tau_i = g^{x_i+R_i}$, where R_i is chosen uniformly at random from the field $\mathbb{F}$. Using this checksum as the Pedersen commitment in the construction of a range proof would be correct. However if $g = h$ the computationally binding property of a Pedersen commitment would not hold since $log_g(h) = log_g(g) = 1$ which leads to that the left hand side in equation (2.1) is equal to 1. Therefore to construct two commits $\mathbb{E}(x, R)$ and $\mathbb{E}(x', R')$ such that $\mathbb{E}(x, R) = \mathbb{E}(x', R')$ but $x \neq x'$ it is sufficient to solve solve for x' in,

$$1 = \frac{x - x'}{R - R'} \, mod \, N.$$

In other words it is straightforward to create a false commitment hence also a false range proof and set membership proof.

Therefore it is instead investigated if the checksum τ_i can be modified into a Pedersen commitment. This leads to that instead of the previously computed checksum τ_i, the clients compute and output $\pi_i = g^{x_i} h^{R_i}$, where x_i, R_i, g, h are as above. Now a range proof can easily be constructed for the commitment π_i. Below it will be shown that Theorem 3 still hold after replacing τ_i with π_i. It remains to argue that this method ensures the verifier that the secret hidden by the shares is the same secret proven to be in the allowed range or set.

Assume that client c_k commits to the value $\hat{x}_k$ in the Pedersen commitment π_k and constructs shares, x_{kj} such that $x_k = \sum_{j\in\mathcal{M}} x_{kj} \neq \hat{x}_k$. This leads to that c_k generates a range proof that the secret hidden in the commitment belongs to the interval $[a, b]$ but the secret hidden by the shares does not necessarily belong to the range or set. Then the verification of the servers will not hold, since $\prod_{i=1}^{m} \pi_i \neq g^y$. this means that the verification will return false and the protocol will not succeed, even if all range proofs verifies true. Thus any cheating party will be detected, it will not be possible do determine which party that cheated and more precisely not even if the cheating party was a client or a server.

In Construction 5 the extended VAHSS, such that the construction ensures honest clients by including a range proof or set membership proof is described in detail. In order to clarify the modifications made to include a verification of the clients, the differences to the VAHSS construction presented in [16] are pointed out. The algorithms **ShareSecret** and **Verify** has been modified, and the algorithms

RangeProof and **GenerateCommitment** have been added. More precisely in the algorithm **ShareSecret** does not output the checksum τ_i, instead the Pedersen commitment π_i is computed in the algorithm **GenerateCommitment**. The algorithm **GenerateCommitment** can be included in either **ShareSecret** or **RangeProof** instead of being viewed as a separate algorithm. In the implementation discussed later the commitment is generated while constructing the range proof and not explicitly. The algorithm **RangeProof** constructs a range proof (or set membership proof) denoted RP_i given the commitment π_i. It is not specified which range proof or set membership proof that is used to verify clients. Which range proof or set membership proof that is used does not affect the rest of Construction 5, as long as it emanate from a Pedersen commitment, hence it is left unspecified. In addition to the steps in the algorithm **Verify** in construction 1, the algorithm **Verify** verifies the correctness of the range proof RP_i and an additional `AND` operator to compute the total verification.

The algorithms **GenereateCommitment** and **RangeProof** are executed by the clients and the other algorithms are executed by the same party as in the VAHSS construction in [16].

Theorem 4. *The client and server verifiable AHSS presented in Construction 5 satisfies the correctness, security and verifiability requirements described in section 2.2, by replacing τ_i with π_i. Additionally it also satisfies the following extension of the verification requirement:*

Let $\mathcal{A}$ denote any PPT adversary and T denote the set of corrupted clients. The extended verifiability property requires that any $\mathcal{A}$ who can modify the Pedersen commitments π_i to any $\pi_i^{'}$ $\forall i \in T$ has a negligible probability at choosing a commitment $\pi_i^{'}$ such that Verify$(\{\pi_i^{'}\}_{i\in\mathcal{N}}, x, y, \{RP_i\}_{i\in\mathcal{N}}) = 1$.

Proof. To argue that that the correctness, security and verifiability properties for the server verifiable AHSS still holds after replacing τ_i with π_i for $i = 1, ..., n$, it is noted that,

$$\prod_{i=1}^{n} \pi_i = \prod_{i=1}^{n} g^{x_i} h^{R_i} = g^{\sum_{i=1}^{n} x_i} h^{\sum_{i=1}^{n} R_i} = g^{\sum_{i=1}^{n}} h^{\phi(N)\left\lceil \frac{\sum_{i=1}^{n-1} R_i}{\phi(N)} \right\rceil} = g^y$$

Hence it follows that: $\prod_{i=1}^{n} \tau_i = \prod_{i=1}^{n} \pi_i$.

Further the Pedersen commitment is perfectly hiding and computationally binding and hence it follows that the requirements is still fulfilled.

It remains to prove that Construction 5 also fulfils the extended verification requirement. This follows from the soundness of the range proofs or set membership proof used in the construction and by the argument above that the secret hidden in the commitment π_i must be the same as the secret obtain by combining the the shares $\{x_{ij}\}_{j\in\mathcal{M}}$ for all $i = 1, ..., n$.

$\square$

Construction 5 : Client and Server Verifiable additive homomorphic secret sharing

Goal: Construct and share the sum $\sum_{i=1}^{n} x_i$, where x_i is a secret value known by client c_i, where $i \in \mathcal{N}$ without any client needing to revealing their individual secret. All parties are verified to be honest in the construction.

- **ShareSecret** $(1^\lambda, i, x_i) \mapsto \{x_{ij}\}_{j\in\mathcal{M}}$
  Pick uniformly at random $\{a_i\}_{i\in\{1,..,t\}} \in_R \mathbb{F}$ to be the coefficients to a t-degree polynomial p_i on the form $p_i(X) = x_i + a_1X + ... + a_tX^t$. Define the shares as $x_{ij} = \lambda_{i,j} p_i(\theta_{ij})$ for $j \in \mathcal{M}$, the parameters θ_{ij} and Lagrange coefficients λ_{ij} is chosen such that equation 2.2 is satisfied. Output $\{x_{ij}\}_{j\in\mathcal{M}}$.
- **GenereteCommitment**$(1^\lambda, i, x_i) \mapsto \pi_i$
  Let $\mathbb{E} : x, y \to g^x h^y$ be a Pedersen commitment . Let $R_i \in \mathbb{F}$ be the output of a PRF such that $R_n \in \mathbb{F}$ satisfies $R_n = \phi(N)\lceil \frac{\sum_{i=1}^{n-1} R_i}{\phi(N)} \rceil - \sum_{i=1}^{n-1} R_i$. Compute and output $\pi_i = \mathbb{E}(x_i, R_i)$.
- **RangeProof** $(x_i, \pi_i) \mapsto RP_i$
  Construct a range proof, denoted RP_i, for the commitment π_i to the secret x_i, on the range $[0, B]$ (or a set Φ) using Construction 2, 3 or 4. All required parameters and setup is assumed to be pre-shared and known by all parties.
- **PartialEval** $(j, \{x_{ij}\}_{i\in\mathcal{N}}) \to y_j$
  Compute and output $y_j = \sum_{i=1}^{n} x_{ij}$.
- **PartialProof** $(j, \{x_{ij}\}_{i\in\mathcal{N}}) \to \sigma_j$
  Compute and output $\sigma_j = \prod_{i=1}^{n} g^{x_{ij}} = g^{\sum_{i=1}^{n} x_{ij}} = g^{y_j} = H(y_j)$.
- **FinalEval** $(\{y_j\}_{j\in\mathcal{M}}) \to y$
  Compute and output $y = \sum_{j=1}^{m} y_j$.
- **FinalProof** $(\{\sigma_j\}_{j\in\mathcal{M}}) \to \sigma$
  Compute and output $\sigma = \prod_{j=1}^{m} \sigma_j = \prod_{j=1}^{m} g^{y_j} = g^{\sum_{j=1}^{m} y_j} = g^y = H(y)$.
- **Verify** $(\{\pi_i\}_{i\in\mathcal{N}}, x, y, \{RP_i\}_{i\in\mathcal{N}}) \to \{0, 1\}$
  Compute and output $\sigma = \prod_{i=1}^{n} \pi_i \wedge \prod_{i=1}^{n} \pi_i = H(y) \wedge \{\mathbf{Verify}_{rp}(RP_i)\}_{i\in\mathcal{N}}$. Where $\mathbf{Verify}_{rp}$ is the verification algorithm associated with the algorithm used by the clients to construct the range proofs, $\{RP_i\}_{i\in\mathcal{N}}$.

3.3 Improving runtime

A desired property for the above presented server and clients verifiable AHSS would be to aggregate the range proofs into one, since then the verifier would have to perform one range proof verification instead of one for each client as in Construction 5. This would decrease the runtime for the verification significantly, especially in implementations where many clients participate. Aggregating the range proofs would require the proofs to be homomorphic, such that the verification remains valid also for an aggregated proof.

