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

Cover: Description of the picture on the cover page (if applicable)

Typeset in LaTeX
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: Computer, science, computer science, engineering, project, 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, April 2021

Contents

List of Figures

List of Tables

1
Introduction

1.1 Contribution

1.2 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 behind the construction of a client and server verifiable additive homomorphic secret sharing construction presented in chapter 3. First the preliminaries are described, including notation, theorems, definitions, assumptions and cryptographic preliminaries and concepts. Then the VAHSS construction [17] [16], which this reports aims to extend to verify clients input is explained. Finally two range proof and a set membership constructions is presented.

2.1 Preliminaries

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

The discrete logarithm assumption and q-strong Diffie Hellman assumption define below does not hold in the presence of quantum computers. All cryptographic constructions presented in this paper relies on one or both of these two assumptions, hence the security is not guaranteed post quantum.

Definition 1 (**Pseudorandom Function (PRF)**). *Let S be a distribution over $\{0,1\}^l$ and $F_s : \{0,1\}^m \to \{0,1\}^n$ 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 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 in 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

Secret sharing, first mentioned in [13], hides a secret x by splitting it into shares, where 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 in order to reconstruct the value x at least τ shares has to be combined, this is called a (τ, m)-threshold scheme. I this paper the threshold will be equal to the number of shares, $\tau = m$. and additive secret sharing scheme will be considered. Additive secret sharing means that to reconstruct the secret at least τ shares are added, $x = \sum_{i=1}^{\tau} x_i$.

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 such function satisfying the thee properties is $\mathcal{H}_1(x) : \mathbb{F} \mapsto \mathbb{G}$ and $\mathcal{H}_1(x) = g^x$ [18].

Pedersen Commitment scheme

A commitment to a secret $x \in \mathbb{F}$ is the *Pedersen commitment scheme* defined as $\mathbb{E}(x, R) = g^x h^R$, where $R \in_R \mathbb{F}$, originally presented in [12]. This 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 we have two commits satisfying $\mathbb{E}(x, R) = \mathbb{E}(x', R')$ $x \neq x'$ and $x \neq x'$ then it must hold that $R \neq R' \bmod q$ and*

$$log_g(h) = \frac{x - x'}{R' - R} \ mod\ N. \tag{2.1}$$

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 $x \neq x'$. However the $log_g h$ is assumed to be unknown hence it is not possible 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 [12].

Further note that 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 $C_1 \cdot C_2 = \mathbb{E}(x_1 + x_2, R_1 + R_2)$.

A final remark about the Pedersen commitment is the similarity between the hash function $\mathcal{H}_1$ and the Pedersen commitment $\mathbb{E}$, the hash function can be seen as a generalisation of the Pedersen commitment. This will be used later when including verification of client to the VAHSS construction.

A Pedersen commitment scheme can also be defined for vectors and is then called *Pedersen vector commitment*. Lets 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 refereed to as bilinear pairing) maps two group elements from one group 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 to another group, generally the definition of admissible bilinear maps two elements from different groups to a third group, i.e $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$, but in this paper it will always hold that $\mathbb{G}_1 = \mathbb{G}$ 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 and the idea to behind is explained briefly. Bohen-Boyen presented a a signature scheme that exploits the bilinear property of the mapping e to verify the signatures [2]. Shorty the scheme is constructed as, the signer knows the secret key x and distributed the public key g^x, then to sign a message m 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 [9]. 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 non interactive ZKP that ensures proof of knowledge (PoK) is of interest. Before closer defining what this means lets consider the set up and environment of ZKP protocol. 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 know a secret $x \in \mathcal{L}$. The prover constructs a proof that x belongs to $\mathcal{L}$, by using a witness w of x, then 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 and PoK means that the verifier is not only convinces there exist a witness w but also that the prover knows such a witness. A ZKP should fulfil the thee properties defined in Definition 4, these definitions informally means that, a correctly constructed proof of an instance $x \in \mathcal{L}$ should be accepted with probability 1, an incorrect constructed 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;* $\texttt{Prove}(x, w)$ *to be the algorithm for generating a ZKP of instance* $x \in \mathcal{L}$ *and witness* w*, and* `Verify` *to be the verification algorithm of the correctness 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).

2.2 Verifiable additive homomorphic secret sharing

This section will describe a verifiable additive homomorphic secret sharing (VAHSS) Lets assume n clients/data providers and m servers, to simplify notation define the two sets $\mathcal{N} = \{1, ..., n\}$ and $\mathcal{M} = \{1, ..., m\}$. Let c_i and x_i for $i \in \mathcal{N}$ denote the clients (data providers) and their respective data. Denote the servers by s_j, $j \in \mathcal{M}$. The idea of VAHSS is that each client split their secret x_i into m shares, denoted x_{ij} and 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 is then computed by summing the partial sums, this gives $y = \sum_{j=1}^{m} y_j$, this can be computed by any party since the partial sums are public. In verifiable additive homomorphic secret sharing 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 severs computations. Remark that the individual secrets x_i is never revealed in the protocol.

Construction

A construction of VAHSS was presented in [17] and implemented and tested in [16]. The construction consists of the six PPT (probabilistic polynomial time) algorithms: **ShareSecret**, **PartialEval**, **PartialProof**, **FinalEval**, **FinalProof** and **Verify**. The clients/data providers executed the step **ShareSecret**, the servers **PartialEval** and **PartialProof** and the last three steps can run by anyone. A full description of the construction and all six algorithms is seen in Construction 1.

To obtain a secret sharing protocol such that any true subset of shares reviles no information about the secret the construction makes use the following polynomial. For each client, c_i, let $\theta_{i1}, ..., \theta_{im} \in \mathbb{F} \backslash \{0\}$ and $\lambda_{i1}, ..., \lambda_{im} \in \mathbb{F}$ such that the following property for polynomial p_i holds,

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

Note that is the step **ShareSecret** the shares are put to $x_{ij} = \lambda_{ij} p_i(\theta_{ij})$ and the polynomial $p_i(X)$ is a t-degree polynomial defined as $p_i(X) = x_i + \sum_{k=1}^{t} a_k X^k$, thus $\sum_{j=1}^{m} x_{ij} = \sum_{j=1}^{m} \lambda_{ij} p_i(\theta_{ij}) = p_i(0) = x_i$. Which shows that the proposed shares x_{ij} does adds to the secret x_i if all shares are in the sum and the secret is hidden else.

