DS-BS: A Generic Framework for Transforming Digital Signatures to Blind Signatures

Dylan Bell

Supervised by: Dr. Essam Ghadafi

School of Computing Science, Newcastle University, UK

Abstract

This dissertation focuses on providing a generic framework for transforming a family of digital signatures known as Linear-Message Strongly Structure-Preserving Signatures into blind signatures. It will outline the steps required for the transformation, in the most generic format possible, providing a security proof to go along with the framework. Meaning that anyone who transforms their schemes doesn't have to prove their new blind signature scheme. An example instantiation is given and explained, as well as showing that the obtained blind signature scheme is secure by using the framework's security proof.

Keywords: Digital Signatures, Blind Signatures, Generic Transformation Framework

1 Introduction

In today's day and age, electronic communications and transactions have become integral parts of everyone's daily life. We all rely on social media to communicate with friends and family, and use credit cards to pay for services. Consequently, the need for secure authentication methods has never been more important. Digital signatures and blind signatures are cryptographic techniques that play a vital role in providing this need to users.

Digital signatures are a fundamental cryptographic primitive that guarantees documents' authenticity, integrity and non-repudiation through a signing protocol. The signer receives a message and produces a corresponding signature, the sender or any third party can verify that the message is authentic by using the message, a public verification key and a signature. It has various applications, such as producing an electronic version of a handwritten signature or being used to build a more complex primitive, one of which is a Blind Signature. Linear-Message Strongly Partially Structure-Preserving Signatures (LmSPSPS) are a type of digital signature

[1] Email: d.bell11@ncl.ac.uk

scheme outlined in a recent paper by Ghadafi [1] that are well studied and build upon previously secure schemes, they require that the polynomials in the numerator be linear in the message to be signed and can be applied to a vector of messages. Making them extremely versatile in possible ways they can be implemented.

Blind signatures are very similar to digital signatures, except instead of sending the message to the signer, a blinded version of the message is sent, known as a commitment. Ensuring that the signer never sees the message they are signing, and the user still obtains a signature on the message. The user obtains a signature on the commitment and unblinds it to reveal the signature on the original message. They were first introduced by Chaum [2] for untraceable payments in e-cash systems, which was created by using the RSA scheme [3] and introducing blinding to it.

1.1 Motivation

Blind signatures preserve privacy and mitigate any potential power imbalance in the signing process. For any scenario where sensitive information is being signed, a blind signature would be better than a normal digital signature scheme.

In digital payment systems, it is important that a user can spend currency without disclosing their identities or any sensitive information pertaining to the transaction as this information can reveal a lot about the individual's lifestyle, Especially in situations such as paying for transportation or hospitality services, blind signatures ensure that none of this information is known by anyone other than the individual.

In electronic voting systems, it's important that a user can cast their vote and keep their identity and choice confidential so that their rights to anonymity are protected. Blind signatures ensure this happens while still being able to verify that the vote is valid, and would stop fake votes being cast.

Despite all the added benefits, blind signatures have when compared to traditional digital signatures, systems that already implement digital signature schemes wouldn't want to go through the process of having to change their infrastructure to implement a new blind signature scheme, due to the cost involved. This is why a framework for transforming a digital signature scheme to its blind counterpart would make blind signatures a lot more accessible, as these systems would only have to make simple changes to their pre-existing scheme.

LmSPSPS schemes are proven to be secure and efficient, which is why this dissertation will focus on transforming this family of signature schemes. The blind signature that would be obtained from this scheme would have the exact same properties as the original signature scheme, but with the added benefits that come from blind signatures.

1.2 Rationale

This project aims to develop a generic framework to transform any digital signature scheme based on a Linear-Message Strongly Partially Structure-Preserving Signature scheme into provably secure a blind signature scheme. We will review existing blind signature schemes and abstract the way blindness is used so that we can successfully create a new blind signature scheme, then provide a security proof for

this transformed blind signature scheme. The security proof will mean that the individual transforming their scheme does not have to prove the security of the obtained blind signature themselves, providing a motivation of sorts to transform their scheme.

1.3 Aim and Objectives

1.3.1 Aim

To produce a generic framework to transform any Linear-Message Strongly Partially Structure-Preserving Signature (LmSPSPS) scheme into a Blind signature scheme, and evaluate its security.

1.3.2 Objectives

(i) Understand how LmSPSPS schemes work, providing formal definitions for the family of signatures.

(ii) Review and understand existing Blind Signature schemes based on bilinear groups.

(iii) Generalise the way blindness is used in these existing schemes.

(iv) Produce a generic transformation framework to transform an LmSPSPS scheme to its blind counterpart.

(v) Prove the security of the obtained Blind Signature.

(vi) Produce an example instantiation of an LmSPSPS scheme to its blind counterpart.

(vii) Discuss the differences between the original LmSPSPS scheme and the example instantiation.

1.4 Structure of Thesis

(i) Introduction - An overview of digital and blind signatures, the motivation behind transforming the former to the latter, the aim and objectives of the project and the structure of the project.

(ii) Background and Related Work - All the relevant research undertaken in this project will be stated here. Explaining the preliminary concepts that are needed to understand the transformation and the related work section that will cover papers that are important to this project.

(iii) Methodology - An overview of the steps to be taken to develop the transformation framework.

(iv) The Generic Transformation Framework - This is the section where the transformation is situated and explained.

(v) Security Proof - This is the section where the security proof is situated and thoroughly explained.

(vi) Results and Evaluation - An assessment of the transformation and proof, providing an example instantiation to solidify the security proof.

(vii) Conclusions and Future Work - A summary of the project findings, includ-

ing limitations, and the future work that would make the transformation and security proof better.

2 Background and Related Work

2.1 Preliminaries

In this section, we provide definitions necessary to understanding this dissertation. All are concepts introduced by various other research papers.

2.1.1 Bilinear Groups

A bilinear group is a tuple defined by $\mathcal{P} = (\mathbb{G}, \hat{\mathbb{G}}, \mathbb{T}, G, \hat{G}, e, p)$, where $\mathbb{G}, \hat{\mathbb{G}}$ and $\mathbb{T}$ are groups of prime order p and G and $\hat{G}$ are generators for $\mathbb{G}$ and $\hat{\mathbb{G}}$. The function e represents the bilinear map $e : \mathbb{G} \times \hat{\mathbb{G}} \rightarrow \mathbb{T}$. We only use the Type-3 setting [6] in this project, where $\mathbb{G} \neq \hat{\mathbb{G}}$. We use $\mathcal{BG}$ to denote a call on an algorithm that will output the description of a bilinear group for an input security parameter λ.

2.1.2 Digital Signatures

See the paper [4] for the full definition. A Digital Signature scheme DS that operates over a bilinear group $\mathcal{P} \leftarrow \mathcal{BG}$ in the message space $\mathcal{M}$ has the following algorithms:

- $KeyGen_{DS}(\mathcal{P})$, that outputs a secret/verification key pair (Sk_{DS},Vk_{DS}) for the bilinear group $\mathcal{P}$.
- $Sign_{DS}(Sk_{DS}$, m), that outputs a signature Σ_{DS} on a message $m \in \mathcal{M}$ using the secret key Sk_{DS}.
- $Verify_{DS}(Vk_{DS}, m, \Sigma_{DS})$, that outputs 1 if Σ_{DS} is valid signature on m and 0 if not.

In some cases, the DS scheme can include a *Randomise* function. This function takes the verification key Vk_{DS}, the message m and the signature Σ_{DS} then produces a new signature Σ'_{DS} for this message. This new signature is indistinguishable from the original signature.

A Digital Signature scheme must satisfy the following properties to be considered secure.

Definition 1 (Correctness). A Digital Signature scheme DS that operates over a bilinear group using the generator $\mathcal{BG}$ is correct if for all $i \in \mathbb{N}$:

$$Pr\begin{bmatrix} \mathcal{P} \leftarrow \mathcal{BG}(1^i); KeyGen_{DS}(\mathcal{P}) \rightarrow (Sk_{DS}, Vk_{DS}); \\ m \leftarrow \mathcal{M}; Sign_{DS}(Sk_{DS}, m) \rightarrow \Sigma_{DS} : \\ Verify(Vk_{DS}, m, \Sigma_{DS}) = 1 \end{bmatrix} = 1.$$

Definition 2 (Existential Unforgeability). A Digital Signature scheme DS that operates over a bilinear group using the generator $\mathcal{BG}$ is Existentially Unforgeable against adaptive Chosen-Message Attacks if for all $i \in \mathbb{N}$ and for all Probabilistic Polynomial Time (PPT) adversaries $\mathcal{A}$, the following probability is negligible (in

λ):

$$Pr\begin{bmatrix}\mathcal{P} \leftarrow \mathcal{BG}(1^i); KeyGen_{DS}(\mathcal{P}) \rightarrow (Sk_{DS}, Vk_{DS});\\ (\Sigma^*_{DS}, m^*) \leftarrow \mathcal{A}^{Sign(Sk_{DS},\cdot)}(\mathcal{P}, Vk_{DS}) :\\ Verify(Vk_{DS}, m^*, \Sigma^*_{DS}) = 1 \wedge m^* \notin M\end{bmatrix}.$$

where M is the set of messages queried to the $Sign$ function.