A small remark is that the naive approach to aggregate the commitments π_i, $i \in \mathcal{N}$ to $\pi = \prod_{i=1}^{n} \pi_i$. Then construct a range proof for the aggregated commitment $\pi = g^{\sum_{i=1}^{n} x_i}$ is in the range $[n{\cdot}a, n{\cdot}b]$, to prove $x_i \in [a,b]$ for all $i \in \mathcal{N}$ does not satisfy the verification property in Theorem 4. The value $y = \sum_{i=1}^{n} x_i$ is publicly known so to construct a zero knowledge range proof for y provides no new information and given that $y \in [n \cdot a, n \cdot b]$ does not imply $x_i \in [a,b]$ for all $i \in \mathcal{N}$.

Aggregating Set membership proof

In this section the possibility to aggregate the set membership and signature based range proofs is examined. The construction of these two are similar and it is sufficient to consider one of them, due to its simpler notation the set membership proof is considered. Consider two range proofs RP_1 and RP_2 generated by the algorithm **Prove** in construction 2, recall that such set membership proofs are on the form $RP_i = (V_i, a_i, D_i, z_{x_i}, z_{\tau_i}, z_{R_i},)$ for $i = 1, 2$ and that a proof verifies true if $D = C_i^{c_i} h^{z_{R_i}} g^{z_{x_i}} \wedge a_i = e(V_i, y)^c e(V_i, g)^{z_{x_i}} e(g,g)^{z_{\tau_i}}$. In addition to knowledge of the proof the verifier also has knowledge of the commitment C_i published by the clients, group elements h, g and the challenge c_i can be computed by the verifier. To aggregate the proof each element building the proof would need to be aggregated such that the verification of the aggregated proof can be carried out in the same way as before. First test the straight forward aggregation hence let $RP = (V, a, D, z_x, z_\tau, z_R)$ be the aggregated range proof and where $V, a, D, z_x, z_\tau, z_R$ be defined as,

$$\begin{aligned}
V =& V_1 V_2 = g^{\frac{\tau_1}{\chi+x_1}} g^{\frac{\tau_2}{\chi+x_2}} = g^{\frac{\tau_1}{\chi+x_1}+\frac{\tau_2}{\chi+x_2}} \\
a =& a_1 a_2 = \Big(e(V_1,g)^{-s_1})e(g,g)^{t_1}\Big)\Big(e(V_2,g)^{-s_2})e(g,g)^{t_2}\Big) \\
=& \Big(e(g,g)^{\frac{-s_1\tau_i}{\chi+x_1}} e(g,g)^{t_1}\Big)\Big(e(g,g)^{\frac{-s_2\tau_2}{\chi+x_2}} e(g,g)^{t_2}\Big) \\
D =& D_1 D_2 = (g^{s_1} h^{m_1})(g^{s_2} h^{m_2}) = g^{s_1+s_2} h^{m_1+m_2} \\
z_x =& z_{x_1} + z_{x_2} = (s_1 - c_1 x_1) + (s_2 - c_2 x_2) \\
z_R =& z_{R_1} + z_{R_2} = (m_1 - c_1 R_1) + (m_2 - c_2 R_2) \\
z_\tau =& z_{x_1} + z_{x_2} = (t_1 - c_1 \tau_1) + (t_2 - c_2 \tau_2)
\end{aligned} \tag{3.1}$$

Further also calculate the challenges c_1 and c_2 according to $c_i = Hash(a_i, D_i)$, $i = 1, 2$ and remember that the commitments C_i are homomorphic, which follows directly from the homomorphic properties of the Pedersen commitment. It is less obvious to see that the bilinear map can be aggregated, but this has been shown and the security proven in [4]. However the homomorphic properties of the Pedersen commitment and bilinear maps does not guarantee that the range proofs is homomorphic.

Next it will be investigated if the aggregated proof RP verifies true given all parties were honest, i.e see if the completeness property holds after aggregation. For the verification to succeed it must hold that, 1) $D \overset{?}{=} C^c h^{z_R} g^{z_x}$ and 2) $a \overset{?}{=} e(V,y)^c e(V,g)^{-z_x} e(g,g)^{z_\tau}$, hence check if it holds starting with the first.

$$\begin{aligned}
LHS &= D = D_1 * D_2 = g^{s_1+s_2} * h^{m_1+m_2} \\
RHS &= C^c h^{z_R} g^{z_x} = (C_1 * C_2)^{c_1c_2} h^{z_{R_1}+z_{R_2}} g^{z_{x_1}+z_{x_2}} \\
&= (g^{x_1}h^{R_1}g^{x_2}h^{R_2})^{c_1c_2} h^{m_1-R_1c_1+m_2-R_2c_2} g^{s_1-x_1c_1+s_2-x_2c_2} \\
&= g^{c_1c_1(x_1+x_2)+s_1+s_2-x_1c_1-x_2c_2} h^{c_1c_2(R_1+R_2)+m-R_1c_1-R_2c_2}
\end{aligned}$$
$$\implies \text{LHS} \neq \text{RHS}.$$

The equality does not hold due to the $c_1c_2(x_1+x_2) = c_1c_2x_1 + c_1c_2x_2 \neq x_1c_1 + x_2c_2$ and hence the terms does not cancel and the RHS is dependent of x_1, x_2, c_1, c_2 unlike the LHS. Further note that if $c_1 = c_2$ then this would be easy to fix. Clearly it cannot be guaranteed that the two challenges will be equal since they depend on randomness in the proof construction. This leads to that a more cleaver aggregation that circumvent this issue is needed. Again consider the two range proofs RP_1, RP_2 as defined above, but for now only the first equality $D \overset{?}{=} C^c h^{z_R} g^{z_x}$ in the verification is of interest. Remark is that if only half the verification is aggregated it still lead to important reduce of computations for the verifier. The goal is now to combine the two range proofs into one aggregated proof that fulfils first equality. Before aggregation calculate the challenges c_1, c_1 as $c_i = Hash(D_i, a_i)$, $i = 1, 2$. Then define the new aggregated range proof $RP' = (D, z_x, z_R)$, since only the first equality is concerned, for now, the proof does not contain the bilinear map a and the group element V, z_τ. Further the aggregated proof is defined as,

$$\begin{aligned}
&D = D_1^{c_2} \cdot D_2^{c_1} = (g^{s_1}h^{m_1})^{c_2} \cdot (g^{s_2}h^{m_2})^{c_1} = g^{s_1c_2+s_2c_1} h^{m_1c_2+m_2c_1} \\
&z_x = c_2 z_{x_1} + c_1 z_{x_2} = c_2(s_1 - x_1c_1) + c_1(s_2 - x_2c_2) = s_1c_2 + s_2c_1 - c_1c_2(x_1+x_2) \\
&z_R = c_2 z_{R_1} + c_1 z_{R_2} = c_2(m_1 - R_1c_1) + c_1(m_2 - R_2c_2) = m_1c_2 + m_2c_1 - c_1c_2(R_1+R_2)
\end{aligned} \tag{3.2}$$

Additionally also define the aggregated challenge and commitment as,

$$\begin{aligned}
c &= c_1c_2 \\
C &= C_1C_2 = (g^{x_1}h^{R_1})(g^{x_2}h^{R_2}) = g^{x_1+x_2}h^{R_1+R_2} = g^{x_1+x_2}.
\end{aligned}$$

It is assumed that the random values R_i is chosen such that $R_n = \phi(N)\lceil \frac{\sum_{i=1}^{n-1} R_i}{\phi(N)} \rceil - \sum_{i=1}^{n-1} R_i$, which holds for the randomness in a VHASS construction, hence for $i = 1, 2$ if follows that $R_2 = \phi(N)\lceil \frac{R_1}{\phi(N)} \rceil - R_1$, and thus $h^{c_1c_2(R_1+R_2)} = 1$. This property will not be required for the below calculations, however is does reduce notation and therefore the computations are done under this assumption. Using this aggregation to construct the aggregated proof, RP, when evaluate if $D \stackrel{?}{=} C^c h^{z_R} g^{z_x}$ it holds that,

$$\begin{aligned}
LHS &= D = D_1^{c_2} \cdot D_2^{c_1} = g^{s_1c_2+s_2c_1} h^{m_1c_2+m_2c_1} \\
RHS &= C^c h^{z_R} g^{z_x} = (C_1C_2)^{c_1c_2} h^{c_2z_{R_1}+c_1z_{R_2}} g^{c_2z_{x_1}+c_1z_{x_2}} \\
&= (g^{x_1+x_2})^{c_1c_2} h^{m_1c_2+m_2c_1} g^{s_1c_2+s_2c_1-c_1c_2(x_1+x_2)} \\
&= g^{(x_1+x_2)c_1c_2-c_1c_2(x_1+x_2)+s_1c_2+s_2c_1} h^{m_1c_2+m_2c_1} = g^{s_1c_2+s_2c_1} h^{m_1c_2+m_2c_1}
\end{aligned}$$

$\implies$ LHS = RHS.