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** $(1^\lambda, i, x_i) \rightarrow (\tau_i, \{x_{ij}\}_{j \in \mathcal{M}})$
  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** $(j, \{x_{ij}\}_{i \in \mathcal{N}}) \rightarrow y_j$
  Compute and output $y_j = \sum_{i=1}^{n} x_{ij}$.
- **PartialProof** $(j, \{x_{ij}\}_{i \in \mathcal{N}}) \rightarrow \sigma_j$
  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** $(\{y_j\}_{j \in \mathcal{M}}) \rightarrow y$
  Compute and output $y = \sum_{j=1}^{m} y_j$.
- **FinalProof** $(\{\sigma_j\}_{j \in \mathcal{M}}) \rightarrow \sigma$
  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** $(\{\tau_i\}_{i \in \mathcal{N}}, \sigma, y) \rightarrow \{0, 1\}$
  Compute and output $\sigma = \prod_{i=1}^{n} \tau_i \wedge \prod_{i=1}^{n} \tau_i = \mathcal{H}(y)$.

A HSS/additive-HSS construction should satisfy two requirements: *Correctness* and *Security*. A verifiable additive HSS should also satisfy *Verifiability*. The exact definition of the three requirements for Construction 1 is given in [17] and Theorem 3 states that the construction do fulfil these requirements.

Theorem 3. *Construction 1 satisfies the correctness, security and verifiability requirements defined in [17].*

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 with out 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:

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

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 defied below for the separate constructions.

Let's denote two parties prover and verifier as $\mathcal{P}$ respectively $\mathcal{V}$ and explain the statements in equations (2.3), (2.4) informally: After successfully performing a range proof $\mathcal{P}$ has convinced $\mathcal{V}$, that the secret x in a Pedersen commitment C is in an predetermined allowed range (or set) without $\mathcal{V}$ learning anything else about x.

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

In the two subsections below the theory and construction of *Set membership proofs & Signature based range proofs* and *Bulletproofs* are presented. Both range 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 the zero knowledge set membership 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 [7]. Both the ZKSM and ZKRP constructions are modified compared to the original construction according to the Fiat-Shamir heuristic to be non-interactive.

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 commitment, 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 prove). The prover who aims to prove that the secret hidden by a pre published Pedesen commitment, denoted C, is in the allowed set Φ chooses the commitment representing the the secret x, i.e A_x. Then hides this choice by raising A_x to a random value $\tau \in_R \mathbb{F}$, this gives $V = A_x^\tau$, and publishes V. Then the prover has to convince the verifier that 1) the published value V is indeed equal to A_x^τ where A_x is 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 ZKSM algorithms that both constructs the public commitments and convinces the verifier of the above statements is given. The notation $e(\cdot, \cdot)$ in Construction 2 and 3 refers to an admissible bilinear mapping as defined previously in section 2.1.

The ZKSM construction can be turned into a efficient zero knowledge range

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** $(g, h, \Phi) \rightarrow (y, \{A_i\}_{i\in\Phi})$
  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** $(g, h, C, \Phi) \rightarrow$ $proof_{SM} = (V, a, D, z_x, z_\tau, z_R)$
  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 $a = e(V, g)^{-s} e(g, g)^t$ ($e(\cdot, \cdot)$ is a bilinear mapping as described above), $D = g^s h^m$, and $c = \text{Hash}(a, D)$. Finally compute $z_x = s - xc$, $z_R = m - Rc$ and $z_\tau = t - \tau c$ then construct and publish $proof_{SM} = (V, a, D, z_x, z_t au, z_R)$.
- **Verify** $(g, h, C, proof) \rightarrow \{0, 1\}$ Check if $D \stackrel{?}{=} C^c h^{z_R} g^{z_x} \wedge a \stackrel{?}{=} e(V, y)^c e(V, g)^{-z_x} e(g, g)^{z_\tau}$. If the equality holds the prover has convinced the verifier that $x \in \Phi$ return 1 otherwise return 0.

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 described in [7]. 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 though the number $[0, l-1]$ and not $[0, l]$. This has been wrongly notated [7, 10] and therefore an explicit proof of this is given in Appendix A. 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 must hold that $x \in [a, b]$. 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 a `AND` operation to verify that $x - a \in [0, u^l)$ *and* $x - b + u^l \in [0, u^l)$. In [8] an optimised implementation is presented reducing the complexity with a factor 2. This rather small reduction is important when a verifier is required check the range of multiple clients secrets, which is the case in VAHSS where it is done once for each client.

2.3.2 Bulletproofs

Bulletproof is a range proof, logarithmic in the size of the range. The proof relies on the inner product argument which allows a prove to convince a verifier that he knows the opening $\boldsymbol{s}, \boldsymbol{q} \in \mathbb{F}^n$ to a Pedersen vector commitment $P_v = \boldsymbol{g}^{\boldsymbol{s}} \boldsymbol{h}^{\boldsymbol{q}}$ such 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 $D \stackrel{?}{=} C^c h^{z_R} \prod_{j \in \mathbb{Z}_l} (g^{u^j z_{x_j}}) \wedge a_j \stackrel{?}{=} e(V_j, y)^c e(V_j, g)^{-z_{x_j}} e(g, g)^{z_{\tau_j}}$ 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.

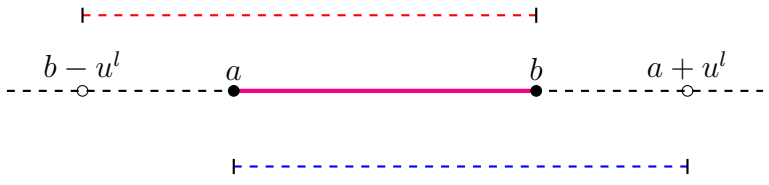


Figure 2.1: Illustration of generalisation to arbitrary intervals $[a, b]$ for range proofs

the inner product of $\boldsymbol{s}, \boldsymbol{q}$ is equal to a known value, c. This can be done with a proof of size $log\ n$, compare to the trivial solution of publishing $\boldsymbol{s}, \boldsymbol{q}$ which is a proof of size n.