2.1.3 Linear-message Strongly Partially Structure-Preserving Signatures

See paper [4] for the definition.

A Digital signature scheme DS is considered to be a Linear-message Strongly Partially Structure-Preserving Signature (LmSPSPS) scheme if the following requirements are met:

- The verification key consists of $\mathcal{P}$ and source group elements $(\boldsymbol{X}, \boldsymbol{Y}) \in \mathbb{G}^{\mu} \times \hat{\mathbb{G}}^{\mu'}$.
- The message space is $\mathcal{M} := \mathbb{Z}_P^n$ for some $n \geq 1$.
- A signature on a message $\boldsymbol{m} \in \mathcal{M}$ is of the form $\sigma := (\boldsymbol{S}, \tilde{\boldsymbol{T}}) \in \mathbb{G}^{\nu} \times \hat{\mathbb{G}}^{\nu'}$ which is computed by a generic signer by sampling a vector $\boldsymbol{r} \in \mathbb{Z}_p^{n'}$ and computing $S_i := G^{\frac{\alpha_j(\boldsymbol{sk},\boldsymbol{m},\boldsymbol{r})}{\alpha'_i(\boldsymbol{sk},\boldsymbol{m},\boldsymbol{r})}}$ and $T_j := \hat{G}^{\frac{\beta_j(\boldsymbol{sk},\boldsymbol{m},\boldsymbol{r})}{\beta'_j(\boldsymbol{sk},\boldsymbol{m},\boldsymbol{r})}}$ for some multivariate polynomials $\alpha_i, \alpha'_i, \beta_j, \beta'_j \in \mathbb{F}_p[X_1, ..., X_\mu, Y_1, ..., Y_{\mu'}, M_1, ..., M_n, R_1, ..., R_{n'}]$ of total degree bounded by $d(\lambda)$.
- Signature verification involves deciding group membership and evaluating a set of pairing-product equations of the following form:

$$\prod_{i=1}^{v} e(\sigma_i, \prod_{j=1}^{u'} \hat{Vk}_j)^{a_{i,j}(m)} \prod_{i=1}^{v'} e(\prod_{j=1}^{u} Vk_j, \hat{\sigma}_i)^{b_{i,j}(m)}$$

$$\prod_{i=1}^{v} e(\sigma_i, \prod_{j=1}^{v'} \hat{\sigma}_j)^{c_{i,j}(m)} \prod_{i=1}^{u} \prod_{j=1}^{u'} e(Vk_i, \hat{Vk}_j)^{d_{i,j}(m)} = Z_l$$

  Where $a_{ij}, b_{ij}, c_{ij}, d_{ij} \in \mathbb{F}_p[M_1, ..., M_n]$ are multivariate polynomials of total degree bounded by $d'(\lambda)$, whereas $Z_l \in \mathbb{T}$ is a public constant.
- It holds for all $i \in \nu$ and for all $j \in \nu'$, the polynomials α'_i and β'_j are independent of the message.
- It holds for all $i \in \nu$ and for all $j \in \nu'$, α_i and β_j are linear in $\boldsymbol{M}$. For all $k \in [n]$, for all $i \in [\nu]$, for all $j \in [\nu']$, the degree of M_k in α_i and β_j is either 0 or 1 and for all $\eta, \eta' \in [n]$ neither of the polynomials contain the monomial $M_\eta, M_{\eta'}$.

2.1.4 Blind Signatures

See the papers [1] and [2] for full definitions. For the sake of this project, we use the definition from section 4 in [1].

A Blind signature scheme BS that operates over a bilinear group $\mathcal{P} \leftarrow \mathcal{BG}$ in the message space $\mathcal{M}$ has the following algorithms:

- $KeyGen_{BS}(\mathcal{P})$ that outputs a secret/verification key pair (Sk_{BS}, Vk_{BS}) for the bilinear group $\mathcal{P}$.
- $Request^0_{BS}(Vk_{BS}, m)$ that takes a message m, where $m \in \mathcal{M}$, from the user and the verification key and produces a signature request ρ and some state st.
- $Issue_{BS}(Sk_{BS}, \rho)$ that takes the signature request ρ and the secret key and produces a pre-signature β.
- $Request^1_{BS}(Vk_{BS}, \beta, st)$ that takes the verification key, the pre-signature β and the state st. It produces a blind signature Σ_{BS} on m by unblinding the pre-signature, or outputs $\perp$ if the pre-signature is invalid.
- $Verify_{BS}(Vk_{BS}, m, \Sigma_{BS})$ that outputs 1 if Σ_{BS} is a valid signature on the message m, or 0 if not.

A Blind Signature scheme must satisfy the following properties to be considered secure.

Correctness: Correctness of blind signatures requires that for all $\lambda \in \mathbb{N}$ and all $m \in \mathcal{M}$, we have

$$Pr\begin{bmatrix}\mathcal{P} \leftarrow \mathcal{BG}(1^\lambda); (Sk_{BS}, Vk_{BS}) \leftarrow KeyGen_{BS}(\mathcal{P}); \\ (\rho, st) \leftarrow Request^0_{BS}(Vk_{BS}, m); \beta \leftarrow Issue_{BS}(Sk_{BS}, \rho); \\ \Sigma_{BS} \leftarrow Request^1_{BS}(Vk_{BS}, \beta, st) : Verify(Vk_{BS}, m, \Sigma_{BS}) = 1\end{bmatrix} = 1.$$

Unforgeability: Unforgeability requires that it is infeasible for an adversarial user who interacts with an honest signer k times to output $k+1$ valid signatures on $k+1$ distinct messages.

Blindness: Blindness requires that an adversarial signer who freely chooses two messages m_0 and m_1 as well as the keys and then takes part in interactions with an honest user to generate signatures on those messages cannot tell the order in which the messages were signed.

2.1.5 Pedersen's Commitment Scheme

See the paper [5] for the full definition.

This commitment scheme has 3 functions. The *setup* function creates a commitment key Ck to use for blinding. The *Commit* function uses the commitment key, an input message m and randomness r, where $m, r \in \mathbb{Z}_p$ and returns the commitment Co. The *open* function opens the commitment value Co using the message m and randomness r used in the *Commit* function.

2.1.6 Notation

This section details some notation used in this dissertation for clarification purposes.

- We use the notation $A^\times$ to denote the group $A \in \{\mathbb{G}, \hat{\mathbb{G}}, \mathbb{T}, \mathbb{Z}_p\}$, such that $A^\times := A \backslash 1_A$.

2.2 Related Work

E. Ghadafi defines the LmSPSPS Digital signature scheme that we will be transforming in his paper [4]. Providing a general overview of what the scheme should entail, giving a new version of the scheme and providing a security proof alongside it. Due to the concise notation and rigorous proof, we chose to transform this family of digital signatures. In this paper, they also produce a framework for transforming an LMSPSPS scheme into a Structure-Preserving scheme. This provides great insights into how a generic framework should be structured and gives examples of how to break down the scheme to allow it to be transformed.

Another work authored by E. Ghadafi [1] defines a blind signature scheme based on an LmSPSPS scheme. They create two blind signature constructions, one which uses a fraction in the signature polynomial, and the other has no denominator value. This gives insights on how a blind signature utilises blindness to hide the message from the signer, as it can be applied to any form of signature, no matter how different the schemes are. They also construct two partially blind constructions, where there is a small amount of information that is allowed to be shown to the signer. All constructions come with complete and comprehensive security proofs using the Generic Group Model [7]. This makes it extremely useful for this project's purpose, as it gives concrete examples as to how a blind signature proof in the Generic Group Model should be structured and explained.

The Generic Group Model was first introduced in [7]. It is a mathematical framework to analyse and prove the security of cryptographic protocols. It is defined by a set of mathematical operations which mainly use group multiplication and inversion. It allows cryptographers to formulate proofs about their protocols' security properties without going into specific details. This is relevant to this project as it has been used to prove many digital signature schemes since its introduction, one being the blind signature scheme in [1].

T. P. Pedersen authored the "Pedersen's Commitment Scheme" in [5]. This paper covers the development of a new non-interactive secure verifiable secret sharing scheme, built on top of various other schemes such as the one in [8], which introduced the first non-interactive verifiable secret sharing scheme. The concept in this paper is relevant to blindness, as the protocol details how to use a commitment value instead of a plain-text message, and how to retrieve the original message from the commitment value. The blind signature scheme in [1] uses a very generalised variant of this commitment scheme, so in unison, these papers have provided a great basis for understanding how a digital signature can use a commitment scheme to form the basis of a blind signature scheme.

