T.R.
GEBZE TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

PAIRING BASED CRYPTOGRAPHY AND ITS APPLICATIONS

ÖZNUR KALKAR
A THESIS SUBMITTED FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
DEPARTMENT OF MATHEMATICS

GEBZE

2022

T.R.
GEBZE TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

PAIRING BASED CRYPTOGRAPHY AND ITS APPLICATIONS

ÖZNUR KALKAR
A THESIS SUBMITTED FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
DEPARTMENT OF MATHEMATICS

THESIS SUPERVISOR
ASSOC. PROF. DR. SEHER TUTDERE KAVUT
II. THESIS SUPERVISOR
DR. İSA SERTKAYA

GEBZE
2022

T.C.

GEBZE TEKNİK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

EŞLEME TABANLI KRİPTOGRAFİ VE UYGULAMALARI

ÖZNUR KALKAR

DOKTORA TEZİ

MATEMATİK ANABİLİMDALI

DANIŞMANI

DOÇ. DR. SEHER TUTDERE KAVUT

II. DANIŞMANI

DR. İSA SERTKAYA

GEBZE

2022

GEBZE TEKNİK ÜNİVERSİTESİ	**YÜKSEK LİSANS JÜRİ ONAY FORMU**

GTÜ Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 07/07/2022 tarih ve 2022/34 sayılı kararıyla oluşturulan jüri tarafından 01/08/2022 tarihinde tez savunma sınavı yapılan Öznur Kalkar'ın tez çalışması Matematik Anabilim Dalında DOKTORA tezi olarak kabul edilmiştir.

JÜRİ

ÜYE
(TEZ DANIŞMANI) : DOÇ. DR. SEHER TUTDERE KAVUT

ÜYE : PROF. DR. CEM GÜNERİ

ÜYE : PROF. DR. MURAT CENK

ÜYE : PROF. DR. MUSTAFA AKKURT

ÜYE : DR. ÖĞR. ÜYESİ ROGHAYEH HAFEZIEH

ONAY

Gebze Teknik Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun

...../...../..... tarih ve ...../..... sayılı kararı.

İMZA/MÜHÜR

ÖZET

Eşleme fonksiyonlarının kriptografide saldırı yöntemi olarak ortaya çıkışından sonra, eşleme tabanlı kriptografi oldukça aktif bir araştırma alanı haline geldi. Eşleme fonksiyonlarının bilineer özelliği, daha önce mümkün olmayan çok çeşitli kriptografik uygulamaları mümkün kılar; kimlik tabanlı şifreleme, kısa imzalar, anonim öznitelik ve kısa sıfır bilgi protokolleri bu örneklerden sadece birkaçıdır. Bununla birlikte, eşleme fonksiyonunun hesaplanması, kriptografik işlemler arasında en pahalı olanlardan biridir. Bu tezin temel olarak eşleme tabanlı kriptografiye üç alanda katkıları vardır. Bu tezde ilk olarak, mahremiyet arttırıcı ve transfer edilebilir bir elektronik çek defteri şeması ve bu şemalar için oyun tabanlı güvenlik tanımları önerikmektedir ve önerilen bu sistemin elektronik çek defterinin değiştirilemezliği, elektronik çek değiştirilemezliği ve manipüle edilemezliği ve elektronik çekin anonimliği özelliklerine sahip olduğu gösterilmektedir.

İkinci olarak, doğrulanabilir öznitelik (örneğin anonim öznitelik) mekanizmalarıyla gerçeklenen blok zinciri tabanlı bir sınav mekanizması verilmektedir. Bu şemanın, elektronik sınavların sağlaması gereken özellikleri sağladığını tartışılmaktadır.

Son olarak, eşleme fonksiyonunun güvenli ve doğrulanabilir bir şekilde tevdii edilmesine odaklanılmaktadır. Tek bir hesaplamanın tevdii edilmesi için önerilen mekanizmalar araştırılmış, bazılarına yapılan saldırılardan bahsedilmektedir, iki sunucu kullanarak güvenli ve tamamen doğrulanabilir bir protokol veriyor ve bahsedilen mekanizmaların verimlilikleri karşılaştırılmaktadır. Ayrıca, eşleme fonksiyonlarının toplu bir şekilde tevdii edilmesi araştırılmakta ve eşleme fonksiyonların girdilerinin gizli/açık, değişken/sabit olma durumlarına göre farklı türler için en verimli algoritmalar önerilmektedir.

Anahtar Kelimeler: Eşleme Tabanlı Kriptografi, Eşleme Fonksiyonunun Delege Edilmesi, Elektronik Çek Defteri, Blokzincir Tabanlı Sınav.

SUMMARY

Pairing-based cryptography has become a highly active research area since pairings' first appearance in the cryptography as an attack method. The bilinear property of pairings enable wide range of cryptographic applications that were not possible before; identity-based encryption, short signatures, anonymous credentials, and zero-knowledge succint non-interactive arguments of knowledge protocols are only a few of these examples. However, pairing computation is one of the most expensive tasks among cryptographic operations. This thesis has contributions mainly in three areas of pairing based cryptography. First, we give a privacy enhanced transferable e-checkbook mechanism along with formal game-based security definitions and prove that the proposed mechanism has e-checkbook unforgeability, e-check unforgeability and non-manipulability, and e-check anonymity properties.

Second, we give a blockchain based remote exam mechanism which is realized by verifiable credentials, for example anonymous credentials. The scheme satisfies test answer authentication, examiner authentication, anonymous marking, anonymous examiner, question secrecy, question privacy, mark privacy, test verifiability, and mark verifiability properties.