Notation and SetUp

The description and construction of Bulletproofs uses some additional notation, the notation used here is equivalent to the notation used in original Bulletproof paper, so for a detailed review of the notations please see this [6] .

Remark that in this section n will denotes 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 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 assumptions about g, h are 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 logarithm relation among $\mathbf{g}, \mathbf{h}$. 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, [10].

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 a argument of knowledge of $\mathbf{s}, \mathbf{q}$ in a Pedersen vector commitment $P_v = \mathbf{g^s h^q}$ satisfying a given inner product denoted c. More formally the argument is a proof system of the statement,

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

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

$$\{(\mathbf{g}, \mathbf{h} \in \mathbb{G}^n,\ u, P_v \in \mathbb{G};\ \mathbf{s}, \mathbf{q} \in \mathbb{F}^n) :\ P_v = \mathbf{g^s h^q} u^{\langle \mathbf{s}, \mathbf{q} \rangle}\}. \tag{2.5}$$

A logarithmic sized proof of the above inner product statement is presented in Construction 4. The construction presented is modified compared to the one presented in [6] 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 relieved.

Inner product rang proof

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

Construction 4 : Inner-product argument

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

- **Prove** $(\mathbf{g}, \mathbf{h}, u, P_v, c, \mathbf{s}, \mathbf{q}) \rightarrow proof_{IP}$
  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 $proof_{IP} = (g, h, P_v, u, s, q, \mathbf{l}, \mathbf{r})$.
  - 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 $P'_v = L^{y^2}PR^{y^{-2}}$. 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** $(proof_{IP} = (\mathbf{g}, \mathbf{h}, u, P_v, s, q, \mathbf{l}, \mathbf{r})) \rightarrow \{0, 1\}$
  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, $\boldsymbol{g}' = \boldsymbol{g}_{[:,n]}^{y^{-1}}\boldsymbol{g}'^{y}_{[n,:]}$, $\boldsymbol{h} = \boldsymbol{h}_l[:, n]^x\boldsymbol{h}_{[n,:]}^{y^{-1}}$ and $P'_v = L^{y^2}PR^{y^{-2}}$. After iterating over all i the dimension of the vectors $\boldsymbol{g}, \boldsymbol{h}$ is one and we can drop the bold font. Compute $c = \langle s, q \rangle$ and accept if $P'_v = g^s h^r u^c$.

to the interval $[0, 2^n)$. 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;

$$\left\langle \boldsymbol{x} - z \cdot \mathbf{1}^n, \boldsymbol{y}^n \circ (\bar{\boldsymbol{x}} + z \cdot \mathbf{1}^n) + z^2 \cdot \mathbf{2}^n \right\rangle = z^2 \cdot x + \delta(y, z), \tag{2.6}$$

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 are the output of a hash function in a non-interactive construction. Here a non-interactive construction will be considered.

Directly using a inner product argument presented in Construction 4 to prove the statement in equation (2.6) 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 even if the inner product argument is not a zero knowledge construction. Given this idea, the inner product in (2.6) 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 (2.6). 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 and innner product Construction 5 gives a non interactive zero knowledge range proof that the secret x belongs to the interval $[0, 2^n]$.

Construction 5 : 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** $(g, h, \mathbf{g}, \mathbf{h}, P, n, x, R, u) \rightarrow \ \mathit{proof}_{RP}$
  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 4 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: $\mathit{proof}_{RP} = (\tau_X, \mu, \hat{t}, P, A, S, T_1, T_2, P_v, \mathit{proof}_{IP})$.
- **Verify** $(g, h, C, \mathit{proof}_{RP}) \rightarrow \{0, 1\}$
  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: $P_l \stackrel{?}{=} P_r \wedge g^{\hat{t}}h^{\tau_X} \stackrel{?}{=} P^{z^2}\, g^{\delta(y,z)}\, T_1^x\, T_2^{x^2}$ and if the output of **Verify** in Construction 4 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 to 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 discussed above to get an understanding of their advantages and disadvantages.This will be used to then determine which are more suitable to use in a extended VAHSS construction.. In the section 3.2 details on how to construct a VAHSS scheme that ensures honest clients is derived. The next section 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 is discussed.

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 to obtain an understanding of their suitability to be combined with the VAHSS scheme described in Construction 1 to verify clients honesty. The aspects that is of interest in the evaluation of the range proofs and their compatibility with the VAHSS construction is presented is the below list;

- Computation complexity (for setup, prover verifier)
- Proof size (communication complexity)
- Flexibility of range

Remark that all of the range proof considered aim to prove that the secret in a Pedersen commitment is in an allowed range (or set). Thus to combine any of the range proofs 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 evaluation.

A small first analysis is that the verification time of the range proofs and set membership proofs will become mostly important when used in a VAHSS construction, this is since there are usually many clients participating and thus the verification will, without aggregation as discussed later, grow linear with the number of clients. This is further discussed later in section 3.3.

3.1.1 Theoretical analysis

The computational complexity will not be compared theoretically since the Bulletproofs is considerable different in the construction compared to the set membership proof and signature based range proof. The latter requires bilinear mappings unlike Bulletproofs, bilinear mappings are relative expensive operation compared to for example group exponentials which are dominating the computational complexity for Bulletproofs. Therefore instead of comparing the theoretical computational complexity the runtime for prototype implementation of the different proof is considered in the next section.

However here the communicational complexity for the set up phase for the set membership proof will be discusses. The set membership proof requires $n = |\Phi|$ digital signatures to be known by both prover and verifier, one signature for each elements in Φ. This 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 application 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 communicational complexity of the proofs is not the most important feature of comparison, 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 theoretically smallest and signature based range proofs largest.

A big 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. Actually Bulletproofs can be modified to also 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 lets a assume a prover wants to prove the range of k commits a naive implementation would lead to a proof size of $k \cdot log_2 n$, but 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 $2log_2 2 = 2$. The computational complexity grows.. TODO Both for signature-based range proof and Bulletproof it is not the actual 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.

Table 3.1: Time Complexity comparison for range proof, values are computed as described in section 3.1.2. The runtime are for implementations written in Golang and the implementations are not guaranteed to be optimized in runtime.