3 Methodology

This section details the steps that will be taken to create the framework so it meets the aim and objectives outlined in section 1.3. Providing an overview of the process and explaining the choices made. Detailing a rough design of the framework.

3.1 High-Level Overview

A high-level view of the process used to develop the framework in shown below in Figure 1. Splitting the process into the 5 main tasks that must be completed to fully develop the framework.

Fig. 1. High-Level Project Diagram

To be able to produce the framework, understanding existing signature constructions is vital so that the obtained blind signature is syntactically correct. As mentioned in the related work section, Ghadafi produced two blind signature constructions in his paper [1], which he outlines in section 5. We will review these constructions and pick out the key information that differentiates them from the original LmSPSPS scheme, abstracting how blindness is used to provide a basis for our framework. Then produce a framework design so that we can efficiently structure the transformation while maintaining readability.

3.2 Abstracting Blindness

3.2.1 How the constructions use blindness

This section will detail the key information extracted from the blind signature constructions and abstract how blindness is used in the constructions.

The main difference between the blind signature constructions and the original LmSPSPS scheme is the function layout. As explained in section 2.1.4, blind signatures operate using a back-and-forth communication between the user and the signer, rather than the traditional sign function that digital signatures use. These constructions utilise Pedersen's commitment scheme, introduced in [5] and briefly outlined in section 2.1.5 of this paper. They integrate the elements of the Commitment key into the verification key and the secret key instead of keeping a separate key.

The only change to the key generation function is adding a new element h to the secret key, and using this to generate 2 new verification key elements H and $\tilde{H}$, H is generated as G^h in both constructions. However, $\hat{H}$ is generated differently in each construction, the first generates it as $\hat{G}^h$, and the second generates it as $\hat{G}^{\frac{1}{h}}$. Both provide valid components for blinding, the choice is dependent on the structure of the signature, as both have slightly different signature structures.

The single Sign function is changed into a 3-function interactive protocol, which is where Pedersen's commitment scheme is integrated. It is changed into $Request^0_{BS}$, $Issue_{BS}$ and $Request^1_{BS}$, which are explained in a general way in section 2.1.4. They both generate the commitment value as $Co := G^m H^r$, using H^r as the blinding factor and send this to the signer.

Then, the signer generates the pre-signature $\beta := (A', B', C')$ by signing the commitment. A' is a signature component independent of the commitment, B' contains the signature on the commitment and C' is the unblinding element, which the user will use to reveal the signature on the original message. All pre-signature elements use some randomness value in both constructions, however, the structure of the signature elements is different (other than A').

The user receives the pre-signature tuple $\beta := (A', B', C')$ from the signer, they check that the unblinding element C' is valid, then compute $B' := B'C'^{-r}$ to unblind the pre-signature. They then check the obtained signature is valid, if so they randomise the signature and the final signature is generated.

Verification is performed in the same way as the original signature scheme, as no blindness remains in the signature after the unblinding process.

3.2.2 Generalising blindness

This section will use the above to generalise the concept of blindness from the constructions, enabling it to be easily implemented into the framework.

The first part of blindness is generating the mechanism to blind the message. The signer creates the necessary components, H and $\hat{H}$ in the key generation function. Then uses H to create a blinding factor by raising it to a randomness value r.

The next step is blinding the message, this is done by replacing the message you would send with the commitment value. The commitment value Co is the group generator raised to the message G^m and multiplied by the blinding factor H^r.

To generate the pre-signature, the signer will replace all instances where the message would appear in the original signature, with Co. So, wherever G^m appears in the signature, Co would replace it. In the case that a signature component contains the message, an unblinding component will be provided alongside the pre-signature. If the signature component is independent of the message, then it is computed exactly the same as would happen in the original digital signature scheme.

To obtain the final signature on the message m, the user will combine the unblinding factor and the randomness used to generate the commitment and multiply this with the blinded signature component. In the case of the construction, they compute $B' := B'C'^{-r}$ to unblind the pre-signature component.

3.3 High-Level Transformation Framework Design

This section will provide the design process for the generic transformation framework.

We will split the framework up into the blind signature functions, and detail how each function will be derived from the LmSPSPS scheme. Using the generalised concept of blindness detailed above, as well as the definition of blind signatures in section 2.1.4, we start to build the framework for the transformation.

3.3.1 Key Generation:

Transformation from $KeyGen_{DS}(\mathcal{P})$ to $KeyGen_{BS}(\mathcal{P})$.

- $KeyGen_{BS}(\mathcal{P})$:
  - Run $KeyGen_{DS}$ as normal to obtain the digital signature key pair (Sk_{DS}, Vk_{DS}).

- Generate a random value h, add this to the secret key to create Sk_{BS}.
- Generate H and $\hat{H}$ by raising the group element to the random value h.. add both of these to the verification key to create Vk_{BS}.
- Return the new secret key and verification key pair (Sk_{BS}, Vk_{BS}).

3.3.2 Signature Creation:

The transformation from the single $SignDS$ function into 3 functions based on Pedersen's commitment scheme: $Request^0_{BS}, Issue_{BS}$ and $Request^1_{BS}$.

- $Request^0_{BS}(Vk_{BS}, m)$:
  - Generate commitment randomness r and save the state st as $st = (m, r)$
  - Parse the original digital signature Σ_{DS} to see all components.
  - If there exists a component that is dependent on the message, generate the commitment value as $G^m H^r$.
  - Return $\rho := Co$ to the signer.
- $Issue_{BS}(Sk_{BS}, \rho)$:
  - Define the pre-signature array β to store the pre-signature components.
  - Generate a new randomness value $\boldsymbol{l}$, to be used for the pre-signature.
  - Parse the digital signature as an array of components, σ_1 to σ_n. So that we can operate over the signature components individually.
  - Break down each signature component into it's linear combinations, defining a set of indices I where the message is present, and some set $\hat{I}$ where the message isn't present.
  - Compute the new signature components where the message is present, using the set I and replacing the instances where m would appear with the Commitment value Co.
  - Compute the new signature components where the message isn't present, using the set $\hat{I}$.
  - Generate an unblinding component for all messages where m is present, using the exponent of the commitment value Co.
  - Add all of these computed components to the pre-signature β, and return it to the user.
- $Request^1_{BS}(Vk_{BS}, \beta, st)$:
  - Define the signature array Σ_{BS} to store the final signature components.
  - Parse the pre-signature array into components.
  - If the pre-signature contains the Commitment value, then we check that the pre-signature is valid.
  - Unblind the signature using the unblinding component, and check that it's valid using the verification equation.
  - Add all of the components into the final signature array.
  - Use the Randomise function to randomise the final signature array to obtain the final signature.

3.3.3 Verification:

Verification is executed exactly the same as the LmSPSPS scheme used, through group membership checks. See section 2.1.3 for the equation format.

4 The Generic Transformation Framework

This section details the final generic transformation framework to transform a Lm-SPSPS scheme into its blind counterpart. It covers all possibilities of signatures, and how to alter them to obtain the new scheme, using the rough design outlined in the previous section.

4.1 Steps

4.1.1 Key Generation:

Transformation from $KeyGen_{DS}(\mathcal{P})$ to $KeyGen_{BS}(\mathcal{P})$.

- $KeyGen_{BS}(\mathcal{P})$:
  - Run $KeyGen_{DS}(\mathcal{P}) \rightarrow (Sk_{DS} = (Sk_1, ..., Sk_\delta), Vk_{DS} = ((Vk_1, ..., Vk_{\delta'}), (\hat{Vk}_1, ..., \hat{Vk}_{\delta''}))$.
  - Select h $\leftarrow \mathbb{Z}_p^\times$, append to Sk_{DS} to generate Sk_{BS}.
  - Compute $(H, \hat{H}) := (G^h, \hat{G}^t)$, where t is either h or $\frac{1}{h}$. Append H and $\hat{H}$ to Vk_{DS} to generate Vk_{BS}.
  - Return $(Sk_{BS} := (Sk_{DS}, h), Vk_{BS} := (Vk_{DS}, H, \hat{H}))$.

4.1.2 Signature Creation:

Transformation from the single $Sign_{DS}$ function into 3 functions based on Pedersen's Commitment Scheme: $Request^0_{BS}, Issue_{BS}$ and $Request^1_{BS}$.

- $Request^0_{BS}(Vk_{BS}, m)$:
  - Select r $\leftarrow \mathbb{Z}_p^\times$.
  - Save current state as $st = (m, r)$.
  - Parse Σ_{DS} as $([\sigma_1, ..., \sigma_{n'}], [\hat{\sigma}_1, ..., \hat{\sigma}_{n''}])$.
  - If $\exists\ \sigma_i \in \Sigma_{DS}$, where σ_i is dependent on m:
    - Generate commitment value $Co := G^m \cdot H^r$.
    - Add Co to ρ.
  - If $\exists\ \hat{\sigma_j} \in \Sigma_{DS}$, where $\hat{\sigma_j}$ is dependent on m:
    - Generate commitment value $\hat{Co} := \hat{G}^m \cdot \hat{H}^r$.
    - Add $\hat{Co}$ to ρ.
  - Return ρ to signer.