Lastly, we focus on secure and verifiable pairing outsource. For the single pairing outsource, we investigate the proposed mechanisms, include attacks for some of them, give a secure and fully verifiable outsourcing scheme in multi server setting, and compare their efficiency. Furthermore, we delve into batch pairing outsource and propose the most efficient algorithms for different types based on inputs to the pairing being secret/public, variable/constant.

Keywords: Pairing-based Cryptography, Pairing Outsource, Pairing Delegation, Electronic Checkbook, Blockchain-based Exam.

ACKNOWLEDGEMENTS

I would like to express my deepest appreciation to my co-advisor Dr. İsa Sertkaya. He has been a great mentor since the day we met. If he has not been there, this thesis would not be completed. I am also grateful to my advisor Assoc. Prof. Dr. Seher Tutdere and I also would like to thank to my committee members.

I would like to dedicate this thesis to my son Deniz, and would like to extend my sincere thanks to my husband Sabri for being there for me, and Deniz for making me see the joy in little things. I wish he never stops experimenting and having joy. My thanks also go to my family for always believing in me.

Studying mathematics was a fun but challenging but fun journey. So, thank you Roger Penrose for writing The Emporer's New Mind, and Dan Brown for making me interested in cryptography and leading to creation of my own super unsafe ciphers in the high school. All my mathematician friends whom we studied together so many nights also deserve a sincere thank you. Studying mathematics gave me the opportunity to study/work/converse with many bright people, for that, i am very grateful.

TABLE of CONTENTS

LIST of ABBREVIATIONS and ACRONYMS

Abbreviations and Acronyms		**Explanations**
E	:	Elliptic Curve
e	:	Bilinear Pairing Map
$\mathbb{G}_1$	:	Additively written group of prime order r
$\mathbb{G}_2$	:	Additively written group of prime order r
$\mathbb{G}_T$	:	Multiplicatively written group of prime order r
$\mathcal{H}$	:	Cryptographically secure hash function
pec	:	Public-key encryption scheme
sig	:	Digital signature scheme
$\mathsf{pp}_{(\cdot)}$	:	Public parameters for the cryptographic primitive or protocol $(\cdot)$
$\mathsf{sk}_{(\cdot)}$	:	Secret key of $(\cdot)$
$\mathsf{pk}_{(\cdot)}$	:	Public key of $(\cdot)$ corresponding to the secret key $\mathsf{sk}_{(\cdot)}$
DLP	:	Discrete Logarithm Problem
ECDLP	:	Elliptic Curve Discrete Logarithm Problem
SSI	:	Self-sovereign Identity

LIST of FIGURES

LIST of TABLES

1. INTRODUCTION

Bilinear pairings have great importance in cryptography and this thesis focus on both applications of pairings and how to outsource computation of pairings.

The security of public key cryptosystems relies on difficulty of solving some mathematical problems. In order to break the public key cryptosystems, people focus on finding fast ways to solve these problems. One of these problems is called Discrete Logarithm Problem (DLP). Suppose that we have a multiplicatively written cyclic group $\mathbb{G} =< g >$ of order n, then DLP can be described as given g, h, finding $1 \leq x \leq n-1$ such that $h = g^x$.

In 1985, Koblitz [1987] and Miller [1985] proposed using elliptic curves in public key cryptosystems and their setting introduced another hard problem, namely Elliptic Curve Discrete Logarithm Problem (ECDLP). This problem is very similar to the discrete logarithm problem as name suggests. Suppose we are given an elliptic curve E defined over a finite field $\mathbb{F}_q$, $P \in E(\mathbb{F}_q)$ of order n and $Q \in< P >$, the ECDLP is to find the integer $1 \leq x \leq n-1$ such that $xP = Q$. In 1993, Menezes et al. [1993] proposed an algorithm to attack the Elliptic Curve Discrete Logarithm Problem. The MOV attack used elliptic curve pairings successfully in order to reduce ECDLP to DLP in a smaller group. This was the introduction of bilinear pairings to the cryptography community.

Key exchange protocols have importance in cryptography. A key-exchange algorithm basically enables multiple parties to agree on a shared secret in order to use that secret for encryption. In 2000, Joux [2000] proposed a fairly simple three party key exchange protocol using bilinear pairings which allows three parties to construct a shared secret in a single round. This was the first construction which used elliptic curve pairings. In 2001, another two cryptographic schemes were proposed which are identity based encryption by Boneh and Franklin [2001] and short signature by Boneh et al. [2001]. Prior to identity-based encryption, if someone wanted to send an encrypted message to a party, then that someone needed to know the public key of the recipient. Identity-based encryption enabled sending an encrypted message to a party without knowing their public key before hand with the help of a key-generation center. These applications made bilinear pairings gain public attention. Later on, another

use cases were discovered for bilinear pairings like attribute-based encryption and searchable encryption. Nowadays one of the most popular use case of bilinear pairings is verification of a zero-knowledge proof. A zero-knowledge proof is a method where a prover can convince a verifier that a given statement is true while without giving any additional information other than the fact that the statement is true. These proofs have many use-cases in the blockchain space, private-payments, self-sovereign identity, and roll-ups are the most significant examples.

Even though elliptic curve pairings offer many use-cases, they are the most expensive computation in cryptography. In order to benefit elliptic curve pairings, people have been working on finding ways to improve efficiency of calculating a pairing. Some are trying to achieve this by means of finding faster calculation and implementation methods Koblitz and Menezes [2005]; Hess et al. [2006]; Scott et al. [2006]; Barreto et al. [2007]; Beuchat et al. [2010] and some others are trying to find secure and efficient ways to outsource this computation to more powerful agents Hohenberger and Lysyanskaya [2005]; Chevallier-Mames et al. [2005]; Kang et al. [2005]; Canard et al. [2014]; Chen et al. [2015]; Tian et al. [2015]; Arabacı et al. [2015]; Ren et al. [2017].