This means that this aggregation to construct the proof RP , from the two range proofs RP_1, RP_2 as above satisfies the first equality of the verification. Next it will be tested if this can aggregation can be extended to an aggregate arbitrary number of range proofs. Consider $|\mathcal{S}|$ clients, $c_i\, i \in \mathcal{S}$, and $|\mathcal{S}|$ range proofs denoted RP_i, $i \in \mathcal{S}$. The aggregation in equation (3.2) written for $|\mathcal{S}|$ proofs is then,

$$\begin{aligned}
D &= \prod_{i\in\mathcal{S}} D_i^{\prod_{j\in\mathcal{S}, j\neq i} c_j} = \prod_{i\in\mathcal{S}} (g^{s_i}h^{m_i})^{\prod_{j\in\mathcal{S}, j\neq i} c_j} = g^{\sum_{i\in\mathcal{S}}\left(\prod_{j\in\mathcal{S}, j\neq i} c_j\right)s_i} h^{\sum_{i\in\mathcal{S}}\left(\prod_{j\in\mathcal{S}, j\neq i} c_j\right)m_i} \\
z_x &= \sum_{i\in\mathcal{S}}\Big(\prod_{\substack{j\in\mathcal{S}\\ j\neq i}} c_j\Big) z_{x_i} = \sum_{i\in\mathcal{S}}\Big(\prod_{\substack{j\in\mathcal{S}\\ j\neq i}} c_j\Big) s_i - \Big(\prod_{j\in\mathcal{S}} c_j\Big)\sum_{i\in\mathcal{S}} x_i \\
z_R &= \sum_{i\in\mathcal{S}}\Big(\prod_{\substack{j\in\mathcal{S}\\ j\neq i}} c_j\Big) z_{R_i} = \sum_{i\in\mathcal{S}}\Big(\prod_{\substack{j\in\mathcal{S}\\ j\neq i}} c_j\Big) m_i - \Big(\prod_{j\in\mathcal{S}} c_j\Big)\sum_{i\in\mathcal{S}} R_i
\end{aligned} \tag{3.3}$$

Let $c = \prod_{i\in\mathcal{S}} c_i$ be the product of all challenges and $C = \prod_{i\in\mathcal{S}} C_i$ the product of the commitments. Then define the aggregated range proof $RP = (D, z_x, z_r)$ aggregated according to equation 3.3 and examine if it holds $D \stackrel{?}{=} C^c h^{z_R} g^{z_x}$,

$$\begin{aligned}
LHS =& D = g^{\sum_{i\in\mathcal{S}}\left(\prod_{j\in\mathcal{S}, j\neq i} c_j\right)s_i} h^{\sum_{i\in\mathcal{S}}\left(\prod_{j\in\mathcal{S}, j\neq i} c_j\right)m_i} \\
RHS =& C^c h^{z_R} g^{z_x} = \Big(\prod_{i\in\mathcal{S}} C_i\Big)^{\prod_{i\in\mathcal{S}} c_i} h^{\sum_{i\in\mathcal{S}}\left(\prod_{j\in\mathcal{S}, j\neq i} c_j\right)m_i} \\
& g^{\sum_{i\in\mathcal{S}}\left(\prod_{j\in\mathcal{S}, j\neq i} c_j\right)s_i - \left(\prod_{j\in\mathcal{S}} c_j\right)\sum_{i\in\mathcal{S}} x_i} \\
=& \left(g^{\sum_{i\in\mathcal{S}} x_i}\right)^{\prod_{i\in\mathcal{S}} c_i} h^{\sum_{i\in\mathcal{S}}\left(\prod_{j\in\mathcal{S}, j\neq i} c_j\right)m_i} g^{\sum_{i\in\mathcal{S}}\left(\prod_{j\in\mathcal{S}, j\neq i} c_j\right)s_i - \left(\prod_{j\in\mathcal{S}} c_j\right)\sum_{i\in\mathcal{S}} x_i} \\
=& g^{\sum_{i\in\mathcal{S}}\left(\prod_{j\in\mathcal{S}, j\neq i} c_j\right)s_i} h^{\sum_{i\in\mathcal{S}}\left(\prod_{j\in\mathcal{S}, j\neq i} c_j\right)m_i}
\end{aligned}$$

$\implies$ LHS = RHS.

Above it has been seen that set memberships proofs can be aggregated according to equation (3.3) such that $D = C^c h^{z_R} g^{z_x}$, i.e this equality is checked once instead of once for each client. Given this, it is of interest to examine the possibilities to construct a aggregation such that multiple set membership proofs can be combined into one single proof, verifying all individual proofs by performing one verification. Hence it has to also hold that $a \overset{?}{=} e(V,y)^c e(V,g)^{-z_x} e(g,g)^{z_\tau}$, where a, V, z_x, z_τ are obtained by aggregating multiple set membership proofs. First it is tested if aggregation according to equation (3.1), under the assumption $c_1 = c_2$ is sufficient to satisfy this. If the equality would be satisfied then the same trick as above, namely aggregating such that the challenges appear only as a product, would yield a method for fully aggregating set membership proofs. After some calculation, see Appendix C for the details of the calculations, it is realised that the terms,

$$e(V_1,g)^{z_{x_2}} e(V_2,g)^{z_{x_1}} = e(g,g)^{\frac{\tau_1}{\chi+x_1}(-s_2+x_2c)+\frac{\tau_2}{\chi+x_2}(-s_1+x_1c)},$$

on the right side of the equality are not cancelled. This concludes that even if $c_1 = c_2$ the verification will not hold after aggregating according to equation (3.1). This implies that in order to aggregate the whole set membership proof additional tricks are required. Several attempts to construct an aggregation that will cancel these terms, whichout introducing new terms, has been made without any success. Therefore whether it can be done or not is left unanswered. What can be seen as an indicator to that it is not be possible is that in the algorithm **Verify** in Construction 3, the bilinear mappings are checked separately for each $j \in \mathbb{Z}_l$.

The above discussion about possibilities to aggregate multiple set memberships proofs to reduce the computational complexity in applications where many proofs are verified simultaneously, ended in a method for partly aggregating the proofs. This modified version of the set membership proof, originally presented in [**?**], is presented in Construction 6. The algorithm **Prove** in the aggregated set membership proof is run by all proving parties wanting to prove that their secret is in the set. Then the aggregating party takes all the set membership proofs as input and outputs an aggregated proof. The verifier takes the aggregated proof as input, and hence the equality check $D \overset{?}{=} (\prod_{i=1}^{n} C_i)^c h^{z_R} g^{z_x}$ is done once instead of once for each proving party in the protocol.

In Construction 6 the challenges are given as input to the verifier, unlike the original set membership proof where the verifier calculates the challenge $c_i = Hash(a_i, D_i)$. The reason for this is that the values D_i is not known to the verifier in the aggregated set membership proof. This raises the question about which party that should be responsible for calculating the challenges. It is not desired that the party performing the aggregation also computes the challenger since this opens up for cheating, for the same reason should the proving party not compute the challenge. Thus, the challenges are computed by a different party, independent of both the proving-parties and the aggregating party, and finally multiplied together by the verifier. Another possibility is to include the values $\{D_i\}_{i \in \mathcal{S}}$ as input to the verification, leading to that the verifying party can compute the challenges, however this leads more computation needed in the verification.

Construction 6 : Aggregation of non interactive set membership proof

Goal: Given the Pedersen commitments $C_i = g^{x_i}h^{R_i}$ and a set Φ, prove that the secrets x_i , for $i \in \mathcal{S}$, in the commitments belongs to the set Φ without revealing anything else about the secrets x_i.

- **SetUp** $(g, h, \Phi) \rightarrow (y, \{A_i\}_{i\in\Phi})$
  Pick uniformly at random $\chi \in_R \mathbb{F}$. Define $y = g^{\chi}$ and $A_i = g^{\frac{1}{\chi+i}}$ $\forall i \in \Phi$, output y and $\{A_i\}_{i\in\Phi}$.
- **Prove** $(g, h, C_i, \Phi) \rightarrow$ $proof_{SM,i} = (V_i, a_i, D_i, z_{x_i}, z_{\tau_i}, z_{R_i})$
  Pick uniformly at random $\tau_i \in_R \mathbb{F}$, choose from the set $\{A_i\}$ the element A_{x_i} and calculate $V = A_{x_i}^{\tau_i}$. Pick uniformly random three values $s_i, t_i, m_i \in_R \mathbb{F}$. Put $a_i = e(V_i, g)^{-s_i}e(g, g)^{t_i}$ ($e(\cdot, \cdot)$ is a bilinear mapping as described above), $D = g^{s_i}h^{m_i}$, and $c = \text{Hash}(a_i, D_i)$. Finally compute $z_{x_i} = s_i - x_ic_i$, $z_{R_i} = m_i - R_ic_i$ and $z_{\tau_i} = t_i - \tau_ic_i$ then construct and publish $proof_{SM,i} = (V_i, a_i, D_i, z_{x_i}, z_{\tau_i}, z_{R_i})$.
- **Aggregate** $(g, h, \{proof_{SM,i}\}_{i\in\mathcal{S}} \rightarrow proof_{SM,a}$
  Given a subset of range proofs $\{proof_{SM,i}\}_{i\in\mathcal{S}}$ where $\mathcal{S} \subseteq \{1, ..., n\}$. Aggregate the values $\{D_i\}_{i\in\mathcal{S}}\{z_{x_i}\}_{i\in\mathcal{S}}\{z_{r_i}\}_{i\in\mathcal{S}} \mapsto D_a, z_{x_a}, z_{R_a}$ according to equation (3.3). Then construct and publish the aggregated range proof $proof_{SM,a} = (\{V_i\}_{i\in\mathcal{S}}, \{a_i\}_{i\in\mathcal{S}}, D_a, \{z_{x_i}\}_{i\in\mathcal{S}}, z_{x_a}, \{z_{\tau_i}\}_{i\in\mathcal{S}}, z_{R_a})$.
- **CalculateChallenges** $(\{D_i\}_{\in\mathcal{S}}, \{a_i\}_{i\in\mathcal{S}}) \rightarrow \{c_i\}_{i\in\mathcal{S}}$
  Compute and output $c_i = Hash(a_i, D_i)$ for all $i \in \mathcal{S}$.
- **VerifyAggregatedProof** $(g, h, \{C_i\}_{i\in\mathcal{S}}, \{c_i\}_{i\in\mathcal{S}}, proof_{SM,a}) \rightarrow \{0, 1\}$
  Compute the product of the challenges $c = \prod_{i\in\mathcal{S}} c_i$. Check if $D_a \overset{?}{=} \left(\prod_{i\in\mathcal{S}} C_i\right)^c h^{z_{R_a}}g^{z_{x_a}} \wedge a_i \overset{?}{=} e(V_i, y)_i^c e(V_i, g)^{-z_{x_i}}e(g, g)^{z_{\tau_i}}$ for all $i \in \mathcal{S}$. If the equalities holds the provers has convinced the verifier that $x_i \in \Phi$ for all x_i such that $i \in \mathcal{S}$ return 1 otherwise return 0.