- $Issue_{BS}(Sk_{BS}, \rho)$:
  - Define pre-signature $\beta = (\boldsymbol{\beta}, \hat{\boldsymbol{\beta}})$ to store computed pre-signature components.
  - Generate new random exponent $\boldsymbol{l} \leftarrow \mathbb{Z}_p^n$.
  - Parse Σ_{DS} as $([\sigma_1, ..., \sigma_{n'}], [\hat{\sigma}_1, ..., \hat{\sigma}_{n''}])$.
  - For each signature component $\sigma_i \in \mathbb{G}$:
    - Define corresponding pre-signature component β_i.
    - Parse σ_i as:

$$G^{\frac{\sum_{i=1}^{q}(c_{ij}Sk_il_j)+(c'_{ij}\prod_i(Sk_i)(\alpha_{ij}+\alpha'_{ij}l_j)\cdot(c''_{ij}+k_{ij}m)}{\sum_i(b_{ij}Sk_il_j)+(b'_{ij}\prod_i(Sk_i)(\alpha_{ij}+\alpha'_{ij}l_j)}}$$

    - Define the set $\hat{I} \subseteq [q]$ as the subset of indices i where $k_{ij} \neq 0$ and let

$\breve{I} = [q] \backslash \hat{I}$.

· Define A' as:

$$(G^{\sum_{i \in \breve{I}} (c_{ij} Sk_i l_j) + (c'_{ij} \prod_i (Sk_i)(\alpha_{ij} + \alpha'_{ij} l_j)} \prod_{i \in \hat{I}} Co^{k_{ij}((c_{ij} Sk_i l_j) + (c'_{ij} \prod_i (Sk_i)(\alpha_{ij} + \alpha'_{ij} l_j))})^{\frac{1}{V}}$$

where V=

$$\sum_i (b_{ij} Sk_i l_j) + (b'_{ij} \prod_i (Sk_i)(\alpha_{ij} + \alpha'_{ij} l_j)$$

· If $|\hat{I}| > 0$:

- Parse A' as $(G^U Co^W)^{\frac{1}{V}}$.
- Define $B' := H^{\frac{W}{V}}$.

· Otherwise:

- Define $B' := \perp$.

· Set $\beta_i := (A', B')$.

· Append β_i to $\boldsymbol{\beta}$.

- For each signature component $\hat{\sigma}_j \in \hat{\mathbb{G}}$:
  · Define corresponding pre-signature component $\hat{\beta}_j$.
  · Parse $\hat{\sigma}_j$ as:

$$\hat{G}^{\frac{\sum_{i=1}^{q} (c_{ij} Sk_i l_j) + (c'_{ij} \prod_i (Sk_i)(\alpha_{ij} + \alpha'_{ij} l_j) \cdot (c''_{ij} + k_{ij} m)}{\sum_i (b_{ij} Sk_i l_j) + (b'_{ij} \prod_i (Sk_i)(\alpha_{ij} + \alpha'_{ij} l_j)}}$$

  · Define the set $\hat{I} \subseteq [n]$ as the subset of indices i where $k_{ij} \neq 0$ and let $\breve{I} = [n] \backslash \hat{I}$.
  · Define A' as:

$$(\hat{G}^{\sum_{i \in \breve{I}} (c_{ij} Sk_i l_j) + (c'_{ij} \prod_i (Sk_i)(\alpha_{ij} + \alpha'_{ij} l_j)} \prod_{i \in \hat{I}} \hat{Co}^{k_{ij}((c_{ij} Sk_i l_j) + (c'_{ij} \prod_i (Sk_i)(\alpha_{ij} + \alpha'_{ij} l_j))})^{\frac{1}{V}}$$

  where V=

$$\sum_i (b_{ij} Sk_i l_j) + (b'_{ij} \prod_i (Sk_i)(\alpha_{ij} + \alpha'_{ij} l_j)$$

  · If $|\hat{I}| > 0$:

  - Parse A' as $(\hat{G}^U \hat{Co}^W)^{\frac{1}{V}}$.
  - Define $B' = \hat{H}^{\frac{W}{V}}$.

  · Otherwise:

  - Define $B' := \perp$.

  · Set $\hat{\beta}_j := (A', B')$

  · Append $\hat{\beta}_j$ to $\hat{\boldsymbol{\beta}}$.
- Return $\beta = (\boldsymbol{\beta} = (\beta_1, ..., \beta_{n'}), \hat{\boldsymbol{\beta}} = (\hat{\beta}_1, ..., \hat{\beta}_{n''}))$ to user.

- $Request^1_{BS}(Vk_{BS}, \beta, st)$:
  - Define signature $\Sigma_{BS} = (\boldsymbol{\sigma}, \hat{\boldsymbol{\sigma}})$.
  - For each pre-signature component $\beta_i \in \boldsymbol{\beta}$:
    · Parse β_i as (A', B').

· If $B' \neq \perp$:
- Check pre-signature validity by computing $e(A', \hat{G}) = e(B', \prod_{j=1}^{\delta'} Vk_j)$, if invalid return $\perp$.
- Unblind signature by computing $A' := A' \cdot B'^{-r}$.
- Use verification product-pairing equation to check signature on m is valid, if not return $\perp$.

· Set $\sigma_i := A'$.
· Add σ_i to $\boldsymbol{\sigma}$.

• For each pre-signature component $\hat{\beta}_j \in \hat{\boldsymbol{\beta}}$:
· If $B' \neq \perp$:
- Check pre-signature validity by computing $e(A', G) = e(B', \prod_{j=1}^{\delta''} \hat{Vk_j})$, if invalid return $\perp$.
- Unblind signature by computing $A' := A' \cdot B'^{-r}$.
- Use verification product-pairing equation to check signature on m is valid, if not return $\perp$.

· Set $\hat{\sigma}_j := A'$.
· Append $\hat{\sigma}_j$ to $\hat{\boldsymbol{\sigma}}$.

• Return $\Sigma_{BS} \leftarrow Randomise_{DS}((\boldsymbol{\sigma} = (\sigma_1, ..., \sigma_{n'}), \hat{\boldsymbol{\sigma}} = (\hat{\sigma}_1, ..., \hat{\sigma}_{n''})))$

4.1.3 Verification:

There is no need to transform $Verify_{DS}(Vk_{DS}, m, \Sigma_{DS})$ to $Verify_{BS}(Vk_{BS}, m, \Sigma_{BS})$ as the verification will be the same as the original DS scheme. For consistency, below is the outline of $Verify_{BS}$.

- $Verify_{BS}(Vk_{BS}, m, \Sigma_{BS})$:
  - Parse Σ_{BS} as $(\boldsymbol{\sigma}, \hat{\boldsymbol{\sigma}})$.
  - Verification of group membership will take the following format:

$$\prod_{i=1}^{v} e(\sigma_i, \prod_{j=1}^{u'} \hat{Vk_j})^{a_{i,j}(m)} \prod_{i=1}^{v'} e(\prod_{j=1}^{u} Vk_j, \hat{\sigma}_i)^{b_{i,j}(m)}$$

$$\prod_{i=1}^{v} e(\sigma_i, \prod_{j=1}^{v'} \hat{\sigma}_j)^{c_{i,j}(m)} \prod_{i=1}^{u} \prod_{j=1}^{u'} e(Vk_i, \hat{Vk_j})^{d_{i,j}(m)} = Z_l$$

  - If valid, return 1. Otherwise, return 0.

5 Security Proof

5.1 DS Assumption

To build the security proof for the new Blind Signature, the first step is to define an assumption for the original Digital Signature scheme DS. Since the DS corresponds to a LmSPSPS scheme that is proven to be unforgeable, there has to exist an assumption and subsequent proof for it. The general form of this will be the following:

Definition 1 (DS Assumption). Let $\mathcal{P} = (\mathbb{G}, \hat{\mathbb{G}}, \mathbb{T}, G, \hat{G}, e, p)$ be the description of a type-3 bilinear group generated by $\mathcal{BG}(1^\lambda)$. Let $Sk_{DS} = (Sk_1, ..., Sk_\delta) \leftarrow$

$\mathbb{Z}_p^{\delta}$ and let $Vk_{DS} = ((G_1^{\gamma(Sk_{DS})}, ..., G_{\delta'}^{\tau(Sk_{DS})}), (\hat{G}_1^{\gamma(Sk_{DS})}, ..., \hat{G}_{\delta''}^{\tau(Sk_{DS})}))$. Let $\mathcal{ODS}_{Vk_{DS}}(\cdot)$ be an oracle that when queried on some $m \in \mathbb{Z}_p$, signs the message and returns a signature $\Sigma_{DS} = (\boldsymbol{\sigma}, \hat{\boldsymbol{\sigma}})$, where $\boldsymbol{\sigma} \in \mathbb{G}^{n'}$ and $\hat{\boldsymbol{\sigma}} \in \hat{\mathbb{G}}^{n''}$.