This thesis gives the background on elliptic curves and pairings and describes applications of pairing based cryptography in Chapter 2. Chapter 3 defines electronic checkbook schemes, gives game-based security definitions, proposes an electronic checkbook mechanism and proves that the scheme satisfies e-checkbook unforgeability, e-check unforgeability and non-manipulability, e-check anonymity and resistant againt double payments and replay attacks. Chapter 4 defines electronic exams, gives requirements and security definitions, and propose a blockchain based solution which satisfies test answer authentication, examiner authentication, anonymous marking, anonymous examiner, question secrecy, question privacy, mark privacy, test verifiability, and mark verifiability properties. Chapter 5 describes delegation of a single pairing, gives a literature review and compares the algorithms. After single pairing outsourcing, Chapter 6 studies batch pairing delegation, defines the problem, states the taxonomy, proposes mechanisms for different types, and gives a comparison. Finally, Chapter 7 concludes the thesis.

2. FOUNDATIONS

This chapter aims to give the necessary background on the concepts that are mentioned throughout the thesis. We set the notation, describe some cryptographic primitives, give a brief description of elliptic curves and elliptic curve pairings, and present some of the fundamental pairing based cryptographic applications.

2.1. Notation

- $\leftarrow$: assigning an output value to a specific variable,
- $\leftarrow_{\$}$: assigning to a variable, a random, uniformly distributed element of a set,
- κ: security parameter,
- $\{0,1\}^*$: arbitrary length bit-string,
- $\{0,1\}^\kappa$: bit string of length κ,
- $\mathsf{negl}(\kappa)$: negligible function that not only tends to zero as κ increases but also does so faster than the inverse of any polynomial,
- $s||t$: ordered concatenation of two strings s then t,
- $\perp$: error returned from a process,
- $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^\kappa$: pre-image resistant hash function,
- $\mathcal{H}^x(\cdot)$: iterative computation of x-th hash of the given input,
- $[a]P$: a scalar multiplication of in an additive group containing P of order r by a scalar $a \in \mathbb{Z}_r$, i.e. $P + P + \cdots + P$, a times,

2.2. Cryptographic Primitives

Before public key cryptography, there was symmetric cryptography where a key is used both for encrypting and decrypting a message. For that to happen, sender and recipient needs to somehow agree on a shared secret key. This process is quite tricky because either this key can be shared in-person or via a secure-channel. Even if this is satisfied, this key need to be updated and this sharing process becomes a burden when the number of participants increases. Asymmetric-key cryptography on the other hand,

allows anyone knowing the public key of the recipient to send an encrypted message and sharing that public key is not a problem. In the asymmetric-key cryptography, people have two keys, a secret key and a public key which is generated using the private key. For an encryption algorithm, anyone can send a message using the public key while only the secret key can decrypt the encrypted message.

Asymmetric-key cryptography also provides a mechanism for signature. Again, people have two keys and this time the signer can sign a message with his private key while the verifier is able to verify the given signature if he has the message and the public key of the signer. Generally, that message needs to be hashed before signing process and hashing can be described as mapping an arbitrary length data to a fixed-size bit array. They are one-way functions, i.e given a message, computing the hash is easy while given the hash value, finding the message that results in the given hash is infeasible.

Now we describe public key encryption and digital signature scheme in more details.

- Public key encryption scheme $\mathrm{Pec} = (\mathsf{G}_{\mathsf{pec}}, \mathsf{K}_{\mathsf{pec}}, \mathsf{E}_{\mathsf{pec}}, \mathsf{D}_{\mathsf{pec}})$ where
  - Setup $(\mathsf{pp}_{\mathsf{pec}} \leftarrow \mathsf{G}_{\mathsf{pec}}(\kappa))$: On input a security parameter κ, outputs public parameters as $\mathsf{pp}_{\mathsf{pec}}$,
  - Keygen $((\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{K}_{\mathsf{pec}}(\mathsf{pp}_{\mathsf{pec}}))$: Takes public parameters $\mathsf{pp}_{\mathsf{pec}}$ and outputs a public/private encryption key pair,
  - Encrypt $(c \leftarrow \mathsf{E}_{\mathsf{pec}}(\mathsf{pk}, m))$: Takes a public key pk and a message m, encrypts the message m and outputs ciphertext as c,
  - Decrypt $(m \leftarrow \mathsf{D}_{\mathsf{pec}}(\mathsf{sk}, c))$: On input of private key sk and a ciphertext c, decrypts c to m.
- Digital signature scheme $\mathrm{Sig} = (\mathsf{G}_{\mathsf{sig}}, \mathsf{K}_{\mathsf{sig}}, \mathsf{S}_{\mathsf{sig}}, \mathsf{V}_{\mathsf{sig}})$ where
  - Setup $(\mathsf{pp}_{\mathsf{sig}} \leftarrow \mathsf{G}_{\mathsf{sig}}(\kappa))$: Given a security parameter κ, yields public parameters as $\mathsf{pp}_{\mathsf{sig}}$,
  - Keygen $((\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{K}_{\mathsf{sig}}(\mathsf{pp}_{\mathsf{sig}}))$: On input of public parameters $\mathsf{pp}_{\mathsf{sig}}$, outputs a public/private signing key pair,
  - Sign $(\sigma \leftarrow \mathsf{S}_{\mathsf{sig}}(\mathsf{sk}, m))$: On input a message m and private signing key sk, generates the signature σ,
  - Verify $(\{0,1\} \leftarrow \mathsf{V}_{\mathsf{sig}}(\mathsf{pk}, m, \sigma))$: On input public key pk, a message m and signature σ, checks if the signature σ is valid. If valid, outputs 1, otherwise 0.