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

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 even a small range of large numbers results in longer runtime. 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. Concluding set membership proof allows proving belonging to much more flexible sets than the range proofs, but both range proofs considered can prove belonging to ranges $[a, b]$, where $a > 0$ and $b > a$.

3.1.2 Prototype Analysis

Implementation of Bulletproofs and signature-based range proof has been done and compared between them self in [7], this 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 [7] is benchmarked for all three constructions. The reason for redoing the benchmarking for Bulletproof and signature based range proofs is since else the hardware differences would not lead to a fair comparison. Table 3.1 shows the time complexity comparison between Bulletproofs and 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 [7] and the code [11] except the hardware. The set used in the set membership contain 182 elements, which thereby is the same amount of elements in the set as length of 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 then Bulletproofs, especially in the verification. The runtime for verifying the proofs is most important for applications with multiple clients since a verifier has to verify all clients. The runtime for verification between Bulletproofs and Set memberships proofs are rather similar, although Bulletproofs are somewhat faster, which as mentioned is of importance in applications with many clients.

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

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 this VAHSS construction to verify both client and servers. The clients does however not which to reveal their secrets. A method for verifying clients honesty is zero-knowledge range proof or set membership proofs. If such a proof was included in the VAHSS construction then, under the assumption that there exist an allowed rang or set to which the input must belong, a potentially malicious client can only have limited influence on the computed sum. This is due to that a malicious input must still belong to the allowed range or set and hence the impact on the sum is bounded by the size of the range or the elements in the set.

Next the combining of range proofs and the VAHSS construction will be discussed, it is not sufficient to construct and perform a range proof and VAHSS scheme separately, since then 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. Remark that all considered range 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 range proofs construction differ considerably. Hence the possibility to exploit the Pedersen commitment to link the VAHSS construction with a range proof is investigated, more precisely a link between the shares hiding the secret generated in the algorithm **ShareSecret** in the VAHSS construction and the secret hidden in the Pedersen commitment used in the range proof is desired to convince the verifier that the shares represents a secret that is in the allowed range. As mentioned publishing a Pedersen commitment of the secret itself does not provide any guarantee that it is the same secret that is hidden in shares. Committing to the shares in the Pedersen commitment is neither an option, 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. The aggregation used in the VAHSS construction to prove the honesty of the servers is tested if it can also be used to also connect the range proofs to the VAHSS construction.

Recall that the clients except from the shares also publishes the checksum τ_i for the secret x_i, more precisely the definition of the checksum is $\tau_i = g^{x_i+R_i}$, where R_i chosen uniformly at random. This checksum is indeed equal to a Pedersen commitment where $g = h$. Using this checksum as the Pedersen commitment in the constructing of a range proof would be sound. However if $g = h$ the computationally biding 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 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. Lets instead investigate modifying the checksum τ_i to a Pedersen commitment. Let the clients compute and output $\pi_i = g^{x_i} h^{R_i}$, where x_i, R_i, g, h are as defined above, instead of τ_i as before. 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 by the range proof.

Assume that client k commits to the value $\hat{x}_k$ in the Pedersen commitment π_k and generates a range proof that the secret hidden in the commitment belongs to the interval $[a, b]$ but constructs shares $\{x_{kj}\}_{j=1}^{m}$ such that $\sum_{j=1}^{m} x_{kj} = x_k \neq \hat{x}_k$. Then when verification of the servers honest it will not hold that $\prod_{i=1}^{m} \pi_i = g^y$. Therefore the verification will return false and the protocol will not succeed even if the range proof does. Although any cheating party will be detected, it will not be possible do determined which party that cheated more precisely not even if the cheating party was a client or a server.

In Construction 6 the extended VAHSS is described in detail. In order to clarify the modifications made to include a range proof, lets briefly mention some differences to the VAHSS construction presented in [17] ans also presented in Construction 1. 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**, this algorithm can be included in either **ShareSecret** or **RangeProof** instead of being viewed as a separate algorithm, in the implementation discussed below 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 and secret x_i. Note that it is not specified which range proof construction that is used since it does not affect the rest of construction as long as the verification algorithm used to verify the range proof is the compatible with the construction of the proof. Both algorithms **ConstructRangeProof** and **RangeProof** are executed by the clients.. The algorithm **Verify** additionally to the steps of the algorithm in Construction 1 also verifies the correctness of the range proof RP_i and an additional `AND` operator to compute the total verification.

Theorem 4. *The client and server verifiable AHSS presented in Construction 6 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$(\{\pi_i^{'}\}_{i \in \mathcal{N}}, x, y, \{RP_i\}_{i \in \mathcal{N}}) = 1$.

Construction 6 : 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 $p_i(X) = x_i + a_1 X + ... + a_t X^t$. 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 5. 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 $\sigma = \prod_{i=1}^{n} \pi_i \wedge \prod_{i=1}^{n} \pi_i = H(y) \wedge \{\textbf{Verify}_{rp}(RP_i)\}_{i \in \mathcal{N}}$. Where $\textbf{Verify}_{rp}$ is the verification algorithm associated with the algorithm used by the clients to construct the range proofs, $\{RP_i\}_{i \in \mathcal{N}}$.

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 6 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 $i = 1, ..., n$.

□

3.3 Aggregate range proofs

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 6. This would decrease the runtime significantly in the verification step, especially in implementations where hundred 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 $\pi = \prod_{i=1}^{n} \pi_i$ and then construct a range proof for the aggregated commitment $\pi = g^{\sum_{i=1}^{n} x_i}$ over 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 $i \in \mathcal{N}$.

Now lets examine the possibility to aggregate the set membership and signature based range proofs, the construction of these two are similar and hence it is sufficient to consider one of them, due to its simpler notation the set membership proof is considered. Assume 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 $i = 1, 2$, additionally the commitment C_i, $i = 1, 2$ and group elements h, g are know to 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,

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

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 [3]. However the homomorphic properties of the Pedersen commitment and bilinear maps does not guarantee that the range proofs is homomorphic.