Completeness, Soundness and Zero-knowledge

This section will investigate if the proposed aggregated set membership proof presented in Construction 6 fulfils the completeness, soundness and zero-knowledge requirements stated in Definition 4 under the assumption that the aggregation has been done correct by a trusted party.

Completeness follows from the argument above that it holds that $D \overset{?}{=} C^c h^{z_R} g^{z_x}$, given all parties where honest. It is also clear that multiplying and adding elements which perfectly hides the secret x_i, with other elements independent of the secret will not reveal any information about the secret, hence the zero-knowledge property holds.

This leads to that it remains to see if the soundness property holds for Construction 6. The equality check $D \overset{?}{=} C^c h^{z_R} g^{z_x}$ in the original set membership construction servers the purpose of ensuring the verifier that the secret hidden in the commitment C is the same as the secret used to construct the value z_x. Hence this property need to be checked that it remains after the aggregation, i.e that the verifier can be sure that all secrets hidden in the commitments, are equal to the secrets used to compute the values z_{x_i}, in turn used in the aggregation to construct z_x. Consider two set membership proofs, RP_1 and RP_2, computed by the algorithm **Prove** in construction 6 are aggregated into $RP = (\{V_1, V_2\}, \{a_1, a_2\}, D, z_x, \{z_{x_1}, z_{x_2}\}, \{z_{\tau_1}, z_{\tau_2}\}, z_R))$, according to algorithm **Aggregate** in construction 6. Given this set up, it will be investigated if it can hold that: $D = (C_1C_2)^c h^{z_R} g^{z_x}$, where $C_i = g^{x_i} h^{R_i}$ and $z_{x_i} = s_i - c_i\tilde{x}_i$ such that $x_i \neq \tilde{x}_i$ for i equal to either 1, 2 or both. Consider the below equations,

$$
\begin{aligned}
LHS =& g^{s_1c_2+s_2c_1} h^{m_1c_2+m_2c_1} \\
RHS =& g^{c_1c_2(x_1+x_2)} h^{c_1c_2(R_1+R_2)} g^{c_2(s_1-c_1\tilde{x}_1)+c_1(s_2-c_2\tilde{x}_2)} h^{c_2(m_1-c_1\tilde{R}_1)+c_1(m_2-c_2\tilde{R}_2)} \\
=& g^{s_1c_2+s_2c_1} h^{m_1c_2+m_2c_1} g^{c_1c_2(x_1+x_2)-c_1c_1(\tilde{x}_1+\tilde{x}_2)} h^{c_1c_2(R_1+R_2)-c_1c_1(\tilde{R}_1+\tilde{R}_2)}.
\end{aligned}
$$

Under the assumption that the aggregation was done correctly, one of the following cases has to hold if $LHS = RHS$,

1. $x_1 = \tilde{x}_1$ and $x_2 = \tilde{x}_2$
2. $x_1 = \tilde{x}_2$ and $x_2 = \tilde{x}_1$
3. $(x_1 + x_2) = (\tilde{x}_1 + \tilde{x}_2) \bmod \Phi(N)$

The first case corresponds to honest clients while the second and third case corresponds to cheating clients. In the second case the clients are cheating, since they do not use the same secret in the commitments as in z_{x_i}. If this is considered in the context of using the aggregated set membership proof in combination to a VAHSS construction the sum y calculated in the VAHSS construction evaluates to the same value in both case 1 and 2, and all used terms in the summation is proved to be in the set Φ. So despite cheating clients the result from the protocol is unaffected. This is since a cheating party in case 2 only achieves to having his secret being committed to by another clients, as he commits to this clients secret. For the VAHSS construction it is not of relevance who committed what as long as the sum is the same and all terms in the sum is verified to be in the set. The third case shows that if two clients collaborate it is possible for them to use to the secrets x_1 and x_2 in the Pedersen commitment while using two other secrets $\tilde{x}_1$ and $\tilde{x}_2$ such that the they satisfies the third case, then $D \overset{?}{=} (\prod_{i=1}^{2} C_i)^c h^{z_R} g^{z_x}$ holds true. Thereby using $\tilde{x}_1$ and $\tilde{x}_2$ for the remaining of the proof and if they belong to the set Φ, the clients has successfully cheated, since they have ensured the verifier that the secrets x_1 and x_2 are in the set, by falsely proven that the secrets in x_1, x_2 are equal to $\tilde{x}_i, \tilde{x}_2$. Thus it has to be assumed that clients cannot communicate in order for the soundness property to hold for the aggregated set membership. Under this assumption both case two and case three has a negligible probability of succeeding, since the probability of two clients committing to secrets that by chance satisfies either case two or three is sufficiently small.

Assumptions about Aggregation

In the previous section it was assumed that the aggregation was done according to equation (3.3) by a trusted party. Under this assumption is was argued that if the proving parties cannot communicate then the correctness, soundness and zero-knowledge property holds for the aggregated set membership proof in Construction 6. In this section it will be investigated if the trusted party assumption is necessary or if some weaker assumption is sufficient alternatively if the aggregation can be checked. It will be assumed that proving parties cannot communicate between themselves and that proving parties cannot communicate with the aggregating party.

Note that since all input used to aggregate the proves are public, the aggregation can be done by anybody. Therefore an aggregation performed can always be checked if it is correct by performing the aggregation on one's own and then examine if the two aggregation are the same. To aggregate the set membership proofs is an expensive operations and hence arguments on an aggregations correctness without having to redo the aggregation would be useful.

Before looking into how the aggregation party might cheat and depart from the aggregation in equation (3.3), it will be cleared out what values the aggregation party can affect and what requirements that has hold. The aggregation party need to provide values for the proof parameters D, z_x, z_R such that $D = C^c h^{z_R} g^{z_x}$. The party performing the aggregation can not modify the values of the product of commitments, denoted C, nor the value of the product of the provers challenges, denoted c. Further it will be assumed that the if an aggregating party can choose the values D, z_x, z_R freely (not according to equation (3.3)) and the equality above holds, the aggregating party has successfully cheated.

An important feature in the original set membership construction is that the verifier uses the same value z_x is used for the entire verification. In the aggregated set membership, an aggregated value z_x is used to test if $D \overset{?}{=} C^c h^{z_R} g^{z_x}$ while non-aggregated values z_{x_i} are used to verify $a_i = e(V_i, y)^c e(V_i, g)^{-z_x} e(g, g)^{z_{\tau_i}}$ for $i \in \mathcal{S}$. Thereby the same value is not used in the entire verification. This means that in the aggregated set membership the aggregated value z_x has less claims that needs to be fulfilled in order for the verification to succeed.

A second important distinction is that in the original set membership construction the value of the challenge used is not possible to know when construction D, a since it is a function of these values. In the aggregated set membership the challenges depends on the values publicly known for the aggregating party. This leads to that this knowledge can be taken advantage of to cheat. This results in that besides having a party independent of the aggregation, possibly the verifier, computing the product of the challenges, it must also be verified that $D \neq C^c$, due to he argument below.

If it is not checked that $D \neq C^c$ the following would be possible to choose the values D, z_x, z_R according to,

$$\begin{aligned} D =& \prod_{i=1}^{n} C_i^{\prod_{i=1}^{n} c_i} \\ z_x =& \phi(p) \\ z_R =& \phi(p), \end{aligned}$$

where p is the prime underlying the field $\mathbb{F}$. For the above choice of D, z_x, z_R the equation, $D \overset{?}{=} \Big(\prod_{i=1}^{n} C_i\Big)^{\prod_{i=1}^{n} c_i} h^{z_R} g^{z_x}$ holds trivially true, independent of whether the commitment C_i and the values z_{x_i} hide the same secret for all $i = 1, ..., n$. Henceforth it will be assume that it is checked that $D \neq \Big(\prod_{i=1}^{n} C_i\Big)^{\prod_{i=1}^{n} c_i}$.