2.3. Elliptic Curves

To make the thesis self-contained, here we are going to recall elliptic curve definitions, mainly following El Mrabet and Joye [2017]; Silverman [2009].

Elliptic curves are curves of genus one having a specified base point. Every such curve can be written as the locus in $\mathbb{P}^2$ of a cubic equation with only one point, the base point, on the line at ∞. Then, after X and Y are scaled appropriately, an elliptic curve has an equation of the form

$$Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3. \tag{2.1}$$

Here $O = [0, 1, 0]$ is the base point and $a_1, a_2, \ldots, a_6 \in \overline{K}$.

We generally write elliptic curves in non-homogeneous coordinates by letting $x = X/Z$, and $y = Y/Z$,

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \tag{2.2}$$

with an extra point $O = [0, 1, 0]$, point at infinity, in mind. Equation 2.2 is called as Weierstrass equation, and E is said to be defined over K if $a_1, a_2, \ldots, a_6 \in K$.

Definition 2.1: Silverman [2009] Let $P, Q \in E$, let L be the line through P and Q (if $P = Q$, let L be the tangent line to E at P), and let R be the third point of intersection of L with E. Let L' be the line through R and O. Then L' intersects E at R, O, and a third point. We denote that third point by $P \oplus Q$.

Proposition 2.1: Silverman [2009] The composition law (Definition 2.1) has the following properties:

i) If a line L at the (not necessarily distinct) points P, Q, R, then

$$(P \oplus Q) \oplus R = O. \tag{2.3}$$

ii) $P \oplus O = P$ for all $P \in E$.

iii) $P \oplus Q = Q \oplus P$ for all $P, Q \in E$.

iv) Let $P \in E$. There is a point of E, denoted by $\ominus P$, satisfying

$$P \oplus (\ominus P) = O. \tag{2.4}$$

v) Let $P, Q, R \in E$. Then

$$(P \oplus Q) \oplus R = P \oplus (Q \oplus R). \tag{2.5}$$

In other words, the composition law (2.1) makes E into an abelian group with identity element O. Further,

vi) Suppose that E is defined over K. Then

$$E(K) = \{(x, y) \in K^2 : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6\} \cup \{O\} \tag{2.6}$$

is a subgroup of E.

For the following, we drop the special symbols $\oplus, \ominus$ and simply write $+, -$ for the group operation on an elliptic curve E. For $m \in \mathbb{Z}$ and $P \in E$, we let

$$[m]P = P + \ldots + P, \quad m \text{ terms if } m > 0, \tag{2.7}$$

$$[m]P = -P - \ldots - P, \quad |m| \text{ terms if } m < 0, \tag{2.8}$$

$$[0]P = O. \tag{2.9}$$

Definition 2.2: Silverman [2009] An elliptic curve is a pair (E, O), where E is a non-singular curve of genus one and $O \in E$. (We generally denote the elliptic curve by E, the point O being understood.) The elliptic curve E is defined over K, written E/K, if E is defined over K as a curve and $O \in E(K)$.

2.4. Elliptic Curve Pairings

Definition 2.3: El Mrabet and Joye [2017] Let $\mathbb{G}_1, \mathbb{G}_2$ (additively written) and $\mathbb{G}_T$

(multiplicatively written) be groups of prime order r. A pairing *e is defined as a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ having the following properties:*

- *bilinearity: for all $A \in \mathbb{G}_1, B \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_r$, we have*

$$e([a]A, [b]B) = e(A, B)^{ab}, \tag{2.10}$$

- *non-degenerecy: for $A \neq 0_{\mathbb{G}_1}, B \neq 0_{\mathbb{G}_2}$, $e(A, B) \neq 1_{\mathbb{G}_T}$, where $0_{\mathbb{G}_1}$ (resp. $0_{\mathbb{G}_2}$ and $1_{\mathbb{G}_T}$) is the identity element of $\mathbb{G}_1$ (resp. $\mathbb{G}_2$ and $\mathbb{G}_T$).*

Then, a bilinear environment is a tuple, $(r, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P, Q, e)$, where $r, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ and e are defined as above, and P (resp. Q) is a generator of $\mathbb{G}_1$ (resp. $\mathbb{G}_2$).

For cryptographic and efficiency purposes, e is required to be efficiently computable, hard to inverse and to possess underlying groups on which the necessary computational assumptions holds.

2.4.1. Pairing Types

Let $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{G}_T$ be groups of order ℓ, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Pairings are classified into three among cryptographers.

- Type I: $\mathbb{G}_1 = \mathbb{G}_2$,
- Type II: $\mathbb{G}_1 \neq \mathbb{G}_2$ but there is an efficiently computable homomorphism $\phi : \mathbb{G}_2 \to \mathbb{G}_1$,
- Type III: $\mathbb{G}_1 \neq \mathbb{G}_2$ and there is no efficiently computable homomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$.

2.5. Pairing Based Cryptography

Pairing based cryptography is an area of cryptography that uses bilinear pairing functions in order to construct cryptographic algorithms. The unique features of these pairings enabled many new cryptographic algorithms that were not feasible before. When constructed properly, pairings provide the same level of security with traditional public key cryptosytems with smaller key sizes.

2.5.1. Identity Based Encryption

In 1984, the concept of identity-based encryption was introduced by Shamir. Identity-based encryption enables users having a secure communication without the need of exchanging public keys or acquiring a private key before the communication occurs. Although there were some non-pairing based identity based encryption protocols, Boneh and Franklin [2001] proposed the first fully functional scheme.