Lets see if the aggregated proof RP can successfully be verified. For the verification to succeed it must hold that, 1) $D \stackrel{?}{=} C^c h^{z_R} g^{z_x}$ and 2) $a \stackrel{?}{=} e(V,y)^c e(V,g)^{-z_x} e(g,g)^{z_\tau}$, so lets check if it holds starting with the first.

$$\begin{aligned} LHS &= D = D_1 * D_2 = g^{s_1+s_2} * h^{m_1+m_2} \\ RHS &= C^c h^{z_R} g^{z_x} = (C_1 * C_2)^{c_1 c_2} h^{z_{R_1}+z_{R_2}} g^{z_{x_1}+z_{x_2}} \\ &= (g^{x_1} h^{R_1} g^{x_2} h^{R_2})^{c_1 c_2} h^{m_1 - R_1 c_1 + m_2 - R_2 c_2} g^{s_1 - x_1 c_1 + s_2 - x_2 c_2} \\ &= g^{c_1 c_1 (x_1+x_2) + s_1 + s_2 - x_1 c_1 - x_2 c_2} h^{c_1 c_2 (R_1+R_2) + m - R_1 c_1 - R_2 c_2} \end{aligned}$$

$\implies$ LHS $\neq$ RHS.

This is since $c_1 c_2 (x_1 + x_2) = c_1 c_2 x_1 + c_1 c_2 x_2 \neq x_1 c_1 + x_2 c_2$ and hence the terms does not cancel such that the RHS is independent of x_1, x_2, c_1, c_2 unlike the LHS. Further note that if $c_1 = c_2$ then this would be an equality. Clearly it cannot be guaranteed that the two challenges will be equal since they depend on randomness in the proof construction. Lets instead investigate if some cleaver aggregation could circumvent this. Again consider the two range proofs RP_1, RP_2, but only the first equality test in the verification is of interest. The reason only the first is considered is to simplify the procedure, and a remark is that although only half the verification is aggregated it can still lead to important reduce of computations for the verifier. The goal is now to combine the two range proofs into one aggregated proof such that the first equality holds. First calculate the challenges c_1, c_1 as $c_i = Hash(D_i, a_i)$, $i = 1, 2$. 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 z_τ. Further the aggregated proof is defined as,

$$
\begin{aligned}
D &= D_1^{c_2} \cdot D_2^{c_1} = (g^{s_1} h^{m_1})^{c_2} \cdot (g^{s_2} h^{m_2})^{c_1} = g^{s_1 c_2 + s_2 c_1} h^{m_1 c_2 + m_2 c_1} \\
z_x &= c_2 z_{x_1} + c_1 z_{x_2} = c_2(s_1 - x_1 c_1) + c_1(s_2 - x_2 c_2) = s_1 c_2 + s_2 c_1 - c_1 c_2 (x_1 + x_2) \\
z_R &= c_2 z_{R_1} + c_1 z_{R_2} = c_2(m_1 - R_1 c_1) + c_1(m_2 - R_2 c_2) = m_1 c_2 + m_2 c_1 - c_1 c_2 (R_1 + R_2) \\
&= m_1 c_2 + m_2 c_1
\end{aligned}
$$

Additionally also define the aggregated challenge and commitment,

$$
\begin{aligned}
c &= c_1 c_2 \\
C &= C_1 C_2 = (g^{x_1} h^{R_1})(g^{x_2} h^{R_2}) = g^{x_1 + x_2} h^{R_1 + R_2} = g^{x_1 + x_2}.
\end{aligned}
$$

Above it is assumed that the random values R_i is chosen such that $R_n = \phi(N)\lceil \frac{\sum_{i=1}^{n-1} R_i}{\phi(N)} \rceil - \sum_{i=1}^{n-1} R_i$, hence for $i = 1, 2$ if follows that $R_2 = \phi(N)\lceil \frac{R_1}{\phi(N)} \rceil - R_1$, and thus $h^{c_1 c_2 (R_1 + R_2)} = 1$. 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 of range proof to construct RP' lets again evaluate if $D \stackrel{?}{=} C^c h^{z_R} g^{z_x}$,

$$
\begin{aligned}
LHS &= D = D_1^{c_2} \cdot D_2^{c_1} = g^{s_1 c_2 + s_2 c_1} h^{m_1 c_2 + m_2 c_1} \\
RHS &= C^c h^{z_R} g^{z_x} = (C_1 C_2)^{c_1 c_2} h^{c_2 z_{R_1} + c_1 z_{R_2}} g^{c_2 z_{x_1} + c_1 z_{x_2}} \\
&= (g^{x_1 + x_2})^{c_1 c_2} h^{m_1 c_2 + m_2 c_1} g^{s_1 c_2 + s_2 c_1 - c_1 c_2 (x_1 + x_2)} \\
&= g^{(x_1 + x_2) c_1 c_2 - c_1 c_2 (x_1 + x_2) + s_1 c_2 + s_2 c_1} h^{m_1 c_2 + m_2 c_1} = g^{s_1 c_2 + s_2 c_1} h^{m_1 c_2 + m_2 c_1}
\end{aligned}
$$

$\implies$ LHS = RHS.

This means that the new range proof RP' based on the two separate range proofs RP_1, RP_2 as above satisfies the first equality of the verification, further lets see if this can be extended to an aggregation on arbitrary number of range proofs, hence consider n clients and hence n range proofs denoted RP_i, $i \in \mathcal{N}$.

$$
\begin{aligned}
D &= \prod_{i=1}^{n} D_i^{\prod_{j=1, j\neq i}^{n} c_j} = \prod_{i=1}^{n} (g^{s_i} h^{m_i})^{\prod_{j=1, j\neq i}^{n} c_j} = g^{\sum_{i=1}^{n} \left(\prod_{j=1, j\neq i}^{n} c_j \right) s_i} h^{\sum_{i=1}^{n} \left(\prod_{j=1, j\neq i}^{n} c_j \right) m_i} \\
z_x &= \sum_{i=1}^{n} \Big(\prod_{j=1, j\neq i}^{n} c_j \Big) z_{x_i} = \sum_{i=1}^{n} \Big(\prod_{j=1, j\neq i}^{n} c_j \Big) s_i - \Big(\prod_{j=1}^{n} c_j \Big) \sum_{i=1}^{n} x_i \\
z_R &= \sum_{i=1}^{n} \Big(\prod_{j=1, j\neq i}^{n} c_j \Big) z_{R_i} = \sum_{i=1}^{n} \Big(\prod_{j=1, j\neq i}^{n} c_j \Big) m_i - \Big(\prod_{j=1}^{n} c_j \Big) \sum_{i=1}^{n} R_i = \sum_{i=1}^{n} \Big(\prod_{j=1, j\neq i}^{n} c_j \Big) m_i.
\end{aligned}
\tag{3.2}
$$