This leads to that the aggregation party can only cheat if the variables D, z_R, z_x be chosen such that $D \neq \Big(\prod_{i=1}^{n} C_i\Big)^{\prod_{i=1}^{n} c_i}$ and $LHS = RHS$ in the below equation system:

$$\begin{aligned} LHS =& D \\ RHS =& \Big(\prod_{i=1}^{n} C_i\Big)^{\prod_{i=1}^{n} c_i} h^{z_R} g^{z_x} = \Big(g^{\sum_{i=1}^{n} x_i} h^{\sum_{i=1}^{n} R_i}\Big)^{\prod_{i=1}^{n} c_i} h^{z_R} g^{z_x} \\ =& \Big(g^{(\prod_{i=1}^{n} c_i)\sum_{i=1}^{n} x_i)} h^{(\prod_{i=1}^{n} c_i)\sum_{i=1}^{n} R_i)}\Big) h^{z_R} g^{z_x}. \end{aligned}$$

What is left to be investigated is if the aggregating party can cheat by exploiting the fact that the value z_x in the above equation system, is not used in the verification of $a_i \overset{?}{=} e(V_i, y)^c e(V_i, g)^{-z_x} e(g, g)^{z_{\tau_i}}$. In practise this would mean that the value z_x does not need to contain the values x_i in its construction.

In order to have equality between the left hand side and the right hand side, the terms derived from the commitments must be cancelled. The aggregating party does not know the secrets x_i or the random values R_i for any $i = 1, .., n$. Consequently to cancel the terms it is argued that aggregation must be done according to equation (3.3). The argument is that in order to cancel the terms depending on the unknown sum of secrets, x_i, and random values, R_i, the values z_{x_i} and z_{R_i} must be used. These term can not be combined in a way that all cross terms are cancelled besides as in equation 3.3. Remark here that this is assumed and not proved, in other words the security of the aggregated set membership presented in Construction 6 is based on this assumption or alternatively as in the previous section the assumption that the aggregation is done by a trusted party.

When considering the aggregated set membership in combination to a VHASS construction the sum of secrets can be computed by any party after all servers has performed the algorithm **partialEval** and it is known that $h^{\sum_{i=1}^{n} R_i} = 1 \bmod \phi(N)$. This leads to that although the individual secrets are unknown, it is possible to cheat by using the fact that $y, h^{\sum R_i}$ are known. This is see below by choosing $D = C^{k+c}, z_R = k\phi(N)$ and $z_x = ky$, where $y = \sum_{i=1}^{n}$ and $k_R \in \mathbb{F}$, it follows that,

$$\begin{aligned}
LHS =& D = C^{k+c} = \left(g^{k\sum_{i=1}^{n} x_i} h^{k\sum_{i=1}^{n} R_i)}\right)\left(g^{(\prod_{i=1}^{n} c_i)\sum_{i=1}^{n} x_i)} h^{(\prod_{i=1}^{n} c_i)\sum_{i=1}^{n} R_i)}\right) \\
LHS =& C^c h^{z_R} g^{z_x} = C^c h^{k\phi(N)} g^{ky} = \left(g^{(\prod_{i=1}^{n} c_i)\sum_{i=1}^{n} x_i)} h^{(\prod_{i=1}^{n} c_i)\sum_{i=1}^{n} R_i)}\right) h^{k\phi(N)} g^{ky} \\
=& \left(g^{(\prod_{i=1}^{n} c_i)\sum_{i=1}^{n} x_i)} h^{(\prod_{i=1}^{n} c_i)\sum_{i=1}^{n} R_i)}\right) g^{ky} h^{k\phi(N)\lceil \frac{\sum_{i=1}^{n-1} R_i}{\phi(N)} \rceil)} \\
& \implies LHS = RHS.
\end{aligned}$$

If the sum of $\sum_{i=1}^{n} x_i$ and $\sum_{i=1}^{n} R_i$ where unknown then the aggregation party would have to choose the values z_x and z_R such that,

$$h^{z_R} g^{z_x} \overset{!}{=} g^{k\sum_{i=1}^{n} x_i} h^{k\sum_{i=1}^{n} R_i} = \left(g^{k\sum_{i=1}^{n} R_i} h^{k\sum_{i=1}^{n} R_i}\right)^k,$$

which is the same as construction two equal Pedersen commitments, which is assumed to be impossible due to that the $log_g h$ is unknown.

It has been seen that, when one party aggregates the set membership proofs for all clients in a server and client verifiable AHSS construction it is possible for this part to cheat such that aggregated proof verifies true without the statement in equation (2.4) is true.

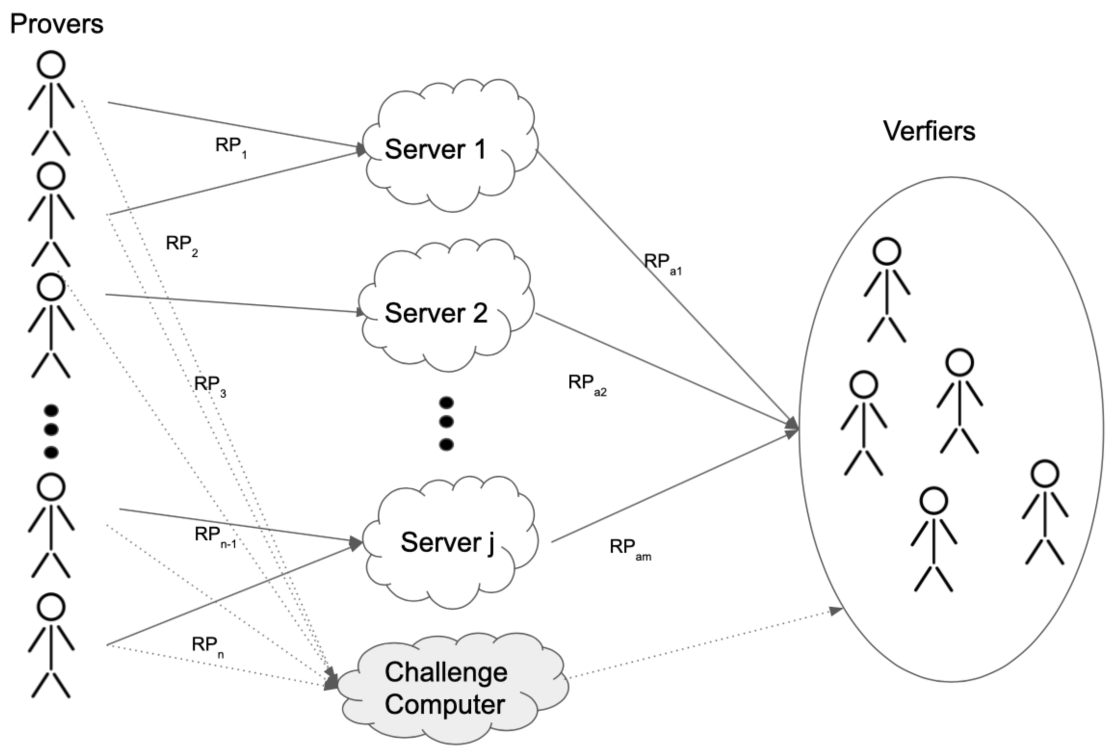
The argument above also provides a method to circumvent this issue. If instead of having one party that aggregates all set membership proofs multiple parties aggregates subsets of the proofs, it would not be possible to cheat. For the VAHSS construction this can be implemented by the server aggregate different subset of the clients range proofs. For example in the case of 100 clients and 5 servers, each server aggregates 20 proofs according to Equation (3.3). The value of the sum of any true subset of the secrets is unknown and likewise the sum of the randomness, hence any party aggregating a subset of all clients range proofs can not cheat as above, since it requires knowledge of the exponentials which are unknown. Given the aggregated range proofs generated by the servers, 5 aggregated range proofs has to be verified, this leads to 5 checks that the first equality holds and 100 checks of the second. Thus the first equality only has to be checked 5 times instead of 100 which would be the case if the range proofs were not aggregated. The workflow is illustrated in in Figure 3.1.

To conclude, if the aggregation is not assumed to be performed by a trusted party then the characteristics of the VAHSS construction all set membership proof cannot be aggregated together, instead two or more subsets of the range proofs must be aggregated separately and then all such aggregated subsets of proofs are verified by the verifier.

Aggregating Bulletproofs

The original paper about Bulletproofs [5] presents a method to aggregate Bulletproofs such that n parties each having a Pedersen commitment C_i, $i = 1, ..., n$ can generate a single Bulletproof verifying that each commitment hides a secret in an

Figure 3.1: TODO



allowed range. The presented approach only works if all parties uses the same challenge c in the proof construction, this is achieved by introducing a dealer. During the constructions of the proofs when computing the challenges each client sends their proof of to this point to the dealer who aggregates the proofs and computes the challenges based on the aggregated proofs. For example, assume n clients and denote their respective proofs with a subscript i, then to compute the challenges y_i in construction 4 instead of each client computing $y_i = Hash(A_i, S_i)$, each client sends A_i, S_i to the dealer who adds then homomorphically $A = \prod_{i=1}^{n} A_i, S = \prod_{i=1}^{n} S_i$ and the send back the challenge $y = Hash(A, S)$ to be use by all clients. This procedure is repeated for each challenge.