An identity-based encryption scheme E requires a Private Key Generator(KGC) in order to create the system parameters and distribute private keys to the users, and described by four randomized algorithms: Setup, Extract, Encrypt, Decrypt:

- Setup: On input of a security parameter K, creates the public system parameters and publishes as params. In addition, it outputs a master-key for PKG.
- Extract: By using master-key, public parameters and an ID, generates a private key d for ID.
- Encrypt: On input of params, ID, and a message M, encrypts M for ID and returns the ciphertext C.
- Decrypt: On input of params, C, and D, decrypts C by d and returns the plaintext message M.

2.5.1.1. Boneh-Franklin Identity Based Encryption

In 2001, Boneh and Frankin proposed a fully functional identity-based encryption scheme Boneh and Franklin [2001] using the Weil pairing. The scheme is defined as follows:

- Setup: Given a security parameter $k \in \mathbb{Z}^+$,

  i) generates a prime q, two groups $\mathbb{G}_1, \mathbb{G}_2$ of order q, and a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ and chooses a random generator $P \in \mathbb{G}_1$.

  ii) picks a random $s \in \mathbb{Z}_q^*$ and sets $P_{pub} = sP$.

  iii) chooses cryptographic hash functions $H_1 : \{0,1\}^* \to \mathbb{G}_1^*$, $H_2 : \mathbb{G}_2 \to \{0,1\}^n$.

  The message space is $\mathcal{M} = \{0,1\}^n$. The ciphertext space is $\mathcal{C} = \mathbb{G}_1^* \times \{0,1\}^n$. The system parameters are $params =< q, \mathbb{G}_1, \mathbb{G}_2, e, n, P, P_{pub}, H_1, H_2 >$. The master-key is $s \in \mathbb{Z}_q^*$.

- Extract: For a given string $\mathsf{ID} \in \{0,1\}^*$,

i) computes $Q_{\mathsf{ID}} = H_1(\mathsf{ID}) \in \mathbb{G}_1^*$, and

ii) sets the private key d_{ID} to be $d_{\mathsf{ID}} = sQ_{\mathsf{ID}}$ where s is the master key.

- Encrypt: To encrypt $M \in \mathcal{M}$ under the public key ID,

i) computes $Q_{\mathsf{ID}} = H_1(\mathsf{ID}) \in \mathbb{G}_1^*$,

ii) chooses a random $r \in \mathbb{Z}_q^*$,

iii) sets the ciphertext to be

$$C =< rP, M \oplus H_2(g_{\mathsf{ID}}^r) >, \quad \text{where} \quad g_{\mathsf{ID}} = e(Q_{\mathsf{ID}}, P_{pub}) \in \mathbb{G}_2^*. \tag{2.11}$$

- Decrypt: Let $C =< U, V > \in \mathcal{C}$ be a ciphertext encrypted using the public key ID. To decrypt C using the private key $d_{ID} \in \mathbb{G}_1^*$, computes

$$V \oplus H_2(e(d_{\mathsf{ID}}, U)) = M. \tag{2.12}$$

Theorem 2.1: Suppose the hash functions H_1, H_2 are random oracles. Then given mechanism is a semantically secure identity based encryption scheme (IND-ID-CPA) assuming BDH is hard in groups $\mathbb{G}_1$, $\mathbb{G}_2$. Concretely, suppose there is an IND-ID-CPA adversary $\mathcal{A}$ that has advantage $\epsilon(k)$ against the scheme. Suppose $\mathcal{A}$ makes at most $q_E > 0$ private key extraction queries and $q_{H_2} > 0$ hash queries to H_2. Then there is an algorithm $\mathcal{B}$ that solves BDH in groups $\mathbb{G}_1, \mathbb{G}_{\not=}$ with advantage at least:

$$Adv_{\mathbb{G}_1,\mathbb{G}_2,\mathcal{B}}(k) \geq \frac{2\epsilon(k)}{e(1+q_E)\cdot q_{H_2}}. \tag{2.13}$$

Here $e \approx 2.71$ is the base of the natural logarithm. The running time of $\mathcal{B}$ is $O(time(\mathcal{A}))$.

2.5.2. Digital Signature

The digital signature concept was suggested by Diffie and Hellman in 1976 when they put forward the idea of public key cryptography Diffie and Hellman [1976]. The aim of digital signature is to ensure the source authentication, non-repudiation and integrity.

The standard security of the digital signature is existential unforgeability against adaptively chosen messages attacks (EUF-CMA), which guarantees that the adversary cannot forge a valid signature on a new message, even if it can access the signing oracle that could provide the signing service.

2.5.2.1. BLS Signature

Boneh Lynn Shacham Short Signature was first proposed by Boneh and Franklin [2001]. Even if this signature allows aggregating of signatures, if extra precautions are not taken, rogue-key attacks are possible. In order to overcome this issue, Boneh et al. [2018] proposed a new version, that also supports multi-signature features.

Definition 2.4: Boneh et al. [2018]
Setup:

i) Generate a prime q, three groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of order q, and an admissible bilineer map $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$. Choose a random generators $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$, i.e. $\mathbb{G}_1 = \langle P \rangle$, $\mathbb{G}_2 = \langle Q \rangle$.

ii) Choose two hash functions $H_0 : \{0,1\}^* \to \mathbb{G}_1$ and $H_1 : \{0,1\}^* \to \mathbb{Z}_q$

iii) The system parameters are, params:$(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P, Q, H_0, H_1)$

Key Generation: Choose private key $\mathsf{sk} \leftarrow \mathbb{Z}_q$ and compute public key as $\mathsf{pk} \leftarrow [\mathsf{sk}]Q$.

Key Aggregation:

$$\mathsf{apk} \leftarrow \sum_{i=1}^{n} [a_i]\mathsf{pk}_i, \tag{2.14}$$

where $a_i \leftarrow H_1(\mathsf{pk}_i, \{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\})$.