The DS assumption will hold (relative to $\mathcal{BG}$) if for all Probabilistic Polynomial Time (PPT) adversaries $\mathcal{A}$ given $(\mathcal{P}, Vk_{DS})$ and unlimited access to the oracle $\mathcal{ODS}_{Vk_{DS}}(\cdot)$, the probability that $\mathcal{A}$ can output a new signature Σ_{DS}^* for a message $m^* \in \mathbb{Z}_p$ that has not been queried to the oracle $\mathcal{ODS}_{Vk_{DS}}(\cdot)$ is negligible (in λ).

5.2 BS Assumption

Due to the new blind signature scheme not revealing the content of the message to the signer, this assumption will be a one-more variant of the DS assumption above. We will be using the BSOMI assumption outlined by Ghadafi in [1], but applying it on a case with a verification key with more than 2 elements. Here the adversary will have to produce one more signature-message pair than they queried for.

For this assumption, we are going to use the case where a unilateral digital signature is being transformed, where all signature components are in $\mathbb{G}$ and all verification key elements, other than H, are in $\hat{\mathbb{G}}$. The signature contains 2 components, one that is dependent on the message ($\tilde{\sigma}$) and the other not dependent on the message (σ), this encapsulates all of the signature possibilities for the group. The final limitation is that we don't allow for signature polynomials that contain a negative exponent, the only part of the entire scheme that contains a negative exponent is $\hat{H}$

Definition 2 (BS Assumption). Let $\mathcal{P} = (\mathbb{G}, \hat{\mathbb{G}}, \mathbb{T}, G, \hat{G}, e, p)$ be the description of a type-3 bilinear group generated by $\mathcal{BG}(1^{\lambda})$. Let $Sk_{BS} = (Sk_{DS}, h)$, where $h \leftarrow \mathbb{Z}_p$, and let $Vk_{BS} = (Vk_{DS}, H, \hat{H})$, where $(H, \hat{H}) = (G^h, \hat{G}^{\frac{1}{h}})$. Let $\mathcal{OBS}_{Vk_{BS}}(\cdot)$ be an oracle that on receiving some input $M \in \mathbb{G}$, where $M = G^m$ for some $m \in \mathbb{Z}_p$, returns the tuple $(\tilde{\sigma} = G^{\beta(Sk_{DS}, \boldsymbol{l})}, A = G^{\beta(Sk_{DS}, l)} \cdot M^{\alpha(Sk_{DS}, l)}, B = H^{\alpha(Sk_{DS}, l)})$, where $l \leftarrow \mathbb{Z}_p$ and (A, B) is the pre-signature to form $\tilde{\sigma}$.

We say that the BS assumption holds for all PPT adversaries $\mathcal{A}$, the following advantage is negligible (in λ):

$$Pr\left[\begin{array}{l} \mathcal{P} \leftarrow \mathcal{BG}(1^{\lambda}); Sk_{BS} = (Sk_{DS}, h); Vk_{BS} = (Vk_{DS}, H = G^h, \hat{H} = \hat{G}^{\frac{1}{h}}); \\ \{(A, \sigma)_i, m_i\}_{i=1}^{k+1} \leftarrow \mathcal{A}^{\mathcal{OBS}^k_{Vk_{BS}}(\cdot)}(\mathcal{P}, Vk_{BS}) : \\ |\{m_i\}_{i=1}^{k+1}| = k+1 \wedge \forall i \in [k+1] : \sigma \neq 1_{\mathbb{G}} \wedge e(A, \hat{G}) = e(\sigma, \prod_{j=2}^{\delta''} Vk_1 Vk_j^{c_j m_i^{c'_x}}) \end{array}\right]$$

We now show that the BS assumption is intractable in the generic group model [7].

Theorem 1. For any generic adversary $\mathcal{A}$ against the BS assumption, if p is the prime order of the bilinear group and $\mathcal{A}$ makes $q_{\mathcal{O}} \leq k$ queries to the BS oracle $\mathcal{OBS}^k_{Vk_{BS}}$, q_G group operation queries and q_P pairing queries then $\mathcal{A}$ is unable to produce $q_{\mathcal{O}} + 1$ signature-message tuples (σ, A, m) and the probability of $\mathcal{A}$ against the BS assumption is $\mathcal{O}(\frac{q_{\mathbb{G}}^2 q_{\mathcal{O}} + q_P^2 q_{\mathcal{O}} + q_{\mathcal{O}}^3}{p})$.

Proof. $\mathcal{A}$ interacts with the BS oracle through group handles, κ, which simulates a message input. The challenger keeps 3 lists $L_{\mathbb{G}}, L_{\hat{\mathbb{G}}}$ and $L_{\mathbb{T}}$ of pairs (κ, P), where κ is the group element of the list raised to some value chosen from a set of a size greater than $3p$, and P is some Laurent polynomial in $\mathbb{Z}_p[\tilde{\sigma}_1, ..., \tilde{\sigma}_{q_{\mathcal{O}}}, Vk_1, ..., Vk_{\delta''}, H^{\pm 1}]$. To update each list, an Update function is used in the format $\kappa_i \leftarrow Update(L_i, P)$. It searches the list for a matching polynomial to P, if a matching polynomial is found then the associated κ value is returned. Otherwise, a new element is chosen from the set to create a κ value to store the polynomial, adding the pair (κ, P) to the list L_i, then returning κ. To make parsing the verification key easier, we define the set $J \subseteq [\delta'']$ as the subset of indices j where $c_j \neq 0$.

To start, the challenger performs $Update(L_{\mathbb{G}}, 1), Update(L_{\mathbb{G}}, H), Update(L_{\hat{\mathbb{G}}}, H^{-1})$ and for all elements Vk_j, where $j \in J$, $Update(L_{\hat{\mathbb{G}}}, Vk_j)$ to initialise the lists. Queries to the BS oracle $\mathcal{O}BS_{Vk_{BS}}$ are handled in the following model:

- **BS Oracle:** $\mathcal{A}$ is able to make $q_{\mathcal{O}}$ queries to the oracle, using some handle κ. For a query κ_i to the oracle, $\mathcal{O}BS_{Vk_{BS}}(\kappa_i)$, the challenger searches $L_{\mathbb{G}}$ for κ_i. If it is not in the list, then $\perp$ is returned. However, if κ exists in $L_{\mathbb{G}}$, this means there is a pair (κ_i, P_i) in the list, the challenger then returns $(\kappa_\sigma, \kappa_A, \kappa_B)$ by computing:

$$\kappa_\sigma \leftarrow Update(L_{\mathbb{G}}, \sigma_i)$$
$$\kappa_A \leftarrow Update(L_{\mathbb{G}}, \sigma_i(\sum_{j \in J} Vk_1 + Vk_j P_i))$$
$$\kappa_B \leftarrow Update(L_{\mathbb{G}}, \sigma_i \sum_{j \in J} HVk_j^{c_j})$$

- **Group Operation Oracles:** $\mathcal{A}$ is able to make q_G queries to these oracles. They have access to the oracles $\mathcal{O}_{\mathbb{G}}, \mathcal{O}_{\hat{\mathbb{G}}}$ and $\mathcal{O}_{\mathbb{T}}$, which correspond to group operation queries in each of the groups $\mathbb{G}, \hat{\mathbb{G}}$ and $\mathbb{T}$. They can query either addition or subtraction operations on 2 κ values, so for a query $\mathcal{O}_i(\kappa_1, \kappa_2, \pm)$ the challenger searches L_i for the pairs (κ_1, P_1) and (κ_2, P_2). If both of the pairs exist, $Update(L_i, P_1 \pm P_2)$ is returned $\mathcal{A}$, if not, $\perp$ is returned.
- **Pairing Oracle:** $\mathcal{A}$ is able to make q_P queries to this oracle. For the query $\mathcal{O}_P(\kappa_1, \kappa_2)$, the challenger searches for the pairs (κ_1, P_1) from $L_{\mathbb{G}}$ and (κ_2, P_2) from $L_{\hat{\mathbb{G}}}$. If both of these pairs exist, then $Update(L_{\mathbb{T}}, P_1 P_2)$ is returned to $\mathcal{A}$. Otherwise, $\perp$ is returned.

Using the above, we can simulate the entire process $\mathcal{A}$ goes through to attempt to succeed.