Let $c = \prod_{i=1}^{n} c_i$ be the product of all challenges and $C = \prod_{i=1}^{n} C_i$ the product of the commitments. Then define the aggregated range proof $RP = (D, z_x, z_r)$ and

examine if it holds $D \stackrel{?}{=} C^c h^{z_R} g^{z_x}$,

$$
\begin{aligned}
LHS =& D = g^{\sum_{i=1}^{n}\left(\prod_{j=1,j\neq i}^{n} c_j\right)s_i} h^{\sum_{i=1}^{n}\left(\prod_{j=1,j\neq i}^{n} c_j\right)m_i} \\
RHS =& C^c h^{z_R} g^{z_x} = \left(\prod_{i=1}^{n} C_i\right)^{\prod_{i=1}^{n} c_i} h^{\sum_{i=1}^{n}\left(\prod_{j=1,j\neq i}^{n} c_j\right)m_i} \\
& g^{\sum_{i=1}^{n}\left(\prod_{j=1,j\neq i}^{n} c_j\right)s_i - \left(\prod_{j=1}^{n} c_j\right)\sum_{i=1}^{n} x_i} \\
=& \left(g^{\sum_{i=1}^{n} x_i}\right)^{\prod_{i=1}^{n} c_i} h^{\sum_{i=1}^{n}\left(\prod_{j=1,j\neq i}^{n} c_j\right)m_i} g^{\sum_{i=1}^{n}\left(\prod_{j=1,j\neq i}^{n} c_j\right)s_i - \left(\prod_{j=1}^{n} c_j\right)\sum_{i=1}^{n} x_i} \\
=& g^{\sum_{i=1}^{n}\left(\prod_{j=1,j\neq i}^{n} c_j\right)s_i} h^{\sum_{i=1}^{n}\left(\prod_{j=1,j\neq i}^{n} c_j\right)m_i}
\end{aligned}
$$

$\implies$ LHS = RHS.

Now lets examines the possibilities to construct a method of aggregating range proof such that also the second equality in the verification also holds after the aggregation. Hence lets test if $a \stackrel{?}{=} e(V,y)^c e(V,g)^{-z_x} e(g,g)^{z_\tau}$, where a, V, z_x, z_τ is as in equation (3.1), under the assumption $c_1 = c_2$. If it hold true if would be sufficient to use the same trick as above, namely aggregate such that the challenges appear only as a product. After some calculation it is realised that the terms,

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

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 a naive aggregation as in equation (3.1), which was the case for the other equality check, hence it is realised that in order to aggregate the whole range proof additional trick is required. Several attempts to construct an aggregation that will cancel these terms has been made without any success. Therefore whether it can be done or not will be left unanswered. What can be seen as an indicator to that is might not be possible is verification of Construction 3, where the equality check for the bilinear mapping statement is done separately for each $j \in \mathbb{Z}_l$.

Thus only the first equity in the verification of Construction 2 has been aggregated. Leading to that a verification algorithm verifying n range proofs to be on the form where $c = \prod_{i=1}^{n} c_i$ but not calculated by the *aggregator*,

- **Verify**$\big(g, h, y, \{C_i\}, c, proof = (\{V_i\}, \{a_i\}, D, z_x, \{z_{x_i}\}, \{z_{\tau_i}\}, z_R)) \to \{0,1\}$
  Check if $D \stackrel{?}{=} (\prod_{i=1}^{n} C_i)^c h^{z_R} g^{z_x}$ and $a_i \stackrel{?}{=} e(V_i,y)^c e(V,g)^{-z_x} e(g,g)^{z_\tau}$ for all $i = 1, ..., n$. If all equalities hold output 1 else output 0.

Comparing this to running the verification in set membership construction once for each clients, it is seen that he first equality check is now run only once instead of n times.

It remains to argue that the range proof fulfils the completeness, soundness and zero-knowledge requirements states in Definition 4 after the aggregation of D, x_R, z_x. Assume that the aggregation has been done in a correctly by a trusted party. Completeness follows from the argument above that it still holds that $D \stackrel{?}{=} C^c h^{z_R} g^{z_x}$.

It is also clear that multiplying and adding elements which perfectly hides the secret with other elements independent of the secret will not reveal any information about the secret, hence the zero-knowledge property still holds. This leads to that it remains to be argued that the soundness still holds. In the set membership proof the equality check $D \stackrel{?}{=} C^c h^{z_R} g^{z_x}$, servers the purpose of ensuring the verifier that the secret hidden in the commitment C is the same as the secret in z_x (plus that the randomness in the commitment is the same as in z_R). Hence this property need to be checked if it remains after the aggregation. To test this, lets again consider the two dimensional case where the range proofs RP_1 and RP_2 are aggregated into $RP = (\{V_1, V_2\}, \{a_1, a_2\}, D, z_x, \{z_{x_1}, z_{x_2}\}, \{z_{\tau_1}, z_{\tau_2}\}, z_R))$ and the combined challenge $c = c_1 c_2$ is also computed and made available. Further test if it can be that $D = (C_1 C_2)^c h^{z_R} g^{z_x}$, 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 or 2 or both.

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

Under the assumption that the aggregation was done correctly one of the following 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. $x_2 = modinv(x_1)$ and $\tilde{x}_2 = modinv(\tilde{x}_1)$
4. $x_1 = -modinv(\tilde{x}_1)$ and $x_2 = -modinv(\tilde{x}_2)$ or $x_1 = -modinv(\tilde{x}_2)$ and $x_2 = -modinv(\tilde{x}_1)$ WHY?