Signing:

i) Each party computes $s_i \leftarrow [a_i \mathsf{sk}_i] H_0(m)$ (recall that $a_i \leftarrow H_1(\mathsf{pk}_i, \{\mathsf{pk}_1, \ldots, \mathsf{pk}_n\})$)

ii) Send s_i to a designated combiner

iii) Combiner computes the final signature as

$$\sigma \leftarrow \sum_{i=1}^{n} s_i. \tag{2.15}$$

Multi-Signature Verification: Accept if and only if the following holds.

$$e\left(\sigma, -Q\right) \cdot e\left(H_0(m), \mathsf{apk}\right) \stackrel{?}{=} 1_{\mathbb{G}_T} \tag{2.16}$$

2.5.3. Signcryption

A signcryption scheme is a cryptographic primitive which combines signature and encryption in one setting, in a more efficient way than performing encryption and signing individually.

2.5.3.1. BLCQ Signcryption

Here, we are going to recall the signcryption scheme proposed in Barreto et al. [2005], that is updated from the protocol given in McCullagh and Barreto [2004]. Please note that, here the protocol will be given in Type-III setting (as defined in Galbraith et al. [2008]). This is mainly due to the recent attacks on elliptic curves and hence on pairings, see Barbulescu and Duquesne [2019] and the references therein. Since we will use this scheme as a building block for our electronic checkbook scheme, it is given in more detail compared to the rest of the given primitives.

Pairing-based protocols generally involve hashing to elliptic curve subgroups, in the sequel, the following hash functions will be utilized.

- $\mathcal{H}_1 : \{0,1\}^* \to \mathbb{Z}_r^*$,
- $\mathcal{H}_2 : \{0,1\}^* \times \mathbb{G}_T \to \mathbb{Z}_r^*$,
- $\mathcal{H}_3 : \mathbb{G}_T \to \{0,1\}^*$.

SignC $= (\mathcal{G}, \mathcal{K}, \mathcal{S}, \mathcal{V})$ is a signcryption scheme, where each algorithm is given as follows.

- Setup ($\mathsf{pp} \leftarrow \mathcal{G}(1^\kappa)$). Given a security parameter κ, Key generation Center (KGC) constructs a bilinear environment with groups $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{G}_T$ of prime order $r > 2^\kappa$. Then, chooses a random secret $s \leftarrow_\$ \mathbb{Z}_r^*$ as its master private key $\mathsf{sk}_{\mathsf{KGC}} = s$, and publishes system wide public parameters pp as

$$\{r, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P, Q, e, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3, g, \mathsf{pk}_{\mathsf{KGC}}\} \tag{2.17}$$

where $g = e(P, Q)$ and $\mathsf{pk}_{\mathsf{KGC}} = ([s]P, [s]Q)$ for the signcryption scheme.

- Keygen $((\mathsf{sk}_{\mathsf{U}}, \mathsf{pk}_{\mathsf{U}}) \leftarrow \mathcal{K}(\mathsf{pp}, \mathsf{U}))$. Given the public parameters pp and a user's identity U, within Keygen phase, private key of U is generated by KGC. First, user's identity U is hashed as a public element $u \leftarrow \mathcal{H}_1(\mathsf{U}) \in \mathbb{Z}_r^*$. Then, KGC computes the user's private keys as

$$\mathsf{sk}_{\mathsf{U}} \leftarrow (\mathsf{sk}_{\mathsf{U}}^{\mathsf{P}}, \mathsf{sk}_{\mathsf{U}}^{\mathsf{Q}}) = ([(s+u)^{-1}]P, [(s+u)^{-1}]Q) \tag{2.18}$$

where the inverses are computed modulo r. The corresponding public keys of the user can be publicly computed from U and $\mathsf{pk}_{\mathsf{KGC}}$ as

$$\begin{aligned}
\mathsf{pk}_{\mathsf{U}} &\leftarrow (\mathsf{pk}_{\mathsf{U}}^{\mathsf{P}}, \mathsf{pk}_{\mathsf{U}}^{\mathsf{Q}}) && (2.19)\\
&= ([s]P + [\mathcal{H}_1(\mathsf{U})]P, [s]Q + [\mathcal{H}_1(\mathsf{U})]Q) && (2.20)\\
&= ([s]P + [u]P, [s]Q + [u]Q) && (2.21)\\
&= ([s+u]P, [s+u]Q)\,. && (2.22)
\end{aligned}$$

- Signcrypt $(\sigma_{\mathsf{UV}} \leftarrow \mathcal{S}(\mathsf{sk}_{\mathsf{U}}, m, \mathsf{V}))$. To signcrypt a message $m \in \{0,1\}^*$ to Bob-with identity V-, Alice-with identity U- generates a random integer $x \leftarrow_{\$} \mathbb{Z}_r^*$ and computes:

$$\begin{aligned}
R &\leftarrow g^x && (2.23)\\
c &\leftarrow m \oplus \mathcal{H}_3(R) && (2.24)\\
h &\leftarrow \mathcal{H}_2(m, R) && (2.25)\\
S &\leftarrow [x+h]\mathsf{sk}_{\mathsf{U}}^{\mathsf{P}} && (2.26)\\
T &\leftarrow [x]\mathsf{pk}_{\mathsf{V}}^{\mathsf{P}} && (2.27)
\end{aligned}$$

The signcrypted message from Alice to Bob is $\sigma_{\mathsf{UV}} \leftarrow (c, S, T)$.