It is noted that although the Fiat-Shamir heuristic is used to generate the challenges the construction is interactive since communication between the dealer and the clients is required during the construction of the aggregated range proof. If this procedure was ignored and each client instead computed their own challenges via Fiat-Shamir heuristic and the proof where aggregated after they were fully constructed, then the challenges would differ between parties and the verification fail.

Concluding, it has been seen that the Bulletproof can be aggregated with the cost on an interactive construction, however this is not a desirable property for the the server and client verifiable AHSS. Investigation about whether this construction can be modified to be completely non-interactive has not been done and remains an open question.

3.4 Implementation

To practically investigate the combining of the VAHSS construction 1 with a range proof, construction 5 is implemented. From the analysis of the range proofs given above it is clear that Bulletproofs is faster than signature-based range proof, however considering the aggregation possibility for signature based range proof these might still be competitive for combination with VAHSS. All three constructions will hence be used to implement the client and server verifiable additive homomorphic secret sharing construction 5.

Implementations of Bulletproofs, set membership proofs and signature based range proofs written in Golang (Go) are all available on Github [10], implementations of the VAHSS construction 1, written in both python and C++, is available at Github [14] [13]. Because the implementation of the range proofs, set membership proofs and the VAHSS construction is not written in the same programming languages one of the two following modifications needs to be done. The first alternative is to write a wrapper for either the Go code so that it can be interpreted by a C++ (or Python) compiler, or wrap the CC++ (or Python) code such that it can be interpreted by a Golang compiler. The second alternative is to translate either the Go implementations to C++ (or Python) or the other way around.

The first alternative appears to be a simpler approach hence this is first tested. In 2016 *cgo* was released which enables calling C functions from Go code. The Go command *cgo* enables Go packages to call C code. TODO...

This lead to instead test the second alternative, that is translating the Go implementations to C++ (or Python) or the other way around. Since the VAHSS construction is more straight forward and much shorter than the two range proofs and the set membership proof all together this direction was chosen and Construction 1 was translated to Go.

Besides translating the VAHSS implementations to Go a small adjustments of the already existing Go implementations of the range proofs and set membership proof had to be done to merge with the VAHSS construction. This adjustment are merely to merge the codes and does not change the semantics of the range proofs. What has been modified is that the randomness used in the Pedersen commitments in the range proof must be chosen such that $R_n = \phi(N)\lceil \frac{\sum_{i=1}^{n-1} R_i}{\phi(N)} \rceil - \sum_{i=1}^{n-1} R_i$, hence is is regarded as input to the proof constructions.The full code for combination of range proofs (or set membership proofs) and VAHSS is available at Git **??**.

Just as in construction 5 the implementation of the code aims to be general, such that all three concerned range proofs can be used to verify clients honesty and the merge of the range proof to the VAHSS construction is the same for all range proofs. However note that although the construction does not specify which range proof that is used the implementation does due o the choice of underlying group in the set up, the signature-based range proofs and set membership proof uses pairing friendly elliptic curve groups in the implementation which is not the case for Bulletproofs. This leads to minor modifications of the implementation to adapt to range proofs and set membership proofs.

Specify what parameters that has been used for the implementation. The number of servers is set to 5 and the number of clients to 100. The range is set to

$[18, 200]$ for both range proof implementations. This leads to that the parameter n in the Bulletproof construction determining the complexity must be atleast 8 since $2^8 > 200$, to keep runtime low it is put equal to eight and for the signature-based range proof, having fixed $u = 57$ it is found sufficient to put $l = 2$. To make the set membership implementation comparable the size of the set is equal to the length of the range , i.e $|\Phi| = 200 - 18 = 182$.

For the prototype analysis of the server verifiable AHSS, performed in [15], the finite field $\mathbb{F}$ used for the secret shares generation is based on a 64-bit prime number, i.e $\mathbb{F} = \mathbb{Z}_p$, where p is a 64-bit prime number, but in the server and client verifiable AHSS the finite field is formed by a 256-bit prime number. The range proofs are based in libsecp256k1 library available in Go-Ethereum and uses elliptic curves and 128-bit security, to provide this security level the underlying field has to be of size $\sim$ 256-bits since the fastest known algorithm to solve elliptic curve discrete logarithm problem (ECDLP) requires $\mathcal{O}(\sqrt{n})$ steps. To use a common underlying field for the both constructions, the size of the field for the VAHSS is 256-bit instead of 64-bit. The hardware is the same as used above for benchmarking in section 3.1.2. For completeness it is repeated here; the computer used has a 1.6 GHz Dual-Core Intel Core i5 $-$ 5250U CPU, 8GB 1600 MHz DDR3 RAM and running macOS 10.15.

A final remark about the implementation is that its purpose is to test the concept on the above proposed construction and provide runtime evaluations, the code has not been tested enough to be used as secure implementation.

4

Results

First the purpose of the paper is repeated to motivate the results obtained below. The main purpose of this paper is to investigate if and how the VAHSS construction 1 can be extended with a range proof to ensure honest clients. Beside this main purpose the aim is also to provide an implementation of such a combined construction and to compare different range proofs and their compatibility to VAHSS. The results for there three questions will be given below.

4.1 Combining

The main result of this paper is that it is possible to combine a VAHSS construction as described in section 2.2 with a range proof to reduce the potential impact of malicious clients. Using range proofs (or set membership proofs) that assumes a Pedersen commitment the combining with the VAHSS construction becomes almost parallel in the scene that the VAHSS and range proof (or set membership proof) are run almost independent of each other. It was also found that the combination could be done without specifying the details about the range proof, hence any range proof that assumes a Pedersen commitment can be used, which leads to a highly flexible combination.

The main factor determining the suitability of different range proofs is their runtime and their possibility to be aggregated, since all can be combined using the same approach to the VAHSS construction. The two range proofs studied are Bulletproofs and signature-based range proofs and the runtime comparison between them presented in Table 3.1 showed that Bulletproofs were significantly faster both in the proof construction and verification algorithm. The insight that a straight forward combination as in Construction 5 leads to that the verifier has to verify all range proofs separately lead to the attempt to aggregate the range proof, i.e combine the individual range proofs such that the verifier could instead verify one combined range proof. This can be compared to the the idea of the two algorithms **PartialProof** and **FinalProof** in the VAHSS construction, where the partial proofs are combined before the verification. It was found that the set membership proof and signature-based range proof could be partly aggregated such that the verifier only had to check one of the two equalities, in the algorithm **Verify** in construction 2 and 3, for all clients and the other only once. The aggregation has been argued not ro weaken the completeness, soundness and zero knowledge requirements of the set membership and signature-based range proof.

Table 4.1: Timing in seconds for server and client verifiable-AHSS. Verification of clients is done using three different constructions namely by implementing Bulletproofs, signature based range proofs and set membership proofs

	Executer	**Time**		
		Bulletproofs	Signature-based	Set membership
GenerateShares	client	95 [μs]	96[μs]	98 [μs]
GenerateRangeProof	clients	53 [ms]	241 [ms]	66 [ms]
PartialEval	server	78 [μs]	72[μs]	71 [μs]
PartialProof	server	273[μs]	5249 [μs]	5255 [μs]
FinalEval	x	689 [ns]	655 [ns]	699 [ns]
FinalProof	x	50 [μs]	114 [μs]	115 [μs]
VerifyRP	x	2979[ms]		9288 [ms]
VerifyServers	x	1672 [μs]	7990 [ms]	7947 [μs]

4.2 Runtime

This is also confirmed to be the case when considering their difference in runtime when combined with a VAHSS, as seen in Table 4.1. However above it has been discussed that set membership proofs and signature-based range proof can be partly aggregated, which would reduce the verification time.

hence more suitable for using in server and client verifiable AHSS presented in Construction 5.

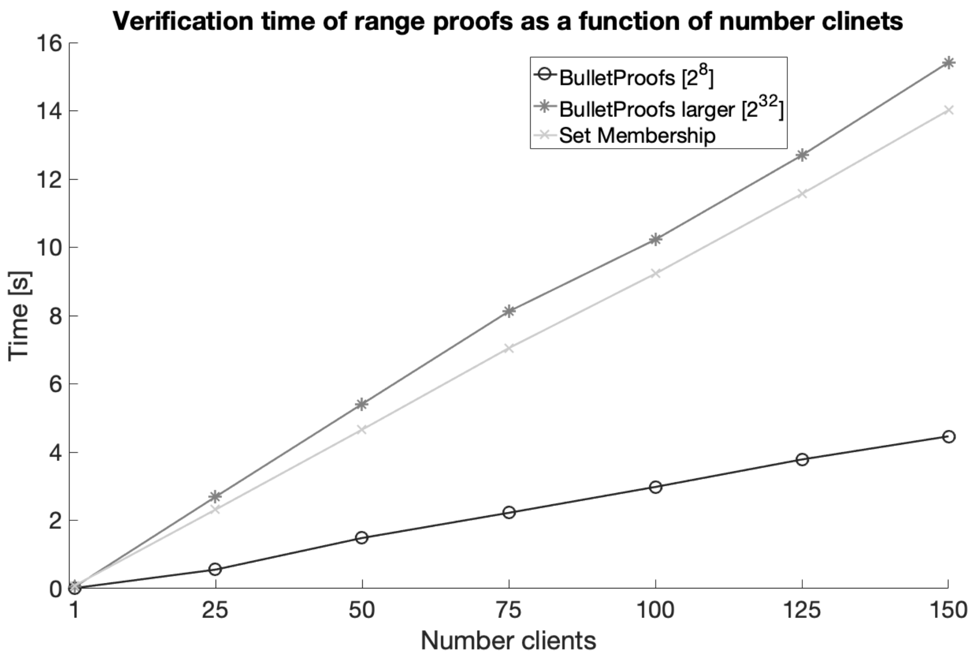
To obtain the values in Table 4.1 the maximum upper bound for Bulletproof is $2^{n=8} = 256$, which is sufficiently large maximum upper bound for the range $[18, 200]$. However is would be interesting to investigate how the runtime of the verification in Construction 5 is affected if this maximum bound of the range was increased. This is seen by the top curve in Figure 4.1. An interesting remark is that the runtime for Bulletproofs, when $n = 32$, are longer then for set membership proofs independent of the number of clients considered and as mentioned earlier the runtime for verification using the set membership protocol in independent of both the size of the set and the values of the elements in the set.