If successful, $\mathcal{A}$ will produce $q_{\mathcal{O}}+1$ tuples $\{\kappa_\sigma^{(i)}, \kappa_A^{(i)}, m_i\}_{i=1}^{q_{\mathcal{O}}+1}$, where $m_i \in \mathbb{Z}_p$ are distinct. Let $P_\sigma^{(i)}$ and $P_A^{(i)}$ be the polynomials associated with the κ values. Since the tuple must be a solution to the BS problem, for all $i \in [q_{\mathcal{O}} + 1]$ we must have:

$$P_A^{(i)} - P_\sigma^{(i)}(\sum_{j \in J} Vk_1 + Vk_j m_i) \equiv 0 \tag{1}$$
$$P_\sigma^{(i)} \neq 0 \tag{2}$$

To start, we show that for all $i \in [q_{\mathcal{O}}+1]$ that $P_A^{(i)}$ satisfies $deg_{Vk_1}(P_A^{(i)}) = 1$. At the start of the game, $L_{\mathbb{G}}$ contains no polynomial P that satisfies $deg_{Vk_1}(P) \neq 0$.

Therefore, on the first oracle query $\mathcal{OBS}_{Vk_{BS}}(\kappa_1)$, $deg_{Vk_1}(P_1) = 0$, as this will correspond to a P_σ value where there is no Vk_1. After this first query, the only polynomial in $L_{\mathbb{G}}$ with a degree of Vk_1 that isn't 0 would be the first P_A polynomial in the list, so we can use P_{A_1} to denote this. Giving us the equation $P_{A_1} = \sigma_1(Vk_1 + \sum_{j=2}^{\delta''} Vk_j^{c_j} P_1)$, which corresponds to κ_{A_1}. From this we clearly see that $deg_{Vk_1}(P_{A_1}) = 1$. Using this information, we can say that if this κ value is used to query the oracle again, the corresponding polynomial P_{A_i} will satisfy $deg_{Vk_1}(P_{A_i}) = 1$. Meaning that for all polynomials P_{A_j}, where $j > 1$, satisfy $deg_{Vk_1}(P_{A_j}) = 1$. So we can now say that for all $i \in [q_{\mathcal{O}} + 1]$, for the pair $(P_\sigma^{(i)}, P_A^{(i)})$ to satisfy the verification equation, we must have that $deg_{Vk_1}(P_A^{(i)}) = 1$ and $deg_{Vk_1}(P_\sigma^{(i)}) = 0$.

The next step is proving that, for all $i \in [q_{\mathcal{O}} + 1]$ and $j \in J$, $deg_{Vk_j}(P_\sigma^{(i)}) = 0$ is satisfied. After starting, all polynomials P in $L_{\mathbb{G}}$ satisfy $deg_{Vk_j}(P) = 0$, as all of these verification elements belong to $\hat{\mathbb{G}}$ so are in $L_{\hat{\mathbb{G}}}$. The only polynomials in $L_{\mathbb{G}}$ that have a degree where all Vk_j aren't 0 are in responses from the BS oracle, which corresponds to κ_A and κ_B values. None of the oracle queries will result in a polynomial in $L_{\mathbb{G}}$ containing the monomial $HVk_1Vk_j^u$, for any value $j \in J$. Therefore, if for any $i \in [q_{\mathcal{O}} + 1]$, $P_\sigma^{(i)}$ contains the monomial HVk_j^n, for the pair $(P_\sigma^{(i)}, P_A^{(i)})$ to validate correctly it means that the monomial $HVk_1Vk_j^n$ must be in $P_A^{(i)}$, which as we stated above, is impossible. Another case is if for any $i \in [q_{\mathcal{O}} + 1]$ the polynomial $P_\sigma^{(i)}$ contains the monomial Vk_j^n, where $n \neq 0$, it means that $P_A^{(i)}$ must contain a monomial $Vk_1Vk_j^n$ for the pair $(P_\sigma^{(i)}, P_A^{(i)})$ to be valid, which means that $deg_{Vk_1}(P_A^{(i)}) = 2$ must hold, which we earlier established was impossible. Therefore, for all values of $i \in [q_{\mathcal{O}} + 1]$ and $j \in J$, we must have that $P_\sigma^{(i)}$ satisfies $deg_{Vk_j}(P_\sigma^{(i)}) = 0$. So for $(P_\sigma^{(i)}, P_A^{(i)})$ to be a valid pair we must have that $P_A^{(i)}$ satisfies $deg_{Vk_j}(P_A^{(i)}) = 1$. Now we show that $deg_H(P_\sigma^{(i)}) = 0$ for all $i \in [q_{\mathcal{O}} + 1]$. If the polynomial $P_\sigma^{(i)}$ contains a monomial H^n, where $n \neq 0$, for the pair to be valid, it means that $P_A^{(i)}$ must contain a monomial H^nVk_1, which is impossible as none of the BS oracle queries would produce a monomial of this form in $L_{\mathbb{G}}$. Therefore, we have that $deg_H(P_\sigma^{(i)}) = 0$ and $deg_H(P_A^{(i)}) = 0$ for all values of $i \in [q_{\mathcal{O}} + 1]$.

It is clear from the above that for all $i \in [q_{\mathcal{O}} + 1]$ that

$$P_\sigma^{(i)} = \alpha_i + \sum_{n=1}^{q_{\mathcal{O}}} \beta_{in}\sigma_n$$

If for any $i \in [q_{\mathcal{O}} + 1]$ $\alpha_i \neq 0$, this would mean that $P_A^{(i)}$ contains some term $\alpha_i Vk_1$, which is impossible as no linear combination of polynomials in $L_{\mathbb{G}}$ would lead to a polynomial with this term. So we must have that the pair $(P_\sigma^{(i)}, P_A^{(i)})$ must be of the following form.

$$P_\sigma^{(i)} = \sum_{n=1}^{q_{\mathcal{O}}} \beta_{in}\sigma_n$$

$$P_A^{(i)} = \gamma_i + \sum_{n=1}^{q_\mathcal{O}} \omega_{in} P_{A_n}$$

For them to satisfy (1), we must have that $\gamma_i = 0$ for all $i \in [q_\mathcal{O} + 1]$. Therefore, we can expand the equation to the following for all $i \in [q_\mathcal{O} + 1]$ and $j \in J$.

$$P_A^{(i)} = \sum_{n=1}^{q_\mathcal{O}} \omega_{in} (\sum_j \sigma_n V k_1 + \sigma_n P_n V k_j)$$

For equality (7) to hold, it means that at least one value of n exists where $\beta_{in} \neq 0$ for any $i \in [q_\mathcal{O} + 1]$. So for (6) to hold, we must have that for all $n \in [q_\mathcal{O}]$ that $\beta_{in} = \omega_{in}$, given by the monomial $\sigma_n V k_1$. The monomial $\sigma_n V k_j$ also gives us that for all $n \in [q_\mathcal{O}]$ and $j \in J$ that $\beta_{in} m_i = \omega_{in} P_n$. Therefore this must mean that $P_n = m_i$. If we have more than one value of n where $\beta_{in} \neq 0$, then this means that they use the same κ value when querying the BS oracle. Therefore, because the only case where you can produce a new signature pair is when the same message m is used, this means that it is impossible to produce $q_\mathcal{O} + 1$ distinct signature-message pairs on $q_\mathcal{O}$ queries.

The next step is bounding the probability of the simulation failing, and show that the probability is negligible (in λ). The simulation will only fail if there are two polynomials in L_i for $i \in \{\mathbb{G}, \hat{\mathbb{G}}, \mathbb{T}\}$ where $P \neq P'$ but $P(l_1, ..., l_{q_\mathcal{O}}, Sk_1, ..., Sk_\delta, h) = P'(l_1, ..., l_{q_\mathcal{O}}, Sk_1, ..., Sk_\delta, h)$ and $l_1, ..., l_{q_\mathcal{O}}, Sk_1, ..., Sk_\delta, h \in \mathbb{Z}_p$. Below states all of these cases formally.

$$P, P' \in L_\mathbb{G}, P \neq P', P(l_1, ..., l_{q_\mathcal{O}}, Sk_1, ..., Sk_\delta, h) = P'(l_1, ..., l_{q_\mathcal{O}}, Sk_1, ..., Sk_\delta, h) \quad (3)$$

$$P, P' \in L_{\hat{\mathbb{G}}}, P \neq P', P(l_1, ..., l_{q_\mathcal{O}}, Sk_1, ..., Sk_\delta, h) = P'(l_1, ..., l_{q_\mathcal{O}}, Sk_1, ..., Sk_\delta, h) \quad (4)$$

$$P, P' \in L_\mathbb{T}, P \neq P', P(l_1, ..., l_{q_\mathcal{O}}, Sk_1, ..., Sk_\delta, h) = P'(l_1, ..., l_{q_\mathcal{O}}, Sk_1, ..., Sk_\delta, h) \quad (5)$$