- Unsigncrypt $(\mathcal{V}(\mathsf{sk}_{\mathsf{V}}, \sigma_{\mathsf{UV}}, \mathsf{U}))$. Given the signcrypted message σ_{UV}, Bob

computes

$$R \leftarrow e(T, \mathsf{sk}_{\mathsf{V}}^{\mathsf{Q}}) \tag{2.28}$$

$$m \leftarrow c \oplus \mathcal{H}_3(R) \tag{2.29}$$

$$h \leftarrow \mathcal{H}_2(m, R) \tag{2.30}$$

$$W \leftarrow e(S, \mathsf{pk}_{\mathsf{U}}^{\mathsf{Q}}) \tag{2.31}$$

and verifies that

$$W \stackrel{?}{=} Rg^h. \tag{2.32}$$

If the verification holds, returns the message m, otherwise outputs an error $\perp$.

Whenever the signer follows this scheme as supposed to, the following and hence correctness holds as expected.

$$R = e(T, \mathsf{sk}_{\mathsf{V}}^{\mathsf{Q}}) = e([x(s+v)]P, [(s+v)^{-1}]Q \tag{2.33}$$

$$= e(P, Q)^x = g^x\,, \tag{2.34}$$

$$W = e(S, \mathsf{pk}_{\mathsf{U}}^{\mathsf{Q}}) = e([(x+h)(s+u)^{-1}]P, [s+u]Q) \tag{2.35}$$

$$= e(P, Q)^{x+h} = Rg^h\,. \tag{2.36}$$

As it can be seen easily, Unsigncrypt step is not publicly verifiable because the signature computation depends on R, that can only be recovered by the legitimate receiver. However, if the the legitimate receiver cooperates and shares σ_{UV} and R, anyone can successfully run Unsigncrypt. Obviously, in this case this would also result in leaking the message itself.

Signcryption schemes naturally involve both encryption and signature procedures. Based on this, Malone-Lee [2002] stated two security notions separately, following the *de facto* security models by Rackoff and Simon [1992]; Bellare et al.

[1998] for public key encryption and by Goldwasser et al. [1988] for signature schemes.

Security model definitions for identity-based signcryption schemes is constructed with two parts, namely indistinguishability of identity-based signcryptions under chosen ciphertext attack (IND-IBSC-CCA) for encryption and existentially signature-unforgeability under adaptive chosen messages and ciphertexts attacks (ESUF-IBSC-CMA), separately. Based on these definitions, Barreto et al. [2005] also shows that security of the given signcryption scheme satisfies both IND-IBSC-CCA and ESUF-IDSC-CMA properties under the assumption of q-Bilinear Diffie-Hellman Inversion Problem and q-Strong Diffie-Hellman Problem are intractable, respectively. Boyen [2003] formalizes security definitions for multi-purpose signcryption schemes, based on the message confidentiality, signature non-repudiation, ciphertext unlinkability, ciphertext authentication and ciphertext anonymity properties. We now recall three of these definitions following the notations of McCullagh and Barreto [2004]; Barreto et al. [2005]; Boyen [2003] on which the eChb scheme's security reductions will be built. For further details on security formalization, reader may also refer to Boyen [2003]; Wang et al. [2013].

Definition 2.5: An identity-based signcryption scheme (IBSC) has the indistinguishability against adaptive chosen ciphertext attacks property (IND-IBSC-CCA) if no polynomially bounded adversary $\mathcal{A}$ has a non-negligible advantage in the following game.

i) The challenger $\mathcal{C}$ runs the Setup algorithm with a security parameter κ and sends the system parameters pp *to the adversary $\mathcal{A}$.*

ii) Find Phase: In this phase, $\mathcal{A}$ adaptively performs a polynomially bounded number of queries to the following oracles:

- *Keygen: returns private keys associated to arbitrary identities.*
- *Signcrypt: given (*U, V, m*) as input with a pair of identities* U, V *(presumably sender's and receiver's, respectively) and a plaintext m, it returns an encryption under the receiver's identity* V *of the message m in the name of the sender's identity* U.
- *Unsigncrypt: given (*σ, U, V*) as input with a pair of identities* U, V *and a ciphertext σ, it generates the receiver's private key* sk_{V} *and returns either a valid message-signature pair* $(m, (h, S))$ *for the sender's identity*

V *or the* $\perp$ *if under the private key* sk_{V}*,* σ *does not decrypt into a valid message-signature pair.*

iii) $\mathcal{A}$ *chooses two plaintexts* m_0*,* m_1*, and identities* U^**,* V^**. She may not have queried the private key of* V^* *and she obtains* $c = Signcrypt(m_b, \mathsf{sk}_{\mathsf{U}^*}, \mathsf{V}^*)$ *under system public parameters* pp*, for a random bit* $b \leftarrow_{\$} \{0, 1\}$*.*

iv) Guess Phase: $\mathcal{A}$ *asks new queries as in Find Phase, however she may not issue a key extraction request on* V^* *and cannot submit* c *to Unsigncrypt oracle for target identity* V^**.*

v) $\mathcal{A}$ *outputs a bit* b' *and wins if* $b' = b$*.*

Then, adversary $\mathcal{A}$*'s advantage is defined to be*

$$Adv(\mathcal{A}) = |\textit{Prob}(b' = b) - \frac{1}{2}| \ , \tag{2.37}$$

hence an IBSC has possesses the IND-IBSC-CCA property only if

$$Adv(\mathcal{A}) = |\textit{Prob}(b' = b) - \frac{1}{2}| \leq \mathsf{negl}(\kappa) \ . \tag{2.38}$$

Definition 2.6: An identity-based signcryption scheme is said to be existentially signature-unforgeable for adaptive chosen messages and ciphertext attacks (ESUF-IBSC-CMA) if no polynomially bounded adversary has a non-negligible advantage in the following game.