The first case corresponds to honest parties. Under the assumption that clients cannot communicate with each other both case two and case three has a negligible probability of succeeding, since the probability of two clients collaboration in such way by change is sufficiently small. Lets examine case two and three for instances where communication between clients is possible. In the second case the parties are cheating, since they do not use the same secret in the commitment (C_i) as in z_{x_i}, however the sum y calculated the servers in the VAHSS construction evaluate to the same value in both case 1 and 2, and all used terms in the sum is verified to be in the set Φ. So despite cheating clients the result from the protocol is unaffected by this cheating. 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. This leaves the third and fourth case to be examined.

Next lets briefly consider Bulletproofs. The original paper about Bulletproofs [6] 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 allowed range. This however only works if all parties uses the same challenge c in the proof construction, this is achieved by introducing a dealer. The dealer can be either one of the clients or another party. 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 5 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. The following steps is not describe here instead it is noted that although the Fiat-Shamir heuristic is used to generate the challenges it 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 construction. An further 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 6 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 6.

Implementations of Bulletproofs, set membership proofs and signature based range proofs written in Golang (Go) are all available on Github [11], implementations of the VAHSS construction 1, written in both python and C++, is available at Github [15] [14]. 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 6 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.

Lets 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 [16], 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 in three sub sections.

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 showed that Bulletproofs were significantly faster, hence more suitable for using in server and client verifiable AHSS presented in Construction 6.

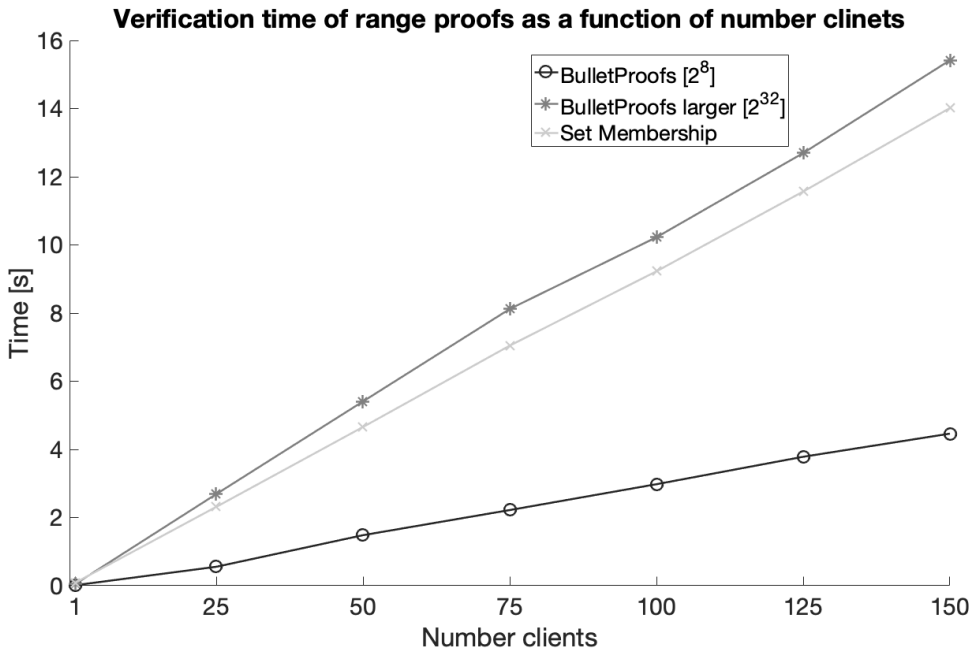
4.2 Runtime

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 6 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.

Table 4.1: Timing in seconds for server and client verifiable-AHSS. Verfication of clients is done by usingBulletProofs

height	**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]

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] 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.

[3] 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.

[4] F. Boudot. Efficient proofs that a committed number lies in an interval. In B. Preneel, editor, *Advances in Cryptology — EUROCRYPT 2000*, pages 431–444, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[5] E. Boyle, N. Gilboa, and Y. Ishai. Function secret sharing: Improvements and extensions. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 1292–1303, New York, NY, USA, 2016. Association for Computing Machinery.

[6] 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.

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

[12] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable se-

cret sharing. In J. Feigenbaum, editor, *Advances in Cryptology — CRYPTO '91*, pages 129–140, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.

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

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

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

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

[17] 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.

[18] 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

Proof of range in signature based range proof

Let $x = \sum_{j=0}^{l-1} x_j u^j$, where x_j is an integer and $x_j \in [0, u)$, u, l are integers and $j \in [0, l-1] (= \mathbb{Z}_l)$. Then it holds that $x \in [0, u^l)$.

$$x = \sum_{j=0}^{l-1} x_j u^j \leq \sum_{j=0}^{l-1} (u-1) u^j = \sum_{j=0}^{l-1} u^{j+1} - \sum_{j=0}^{l-1} u^j = (u-1) \sum_{j=0}^{l-1} u^j =$$
$$(u-1) \frac{u^l - 1}{u-1} = u^l - 1 < u^l$$

Hence the statement is proved and it is trivial to see that if $j \in [0, l]$ the value of x could exceed u^l.

B

LHSS

Questions

- Would it be possible to let x be the solution to $x^{eN} = g^{s_j} h_i g_1^{x_i} g_2^{R_i} = g^{s_j} h_i \pi_i$

Correctness, Securist and Verifiability

- **Correctness** Need to show $\Pr[\textbf{Verify}(vk, f, \sigma, y\pi_1, ..., \pi_n, proof_{RP_1}, ..., proof_{RP_n}) = 1] = 1$. For this it requires to show that all three following holds at the same time
  1. $\tilde{x}^{eN} \stackrel{?}{=} g^s(\prod_{i=1}^n h_i^f) g_1^y$ (*Holds given that* $y = \sum_{i=1}^n x_i$)
  2. $g_1^y \stackrel{?}{=} \prod_{i=1}^n \pi_i$ (*Holds given that* $x_i = \sum_{j=1}^m x_{ij}$ *and* $\pi_i = g_1^{\hat{x}_i} g_2^{r_i}$ *such that* $x_i = \hat{x}_i$)
  3. $\textbf{VerifyRP}(\pi_i, proof_{RP_i}) \stackrel{?}{=} 1 \; \forall i \in \mathcal{N}$ (*Holds given that* $x_i \in [a, b] \; \forall i \in \mathcal{N}$)