The aggregated version of the set membership and signature-based range proof has not been implemented thus result about how this aggregation affects the runtime can not be determined. In order to still get some idea of the runtime reduction, the algorithm **Verify** in Construction 2 was reduced to:

- **Verify** $(g, h, C, \mathit{proof}) \to \{0, 1\}$ Check if $a \overset{?}{=} e(V, y)^c e(V, g)^{-z_x} e(g, g)^{z_\tau}$. If the equality holds the prover has convinced the verifier that $x \in \Phi$ return 1 otherwise return 0.

Then the runtime of this algorithm ran 100 times, once for each clinet, was compared to the original version ran 100 times. This gave the result that the modified version was approximately 20% faster. This given some indication of the speed up obtained by aggregating since the equality check $D \overset{?}{=} C^c h^{z_\tau} g^{z_x}$ will only have to be ran once thus this runtime might be ignored in the context.

Figure 4.1: TODO



5 Conclusion

5.1 Discussion

Limit, only considerd range proof using pedersen commitment scheme.

FFS for intervals: Need comunication between servers. We do not want

5.2 Conclusion

Bibliography

[1] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, CCS ’93, page 62–73, New York, NY, USA, 1993. Association for Computing Machinery.

[2] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS ’12, page 326–349, New York, NY, USA, 2012. Association for Computing Machinery.

[3] D. Boneh and X. Boyen. Short signatures without random oracles. In C. Cachin and J. L. Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, pages 56–73, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[4] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In E. Biham, editor, *Advances in Cryptology — EUROCRYPT 2003*, pages 416–432, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[5] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 315–334, 2018.

[6] J. Camenisch, R. Chaabouni, and a. shelat. Efficient protocols for set membership and range proofs. In J. Pieprzyk, editor, *Advances in Cryptology - ASIACRYPT 2008*, pages 234–252, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[7] R. Chaabouni, H. Lipmaa, and A. Shelat. Additive combinatorics and discrete logarithm based range protocols. In R. Steinfeld and P. Hawkes, editors, *Information Security and Privacy*, pages 336–351, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[8] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[9] E. Morais, T. Koens, C. van Wijk, and A. Koren. A survey on zero knowledge range proofs and applications. *SN Applied Sciences*, 1(946), 2019.

[10] E. Morais, T. Koens, C. van Wijk, A. Koren, P. Rudgers, and C. Ramaekers. Zero knowledge range proof implementation. `https://github.com/ing-bank/zkrp`, 2020.

[11] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In J. Feigenbaum, editor, *Advances in Cryptology — CRYPTO ’91*, pages 129–140, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.

[12] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, Nov. 1979.

[13] G. Tsaloli and G. Banegas. Additative vhsss implemented in c++. `https://github.com/tsaloligeorgia/AddVHSS`, 2020.

[14] G. Tsaloli and G. Banegas. Additative vhsss implemented in python. `https://github.com/gbanegas/VHSSS_temp`, 2020.

[15] G. Tsaloli, G. Banegas, and A. Mitrokotsa. Practical and provably secure distributed aggregation: Verifiable additive homomorphic secret sharing. *Cryptography*, 4(3), 2020.

[16] G. Tsaloli and A. Mitrokotsa. Sum it up: Verifiable additive homomorphic secret sharing. In J. H. Seo, editor, *Information Security and Cryptology – ICISC 2019*, pages 115–132, Cham, 2020. Springer International Publishing.

[17] H. Yao, C. Wang, B. Hai, and S. Zhu. Homomorphic hash and blockchain based authentication key exchange protocol for strangers. In *2018 Sixth International Conference on Advanced Cloud and Big Data (CBD)*, pages 243–248, 2018.

A
Aggregated VHASS

Construction 7 : Client and Server Verifiable additive homomorphic secret sharing

Goal: Construct and share the sum $\sum_{i=1}^{n} x_i$, where x_i is a secret value known by client c_i, where $i \in \mathcal{N}$ without any client needing to revealing their individual secret. All parties are verified to be honest in the construction.

- **ShareSecret** $(1^\lambda, i, x_i) \mapsto \{x_{ij}\}_{j\in\mathcal{M}}$
  Pick uniformly at random $\{a_i\}_{i\in\{1,..,t\}} \in_R \mathbb{F}$ to be the coefficients to a t-degree polynomial p_i on the form $p_i(X) = x_i + a_1X + ... + a_tX^t$. Define the shares as $x_{ij} = \lambda_{i,j}p_i(\theta_{ij})$ for $j \in \mathcal{M}$, the parameters θ_{ij} and Lagrange coefficients λ_{ij} is chosen such that equation 2.2 is satisfied. Output $\{x_{ij}\}_{j\in\mathcal{M}}$.
- **GenereteCommitment**$(1^\lambda, i, x_i) \mapsto \pi_i$
  Let $\mathbb{E} : x, y \to g^x h^y$ be a Pedersen commitment . Let $R_i \in \mathbb{F}$ be the output of a PRF such that $R_n \in \mathbb{F}$ satisfies $R_n = \phi(N)\lceil\frac{\sum_{i=1}^{n-1} R_i}{\phi(N)}\rceil - \sum_{i=1}^{n-1} R_i$. Compute and output $\pi_i = \mathbb{E}(x_i, R_i)$.
- **RangeProof** $(x_i, \pi_i) \mapsto RP_i$
  Construct a range proof, denoted RP_i, for the commitment π_i to the secret x_i, on the set Φ using the agorithm **Prove** in Construction 6. All required parameters and setup is assumed to be pre-shared and known by all parties.
- **PartialEval** $(j, \{x_{ij}\}_{i\in\mathcal{N}}) \to y_j$
  Compute and output $y_j = \sum_{i=1}^{n} x_{ij}$.
- **PartialProof** $(j, \{x_{ij}\}_{i\in\mathcal{N}}) \to \sigma_j$
  Compute and output $\sigma_j = \prod_{i=1}^{n} g^{x_{ij}} = g^{\sum_{i=1}^{n} x_{ij}} = g^{y_j} = H(y_j)$.
- **PartialAggregate** $(g, h, \mathcal{S}\{proof_{SM,i}\}_{i\in\mathcal{S}} \to proof_{SM,a}$
  Given a subset of range proofs $\{proof_{SM,i}\}_{i\in\mathcal{S}}$ where $\mathcal{S} \subseteq \{1, ..., n\}$. Aggregate the values $\{D_i\}_{i\in\mathcal{S}}\{z_{x_i}\}_{i\in\mathcal{S}}\{z_{r_i}\}_{i\in\mathcal{S}} \mapsto D_a, z_{x_a}, z_{R_a}$ according to equation (3.3). Then construct and publish the aggregated range proof $proof_{SM,a} = (\{V_i\}_{i\in\mathcal{S}}, \{a_i\}_{i\in\mathcal{S}}, D_a, \{z_{x_i}\}_{i\in\mathcal{S}}, z_{x_a}, \{z_{\tau_i}\}_{i\in\mathcal{S}}, z_{R_a})$.
- **FinalEval** $(\{y_j\}_{j\in\mathcal{M}}) \to y$
  Compute and output $y = \sum_{j=1}^{m} y_j$.
- **FinalProof** $(\{\sigma_j\}_{j\in\mathcal{M}}) \to \sigma$
  Compute and output $\sigma = \prod_{j=1}^{m} \sigma_j = \prod_{j=1}^{m} g^{y_j} = g^{\sum_{j=1}^{m} y_j} = g^y = H(y)$.
- **Verify** $(\{\pi_i\}_{i\in\mathcal{N}}, x, y, \{RP_{a_i}\}_{i\in\mathcal{M}}) \to \{0, 1\}$
  Compute and output $\sigma = \prod_{i=1}^{n} \pi_i \wedge \prod_{i=1}^{n} \pi_i = H(y) \wedge \{$**VerifyAggregatedProof**$(RP_{a_i})\}_{i\in\mathcal{M}}$. Where **VerifyAggregatedProof** is the verification algorithm in Construction 7.

B

LHSS

Construction 8 : Inner-product argument

Goal: Given a Pedersen vector commitment $P_v = \boldsymbol{g}^{s}\boldsymbol{h}^{q}$ and a value c prove that the two vectors $\boldsymbol{s}, \boldsymbol{q}$ satisfies $\langle \boldsymbol{s}, \boldsymbol{q} \rangle = c$.