The only time a negative power appears in a polynomial is when it contains H. Therefore we can view all polynomials in the lists as a fraction, in some form $P = \frac{N}{D}$, where $N \in \mathbb{Z}_p[l_1, ..., l_{q_\mathcal{O}}, Vk_1, ..., Vk_{\delta''}, H]$ and $D \in \mathbb{Z}_p[H]$. Because D will only contain monomials of the form H^n, for values where $n \geq 0$, we can rewrite the polynomial check as

$$N(l_1, ..., l_{q_\mathcal{O}}, Sk_1, ..., Sk_\delta, h) D'(h) = N'(l_1, ..., l_{q_\mathcal{O}}, Sk_1, ..., Sk_\delta, h) D(h).$$

Now we can bound these polynomials. When the simulation starts, the only non-constant polynomial in $L_\mathbb{G}$ is H. At the end of the game, the polynomial with the largest degrees in $L_\mathbb{G}$ would be the result of querying the BS oracle on the κ value that corresponds to H, then repeatedly querying the oracle using the κ_A value returned. The form of this polynomial would be

$$P = \alpha + \sum_{i=1}^{q_\mathcal{O}} \beta_i \sum_{j \in J} (\prod_{n=1}^{q_\mathcal{O}} \sigma_n V k_1 V k_j^{q_\mathcal{O} - i} + \gamma \prod_{n=1}^{q_\mathcal{O}} \sigma_n H V k_j^{q_\mathcal{O}}).$$

The above ensures that any polynomial in $L_\mathbb{G}$ will satisfy $deg(N) \leq 2q_\mathcal{O} + 1$ and $deg(D) = 0$. From the Schwartz-Zippel lemma, we know that the probability that

(3) happens is bounded from the above by $\frac{2q_{\mathcal{O}}+1}{p}$. All polynomials that $\mathcal{A}$ can obtain in $L_{\hat{\mathbb{G}}}$ are linear combinations of $Vk_1, H^{-}1$ and Vk_j, for some $J \in J$. So, for any polynomial $P = \frac{N}{D}$ in $L_{\hat{\mathbb{G}}}$, the possible degrees that it can have is $deg(N) \in \{0, 1, 2\}$ and $deg(D) \in \{0, 1\}$, from the Schwartz-Zippel lemma, this means we have the probability that (9) happens is bounded from above by $\frac{3}{p}$. So, this means that all polynomials in $L_{\mathbb{T}}$ must also be of the form $\frac{N}{D}$, and have $deg(N) \leq 2q_{\mathcal{O}} + 3$ and $deg(D) \in \{0, 1\}$. Using the Schwartz-Zippel lemma, the probability that (10) occurs is bounded from above by $\frac{2q_{\mathcal{O}}+4}{p}$.

Summing over all possibilities of P and P' for each list, we have that the probability ϵ of the simulation failing for these is

$$\epsilon \leq \binom{|L_{\mathbb{G}}|}{2} \frac{2q_{\mathcal{O}} + 1}{p} + \binom{|L_{\hat{\mathbb{G}}}|}{2} \frac{3}{p} + \binom{|L_{\mathbb{T}}|}{2} \frac{2q_{\mathcal{O}} + 4}{p}.$$

which simplifies to

$$\epsilon \leq \frac{(6 + q_G + 3q_{\mathcal{O}} + q_P)^2(2q_{\mathcal{O}} + 4)}{p}$$

From the Schwartz-Zippel lemma and the bounds on the polynomials above, we have that the probability of sampling the root of a polynomial in the denominator is bounded from above by $\frac{6+q_G+3q_{\mathcal{O}}+q_P}{p}$.

Therefore, the probability of the simulation failing is

$$\leq \frac{(6 + q_G + 3q_{\mathcal{O}} + q_P)^2(2q_{\mathcal{O}} + 4) + (6 + q_G + 3q_{\mathcal{O}} + q_P)}{p})$$

.

Which gives us that $\mathcal{O}(\frac{q_G^2 q_{\mathcal{O}} + q_P^2 q_{\mathcal{O}} + q_{\mathcal{O}}^3}{p})$. Since $q_{\mathcal{O}}$, q_G and q_P are all polynomial in λ whereas $logp \in \Theta(\lambda)$, it means that the adversary $\mathcal{A}$'s advantage is negligible.

6 Results and Evaluation

This section will evaluate the proposed transformation framework, giving results in the form of an example instantiation of the transformation, providing a security proof for it. This example instantiation allows us to infer whether the transformation was successful in yielding a blind counterpart for any LmSPSPS scheme.

6.1 Example Instantiation

The LmSPSPS scheme that we will use for the example is outlined in section 4 of [4]. Figure 2 shows this scheme in the left column, and the obtained blind signature is in the right column. We assume that we are only signing one message in this example. We will not go step by step through the transformation framework, instead we will just provide the basic scheme that it will transform into, this alone gives us enough to prove the security of the framework and the subsequent new blind signature scheme.

LmSPSPS	BS
$KeyGen_{DS}(\mathcal{P})$:	$KeyGen_{BS}(\mathcal{P})$:
Select $x_1, ..., x_n, y, z \leftarrow \mathbb{Z}_p^{\times}$;	Select $x_1, ..., x_n, y, z, h \leftarrow \mathbb{Z}_p^{\times}$;
Set $Sk_{DS} := (y_1, ..., y_{n_1}, x, z)$,	Set $Sk_{BS} := (y_1, ..., y_{n-1}, x, z, h)$,
$Vk_{DS} := (Y_1, ..., Y_{n-1}, X, Z)$;	$Vk_{BS} := (Y_1, ..., Y_{n-1}, X, Z, H, \hat{H})$;
$= (\hat{G}^x, \hat{G}^y, ..., \hat{G}^{y_{n-1}}, \hat{G}^z) \in \hat{\mathbb{G}}^{n+1}$	$= (\hat{G}^{y_1}, ..., \hat{G}^{y_{n-1}}, \hat{G}^x, G^h, \hat{G^{\frac{1}{h}}}) \in \hat{\mathbb{G}}^{n+1}$
$Sign_{DS}$:	$Request^0_{BS}$:
Select $r \leftarrow \mathbb{Z}_p^{\times}$;	Select $r \leftarrow \mathbb{Z}_p^{\times}$;
Compute $\Sigma_{DS} = (S_1, S_2))$;	Generate commitment value $Co := G^m H^r$;
$S_1 = G^r$ and $S_2 = G^{\frac{r(x+my)}{z}}$	Save state $st = (m, r)$
	Return $\rho = Co$
	$Issue_{BS}(Sk_{BS}, \rho)$:
	Select $a \leftarrow \mathbb{Z}_p$;
	$\beta_1 = G^a$;
	$\beta_2 = (A' = (G^x Co^y)^{\frac{a}{z}}, B' = H^{\frac{ay}{z}})$;
	Return $\beta = (\beta_1, \beta_2)$ to user.
	$Request^1_{BS}(Vk_{BS}, \beta, st)$:
	Parse β as (β_1, β_2);
	Parse β_2 as (A', B');
	Compute $A' = A' B'^{-r}$;
	Compute final signature $\Sigma_{DS} = (\sigma_1, \sigma_2)$;
	$= Randomise_{DS}(\beta_1, A')$
$Verify_{DS}$:	$Verify_{BS}$:
Return 1 if $S_1 \neq 1_{\mathbb{G}}$ and	Return 1 if $\sigma_1 \neq 1_{\mathbb{G}}$ and
$e(S_2, Z) = e(S_1, X\hat{G}Y_1^m)$	$e(\sigma_2, Z) = e(\sigma_1, X\hat{G}Y_1^m)$

Fig. 2. Example Instantiation Table

6.2 Proving the security of the example

A blind signature requires three integral properties to be considered proper. Which are correctness, unforgeability, and blindness.

Theorem 2. The construction satisfies the correctness property.

Proof. We generated the commitment $Co = G^m H^r$, and using this commitment generated a pre-signature on the element that contained m (S_2 in the original scheme). This gave us the pre-signature pair $(A' = (G^x Co^y)^{\frac{a}{z}}, B' = H^{\frac{ay}{z}})$. Substituting the commitment value Co into the equation we get $(G^x (G^m H^r)^y)^{\frac{a}{z}}$, which gives us $G^{\frac{a(x+my)}{z}} H^{\frac{ayr}{z}}$. Therefore when we compute $A' = A'B'^{-r}$, we get $G^{\frac{a(x+my)}{z}} H^{\frac{ayr}{z}} \cdot (H^{\frac{ay}{z}})^{-r}$, which gives us $G^{\frac{a(x+my)}{z}}$. Which is the original form of the signature S_2, because the signature verification is the same, this satisfies the correctness property.

Theorem 3. The construction is unforgeable. For this proof, we will take the case where $z = 1$. As this gives us a signature polynomial that corresponds to the BS assumption.