  The first statement is proved to hold by original VAHSS-LHS. For the seconds statment it holds that;

$$LHS = g_1^y = g_1^{\sum_{j=1}^m y_j} \stackrel{*}{=} g_1^{\sum_{j=1}^m \sum_{i=1}^n x_{ij}} = g_1^{\sum_{i=1}^n x_i} \; *\text{Due to first statement.}$$

$$RHS = \prod_{i=1}^n \pi_i = \prod_{i=1}^n g_1^{x_i} g_2^{r_i} = g_1^{\sum_{i=1}^n x_i} g_2^{\sum_{i=1}^n r_i} = g_1^{\sum_{i=1}^n x_i}. \implies LHS = RHS$$

  The last follows from the verification of a correctly constructed range proof.
- **Security**
- **Verifiability**
  - If $x_i = \sum_{j=1}^m x_{ij}$, $\pi_i = g_1^{\hat{x}_i} g_2^{r_i}$ and $x_i \neq \hat{x}_i$, then,

$$g_1^y = \prod_{i=1}^n \pi_i \implies g^{\sum_{i=1}^n x_i} = g^{\sum_{i=1}^n \hat{x}_i} \implies$$

$$\sum_{i=1}^n x_i = \sum_{i=1}^n \hat{x}_i \text{ (Contraction)}$$

Construction 7 : 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.

- **SetUp** $1^\lambda, N$ Let N be the product of two safe primes (p, q) each of length $k'/2$. Choose at random two random safe primes $\hat{p}, \hat{q}$ of length $k/2$, such that $\gcd(N, \Phi(\hat{N})) = 1$, where $\hat{N} = \hat{p}\hat{q}$. Then choose $g, g_1, g_2, h_1, ..., h_n$ in $\mathbb{Z}_{\hat{N}}$ at random. Choose an (efficiently computational) injective function $H : \{0,1\}^* \mapsto \{0,1\}^l$, with $l < k'/2$. Define the public verification key $vk = (N, H, \hat{N}, g, g_1, g_2, h_1, ..., h_n)$ and the private signing key to $sk = (\hat{p}, \hat{q})$.
- **ShareSecret** $(1^\lambda, i, x_i) \to (\{x_{ij}\}_{j \in \mathcal{M}})$
  Pick uniformly at random $\{a_i\}_{i \in \{1,..,t\}} \in \mathbb{F}$ and a t-degree, put $x_{ij} = \lambda_{i,j} p_i(\theta_{ij})$. Output $x_{i,j}$ 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** $(sk, vk, fid, x_i, i) \to \sigma_j$
  Parse the verification key vk. Use the injective function H to compute the prime $e = H(fid)$. Let $R_i \in \mathbb{F}$ be the output of a PRF. $R_n, = \phi(N)\lceil \frac{\sum_{i=1}^{n-1} R_i}{\phi(N)} \rceil - \sum_{i=1}^{n-1} R_i$. Choose at random $s_i \in_R \mathbb{Z}_{\hat{N}}$ and use the secret key sk to solve for x in the equation $x^{eN} = g^{sj} \prod_{j=1}^{n} h_j^{f_j^{(i)}} g_1^{x_i + R_i} \bmod(\hat{N})$. Let the vector $f^{(i)}$ be the canonical bases e_i of $\mathbb{Z}^n$, this reduced the equation to $x^{eN} = g^{s_j} h_i g_1^{x_i + R_i} \bmod(\hat{N})$, set $\tilde{x}_i = x$ and output the partial proof $\sigma_i = (e, s_i, fid, \tilde{x}_i)$.
- **ConstructRangeProof** $x_i, i \mapsto proof_{RP}$
  Let $r_i \in \mathbb{F}$ be the output of a PRF. $r_n, = \phi(N)\lceil \frac{\sum_{i=1}^{n-1} r_i}{\phi(N)} \rceil - \sum_{i=1}^{n-1} r_i$ and compute the Fujisaki-Okamoto commitment $\pi_i = g_1^{x_i} g_2^{r_i}$. Use the commitment π_i to construct a square-based range proof for the secret x_i denoted $proof_{RP}$. Output $(\pi_i, proof_{RP_i})$.
- **FinalEval** $(\{y_j\}_{j \in \mathcal{M}}) \to y$
  Compute and output $y = \sum_{j=1}^{m} y_j$.
- **FinalProof** $(vk, \sigma_1, ..., \sigma_n) \to \sigma$
  Parse the partial proofs σ_i to get $(e, s_i, fid, \tilde{x}_i, \pi_i)$. Let $\hat{f} = (\alpha_1, ..., \alpha_n) \in \mathbb{Z}^n$ and define $f' = (\sum_{i=1}^{n} \alpha_i f(i) - f)/(eN)$, where $f = \sum_{i=1}^{n} \alpha f^{(i)} \bmod eN$. Set $s = \sum_{i=1}^{n} \alpha_i s_i \bmod eN$, $s' = (\sum_{i=1}^{n} \alpha_i s_i - s)/eN \bmod eN$ and $\tilde{x} = \frac{\prod_{i=1}^{n} \tilde{x}_i^{\alpha_i}}{g^{s'} \prod_{j=1}^{n} h_j^{f'}} \bmod \hat{N}$.
  Let $\alpha_i = 1 \, \forall i \in \{1, ..., n\}$, then compute $\tilde{x} = \frac{\prod_{i=1}^{n} \tilde{x}_i}{g^{s'} \prod_{j=1}^{n} h_j^{f'}} \bmod \hat{N}$.
  Output the final proof $\sigma = (e, s, fid, \tilde{x})$.
- **Verify** $(vk, f, \sigma, y, \pi_i, ..., \pi_n, proof_{RP_1}, ..., proof_{RP_n}) \to \{0, 1\}$
  Compute $e = H(fid)$. Check that $y, s \in \mathbb{Z}_{\hat{N}}$ and $\tilde{x}^{eN} = g^s \prod_{j=1}^{n} h_j^f g_1^y$. Further check that $g_1^y = \prod_{i=1}^{n} \pi_i$ and $\mathbf{VerifyRP}(\pi_i, proof_{RP_i}) \, \forall i \in \mathcal{N}$