- **Prove** $(\mathbf{g}, \mathbf{h}, u, P_v, c, \mathbf{s}, \mathbf{q}) \rightarrow \mathit{proof}_{IP}$
  Let $y = \text{Hash}_{IP}(\mathbf{g}, \mathbf{h}, P_v, c) \in \mathbb{F}^*$ and compute $P'_v = u^{y \cdot c}P$. Let $\mathbf{l}, \mathbf{r}$ be two empty vectors. Run the recursive algorithm **GenerateProof**$(\mathbf{g}, \mathbf{h}, u^{x \cdot c}, P_v, c, \mathbf{s}, \mathbf{q}, \mathbf{l}, \mathbf{r})$ use the output $(g', h', u', P'_v, s', q', \mathbf{l}, \mathbf{r})$ to construct the inner product proof $\text{proof}_{IP} = (\mathbf{g}, \mathbf{h}, u', P_v, s', q', \mathbf{l}, \mathbf{r})$ and output proof_{IP}.
- **GenerateProof**$(\mathbf{g}, \mathbf{h}, u, P_v, \mathbf{s}, \mathbf{q}, \mathbf{l}, \mathbf{r}) \rightarrow (g, h, u, P_v, s, q, \mathbf{l}, \mathbf{r})$
  - If the dimension of the vectors $\mathbf{g}, \mathbf{h}, \mathbf{s}, \mathbf{q}$ drop the bold font and publish the proof $\mathit{proof}_{IP} = (g, h, P_v, u, s, q, \mathbf{l}, \mathbf{r})$.
  - Otherwise: Let $n' = n/2$ and define $c_L = \langle \boldsymbol{s}_{[:,n']}, \boldsymbol{q}_{[n',:]} \rangle$ and $c_R = \langle \mathbf{s}_{[n',:]}, \mathbf{q}_{[:,n']} \rangle$. Then use these variables to calculate $L = \mathbf{g}_{[n':]}^{\mathbf{s}_{[:n']}} \mathbf{h}_{[:n']}^{\mathbf{q}_{[n':]}} u^{c_L}$ and $R = \mathbf{g}_{[:n']}^{\mathbf{s}_{[n':]}} \mathbf{h}_{[n':]}^{\mathbf{q}_{[:n']}} u^{c_R}$. Append $L, R \in \mathbb{G}$ to the vectors $\mathbf{l}$ resp $\mathbf{r}$. Now update $y = \text{Hash}_{BP}(L, R)$, and update $\mathbf{g}' = \mathbf{g}_{[:n']}^{y^{-1}} \mathbf{g}_{[n':]}^{y}$, $\mathbf{h}' = \mathbf{h}_{[:n']}^{y} \mathbf{h}_{[n':]}^{y^{-1}}$ and the commitment $P'_v = L^{y^2} P R^{y^{-2}}$. Finally update the vectors $\mathbf{s}, \mathbf{q}$ to $\mathbf{s}' = \mathbf{s}_{[:n']} y + \mathbf{s}_{[n':]} y^{-1}$ and $\mathbf{q}' = \mathbf{q}_{[:n']} y^{-1} + \mathbf{q}_{[n':]} y$. Run the algorithm recursively, **GenerateProof**$(\mathbf{g}', \mathbf{h}', u, P'_v, \mathbf{s}', \mathbf{q}', \mathbf{l}, \mathbf{r})$ with the updated variables. Note that the vectors $\mathbf{g}, \mathbf{h}, \mathbf{s}, \mathbf{q}$ now have the dimension $n' = n/2$, hence performing the recursion until one-dimensional vectors will require $log\, n$ iterations.
- **Verify** $(\mathit{proof}_{IP} = (\mathbf{g}, \mathbf{h}, u, P_v, s, q, \mathbf{l}, \mathbf{r})) \rightarrow \{0, 1\}$
  For $i \in \{0, log(n)\}$ put $n = n/2$ and $y = \text{Hash}(\boldsymbol{l}[i], \boldsymbol{r}[i])$, then update the vectors $\boldsymbol{g}$ and $\boldsymbol{h}$ as well as the variable P'_v according to, $\boldsymbol{g}' = \boldsymbol{g}_{[:,n]}^{y^{-1}} \boldsymbol{g}_{[n,:]}^{\prime y}$, $\boldsymbol{h} = \boldsymbol{h}_l[:, n]^x \boldsymbol{h}_{[n,:]}^{y^{-1}}$ and $P'_v = L^{y^2} P R^{y^{-2}}$. After iterating over all i the dimension of the vectors $\boldsymbol{g}, \boldsymbol{h}$ is one and the bold font can be dropped. Compute $c = \langle s, q \rangle$ and accept if $P'_v = g^s h^r u^c$.

C
Aggregation

Consider two set membership proofs RP_1 and RP_2. Remember that for $i = 1, 2$ it hold that $RP_i = (V_i, a_i, D_i, z_{x_i}, z_{\tau_i}, z_{R_i})$ and $V_i = g^{\frac{\tau_i}{q+x_i}}, a_i = e(V_i, g)^{s_i} e(g, g)^{t_i}, D_i = g^{s_i} h^{m_i}, z_{x_i} = s_i - cx_i, z_{\tau_i} = t_i - c\tau_i$ and $z_{R_i} = m_i + cR_i$, where $s_i, t_i, m_i \in_R \mathbb{F}$, g, h are group elements of the group $\mathbb{G}$, q is a secret key not known by any party, $y = g^q$ is the public key known to any party-.The, here assumed same for both parties, challange is denoted$c$ and is publicly known, and finally x_i, R_i denoted the secret and randomness of prover $i = 1, 2$

The aim is to investigate is an aggregated proof, called RP without subscript, of RP_1 and RP_2 such that $RP = (V, a, D, z_x, z_\tau, z_R)$,$a \overset{?}{=} e(V, y)^c e(V, g)^{-z_x} e(g, g)^{z_\tau}$ and if equality it implies that $x_i \in \Phi$ for $i = 1, 2$. The aggregation is done according equation (3.1), it follows that,

$$\begin{aligned}
LHS =& a_1 a_2 = \Big(e(V_1, g)^{s_1} e(g, g)^{t_1}\Big)\Big(e(V_2, g)^{s_2} e(g, g)^{t_2}\Big) \\
=& \Big(e(g, g)^{\frac{\tau_1 s_1}{q+x_1}} e(g, g)^{t_1}\Big)\Big(e(g, g)^{\frac{\tau_2 s_2}{q+x_2}} e(g, g)^{t_2}\Big) = e(g, g)^{\frac{\tau_1 s_1}{q+x_1} \frac{\tau_2 s_2}{q+x_2} + t_1 + t_2} \\
RHS =& e(V, y)^c e(V, g)^{-z_x} e(g, g)^{z_\tau} = e(V_1 V_2, y)^c e(V_1 V_2, g)^{-z_{x1} - z_{x2}} e(g, g)^{z_{\tau 1} + z_{\tau 2}} \\
=& e(g, g)^{cq\left(\frac{\tau_1}{q+x_1} + \frac{\tau_2}{q+x_2}\right)} e(g, g)^{\left(-(s_1 - cx_1) - (s_2 - cx_2)\right)\left(\frac{\tau_1}{q+x_1} + \frac{\tau_2}{q+x_2}\right)} e(g, g)^{(t_1 - c\tau_1) + (t_2 - c\tau_2)} \\
=& e(g, g)^{\frac{\tau_1}{q+x_1}\left(cq - (s_1 - cx_1) - (s_2 - cx_2) - c(q + x_1)\right) + t_1} e(g, g)^{\frac{\tau_2}{q+x_2}\left(cq - (s_1 - cx_1) - (s_2 - cx_2) + c)(q + x_2)\right) + t_2} \\
=& e(g, g)^{\frac{s_1 \tau_1}{q+x_1} + t_1} e(g, g)^{\frac{\tau_1}{q+x_1}\left(cq + cx_1 - c(q + x_1) - (s_2 - cx_2)\right)} \\
& e(g, g)^{\frac{s_2 \tau_2}{q+x_2} + t_2} e(g, g)^{\frac{\tau_2}{q+x_2}\left(cq + cx_2 - c(q + x_2) - (s_1 - cx_1)\right)} \\
=& e(g, g)^{\frac{s_1 \tau_1}{q+x_1} + t_1} e(g, g)^{\frac{s_2 \tau_2}{q+x_2} + t_2} e(g, g)^{\frac{\tau_1}{q+x_1}\left(-(s_2 - cx_2)\right)} e(g, g)^{\frac{\tau_2}{q+x_2}\left(-(s_2 - cx_2)\right)} \\
=& e(g, g)^{\frac{s_1 \tau_1}{q+x_1} + t_1} e(g, g)^{\frac{s_2 \tau_2}{q+x_2} + t_2} e(V_1, g)^{-z_{x2}} e(V_2, g)^{-z_{x1}} \\
& \implies LHS \neq RHS
\end{aligned}$$

It is seen that the terms $e(V_1, g)^{-z_{x2}} e(V_2, g)^{-z_{x1}}$ are not cancelled, which results in that the left hand side does not equal the right hand side.