Proof. Let $\mathcal{A}$ be an adversary against the unforgeability of this scheme. We show how to use $\mathcal{A}$ to construct an adversary $\mathcal{B}$ against the BS assumption. Adversary $\mathcal{B}$ receives Vk_{BS} and the bilinear group description $\mathcal{P}$ from their game and they have access to the oracle $\mathcal{O}BS_{Vk_{BS}}(\cdot)$, which they can query polynomially many times. Adversary $\mathcal{B}$ starts $\mathcal{A}$ on $Vk_{BS} = (Y_1, Y_{n-1}, X, Z, H, \hat{H})$. When $\mathcal{B}$ is queried on commitment Co_i, they forward the query to the oracle and return the answer to $\mathcal{A}$. Eventually, when the game finishes and $\mathcal{A}$ outputs their $k+1$ signature-message tuples $(m_i, \sigma_1^{(i)}, A_i)_{i=1}^{k+1}$, $\mathcal{B}$ returns that as their answer to their game. Because they both output the same signature-message tuples, it is clear that $\mathcal{A}$ and $\mathcal{B}$ have the same advantage in their respective games.

Theorem 4. The construction is perfectly blind.

Proof. Since the transformation is based on using Pedersen's commitment scheme, it is clear that Co is perfectly hiding and reveals no information about the committed message to the signer. When we check the validity of the pre-signature, this ensures that each pre-signature is a valid signature on the commitment Co. If any of the pre-signatures is invalid we return $(\perp, \perp)$ as it would fail to verify both the pre-signature check and the final signature check. This makes it clear that in the event that the pre-signature isn't valid, no information is leaked, meaning an adversary is unable to gain any information from a failed protocol run. If the checks for all of the pre-signatures pass, this means that all pre-signatures are valid signatures on their commitment Co_n, for a message m_n. In the situation where we have multiple pre-signatures that contain a committed message, the adversary would not be able to differentiate between a signature for the commitment Co_i on a message m_i and a signature for the commitment Co_j on a message m_j. The final step is to show that the pre-signature is unlinkable to the final signature. The $Randomise_{DS}$ function is a function that generates a new random exponent from $\mathbb{Z}_p$, therefore each signature must be evenly distributed across the possible signature space. All of this together proves that the obtained blind signature is perfectly blind.

6.3 Overall Generic Transformation Framework Evaluation

The result of the instantiation was as expected, successful. Using the BSOMI assumption from [1] allowed us to create our own BS assumption, which generalised the deductive reasoning used by Ghadafi to draw conclusions on my own framework's

possible linear combinations.

However, the assumption used to prove the Generic Transformation Framework only functions for the case where the signature consists of 2 components, one which depends on the message and the other does not. It also only covers cases where there is no denominator in the signature polynomial or a case where you can set it equal to 1. The proof was unilateral, which means it would still hold if the groups that the signature and verification key belong to switched, e.g $\Sigma_{BS} = (\sigma_1, \sigma_2) \in \hat{\mathbb{G}}^2$ and $Vk_{BS} = (Vk_1, ..., Vk'_\delta) \in \mathbb{G}^{\delta'}$. But this means wouldn't hold if the signature or verification key consisted of elements from both groups.

The framework provides a good level of readability and leaves a lot up to the interpretation of the reader. For example, the signature is decomposed into a notation that encapsulates all possibilities, which means that any combination of a secret key element, message, and randomness can be transformed using the framework. However, for the cases discussed in the above paragraph, the security proof we provide does not serve as a standalone proof. It would definitely serve as a good foundation for someone to create their own security proof.

6.4 Summary

This section discussed the primary result of the project, a working generic transformation framework for Linear-message Strongly Partially Structure-Preserving Signatures. To show this, we produced an example transformation of the scheme outlined in section 4 of [4]. This section showed that the transformation worked and that the obtained blind signature satisfied all the properties required of a blind signature scheme, provided in section 2.1.4.

7 Conclusions and further work

With the main objective of providing privacy-preserving signatures to the reader, we created a Generic Transformation Framework in section 4, and in the following section, provided a security proof that shows that all possible combinations of signature components cannot lead to an adversary being able to output an extra set of signature-message tuples than what they queried to the oracle. We then showed that the transformation worked by providing a concrete example of it working in the form of the example instantiation, giving us a result to be proud of.

To measure the success, we will elaborate on whether we met the aim and objectives outlined in section 1.3.

7.1 Aim

To produce a generic framework to transform any Linear-Message Strongly Partially Structure-Preserving Signature (LmSPSPS) scheme into a Blind signature scheme, and evaluate its security.

This project aim has been successfully completed. However, the only technicality would be that the security proof was not as rigorous as we would've hoped, the time constraint was the main cause of this.

7.2 Objectives

This section will go over each of the objectives, stating whether they have been met, and if so in which section.

(i) Understand how LmSPSPS schemes work, providing formal definitions for the family of signatures.

This is provided in the section 2.1.3. We reviewed the literature and provided the parts that were relevant to this project. This objective has been comfortably met.

(ii) Review and understand existing Blind Signature schemes based on bilinear groups.

This is provided in sections 2.1.4 and 3.2.1. We reviewed and provided a more abstract version of the protocol than provided by the sources, meeting this objective.

(iii) Generalise the way blindness is used in these existing schemes.

This objective has been met in section 3.2. Which details the process undertaken to produce the generic framework, meeting this objective nicely.

(iv) Produce a generic transformation framework to transform an LmSPSPS scheme to its blind counterpart.

This objective has been met in section 4. Where we defined the full generic transformation.

(v) Prove the security of the obtained Blind Signature.

This objective has been met in section 5. Where we provide the entire assumption behind the proof.

(vi) Produce an example instantiation of an LmSPSPS scheme to its blind counterpart and prove its security.

This is provided in figure 2 of the section 6.1, we used the security proof for the transformation and applied it to the case of the LmSPSPS scheme in the table, giving us a full proof. Meeting this objective.

(vii) Discuss the differences between the original LmSPSPS scheme and the example instantiation.

This is provided in section 6.3. This section gives a brief discussion about whether the results we achieved were expected or not. It mainly covers the security proof limitations, but still meets this objective.

7.3 Future Work

The Future work for this dissertation would mainly focus on providing more rigorous security proof. Whether this be to prove that the entire generic framework is considered secure, or provide more cases to prove the security, like the bilateral case briefly mentioned in the evaluation.

Another way to extend this project would be to extend the framework itself to encapsulate more digital signature schemes, one idea of how this could be done is to transform any Partially Structure-Preserving Signature, rather than just the Linear-message Strongly Structure-Preserving Signature variant.

7.4 Final Conclusions

Overall, this dissertation has been a success. The aim and objectives have all been met, not to the standard we hoped, but nevertheless, the final product is something to be proud of. This framework is provably secure for a large portion of Linear-message Strongly Structure-Preserving Signatures, allowing for it to be developed on in the future potentially.

References

[1] E.Ghadafi, "Efficient Round-Optimal Blind Signatures in the Standard Model," 2017. Accessed: 2023-05-01. [Online]. Available: https://eprint.iacr.org/2017/045.pdf.

[2] D.Chaum, "Blind Signatures for Untraceable Payments," Advances in Cryptology, pp. 199–203, 1983, doi: https://doi.org/10.1007/978-1-4757-0602-4_18. Last accessed 2023-05-05.

[3] R.L.Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," Communications of the ACM, vol. 21, no. 2, pp. 120–126, 1978, doi: https://doi.org/10.1145/359340.359342. Last accessed 2023-05-19.

[4] E.Ghadafi, "Partially Structure-Preserving Signatures: Lower Bounds, Constructions and More," ePrint IACR, 2020. https://eprint.iacr.org/2020/477. Last accessed 2023-05-29.

[5] T.P.Pedersen, "Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing," Advances in Cryptology - CRYPTO '91, pp. 129–140, doi: https://doi.org/10.1007/3-540-46766-1_9. Last accessed 2023-05-29.

[6] S. D. Galbraith, K. G. Paterson, and N. P. Smart, "Pairings for cryptographers," Discrete Applied Mathematics, vol. 156, no. 16, pp. 3113-3121, Sep. 2008, doi: https://doi.org/10.1016/j.dam.2007.12.010. Last accessed 2023-05-29.

[7] V. Shoup, "Lower Bounds for Discrete Logarithms and Related Problems," pp. 256–266, May 1997, doi: https://doi.org/10.1007/3-540-69053-0_18. Last accessed 2023-05-29.

[8] J. C. Benaloh, "Secret Sharing Homomorphisms: Keeping Shares of a Secret Secret (Extended Abstract)," Advances in Cryptology — CRYPTO' 86, pp. 251–260, doi: https://doi.org/10.1007/3-540-47721-7_19. Last accessed 2023-05-31.