i) The challenger $\mathcal{C}$ *runs the Setup algorithm with a security parameter* κ *and gives the system parameters* pp *to the adversary* $\mathcal{A}$*.*

ii) $\mathcal{A}$ *performs a polynomially bounded number of requests as in the Definition 2.5.*

iii) Finally, $\mathcal{A}$ *produces a triple* $(\sigma^*, \mathsf{U}^*, \mathsf{V}^*)$ *and wins the game*

- *if the sender's identity* U^* *was not corrupted and*
- *if the result of Unsigncrypt oracle on* σ^* *under the private key associated to* V^* *is a valid message-signature pair* $(m^*, (h^*, S^*))$ *such that no Signcrypt query*
  - *involved* m^**,* U^* *and some receiver* V' *(possibly different from* V^**) and*
  - *resulted in a ciphertext* σ' *whose decryption under the private key* $\mathsf{sk}_{\mathsf{U}}{}'$ *is alleged forgery*

$$(m^*, (h^*, S^*), \mathsf{U}^*).$$

Then adversary $\mathcal{A}$'s advantage is $Adv(\mathcal{A}) = |Prob(\mathcal{A}\ wins)|$. Thus an IBSC has possesses the ESUF-IBSC-CMA property only if

$$Adv(\mathcal{A}) = |Prob(\mathcal{A}\ wins)| \leq \mathsf{negl}(\kappa)\ . \tag{2.39}$$

Definition 2.7: An identity-based signcryption scheme is said to be ciphertext anonymous against adaptive chosen-ciphertext insider attacks, or (ANON-IBSC-CCA) secure, if no polynomially bounded adversary $\mathcal{A}$ has a non-negligible advantage in the following game.

i) The challenger $\mathcal{C}$ runs the Setup algorithm with a security parameter $\kappa$and provides the public parameters pp to the adversary $\mathcal{A}$.

ii) Find Phase: $\mathcal{A}$ performs a polynomially bounded number of requests as in the Definition 2.5.

iii) $\mathcal{A}$ chooses two sender identities $\mathsf{U}_1, \mathsf{U}_2$ and two recipient identities $\mathsf{V}_1, \mathsf{V}_2$ along with a message m.

iv) $\mathcal{C}$ flips two random coins $b_1, b_2 \in \{0, 1\}$ and gives $c = Signcrypt(m, \mathsf{sk}_{\mathsf{U}_{b_1}}, \mathsf{V}_{b_2})$ to $\mathcal{A}$.

v) Guess Phase: $\mathcal{A}$ performs new queries as in Find Phase, however she may not request key extraction on neither V_{b_1} nor V_{b_2} and cannot submit c to Unsigncrypt oracle.

vi) Finally, $\mathcal{A}$ outputs (b'_1, b'_2) and wins if $(b'_1, b'_2) = (b_1, b_2)$.

Then, adversary $\mathcal{A}$'s advantage is defined to be

$$Adv(\mathcal{A}) = |Prob((b'_1, b'_2) = (b_1, b_2)) - \frac{1}{4}|. \tag{2.40}$$

Similarly, an IBSC has the ANON-IBSC-CCA property only if

$$Adv(\mathcal{A}) = |Prob((b'_1, b'_2) = (b_1, b_2)) - \frac{1}{4}| \leq \mathsf{negl}(\kappa). \tag{2.41}$$

3. ELECTRONIC CHECKBOOK

In a traditional paper check system, when a *payer* chooses to pay with a paper check, the check is signed with the recipient's name, date, and desired amount (*face value*). Naturally, these checks are bundled in a checkbook, a collection of blank checks issued by the payer's bank. The number of checks in a checkbook varies, but a checkbook usually has 10, 20, or as many as 100 checks. An electronic check (e-check) is the electronic version of a paper check.

The idea of e-checks was introduced by Chaum et al. [1990a], where they also provide an offline e-check system. However, the proposed solution has a very high computational complexity. An enhanced version is provided by Chaum et al. [1990b], but the amount must be determined before the electronic check is issued. Then, another offline electronic control mechanism was proposed by Brands [1993] based on the representation problem and claimed to be more efficient than Chaum et al. [1990a] and Chaum et al. [1990b].

An analysis and comparison of initial propositions for electronic payment systems put forward by Yu et al. [2002]. Kim and Oh [2002] proposed a scheme based on partially blind RSA-based signatures and one-way accumulators, but again the amount needs to be determined before e-check issuance. In 2005, Chen [2005] proposed a mechanism where the amount no longer needs to be determined before the e-check was issued, but instead embedded in the e-check. Hinarejos et al. [2012] suggested a solution satisfying anonymity and transferability. Later, security enhanced version of 3D-Secure protocol is given by Plateaux et al. [2013]. Again, these systems require the issuance of an e-check for each payment.

The aforementioned mechanisms require the payer to interact with the issuing bank for each electronic check issuance and are therefore not considered as electronic checkbook mechanisms. To the best of the authors' knowledge, the first attempt at an e-checkbook system was presented by Pasupathinathan et al. [2005]. At the end of the issuance period, the payer receives an e-checkbook with different Schnorr signatures for each e-check. Although the payer no longer has to interact with the issuing bank for each e-check payment, the computational and storage complexity is linear with the number of e-checks. Then, people studied achieving constant computational and

storage complexity. Four e-checkbook schemes were proposed based on Chen [2005] 's scheme and all four Chen et al. [2009]; Chang et al. [2009]; Chen et al. [2010]; Chang et al. [2016] satisfy constant computational complexity and storage property. However, all of these schemes have flows as shown by Sertkaya and Kalkar [2020]:

- Pasupathinathan et al. [2005]'s proposal does not satisfy the correctness, anonymous identity and payment unlinkability properties,
- Chen et al. [2009]'s scheme is not secure against e-check manipulation and e-check forgery attacks,
- Chang et al. [2009]'s protocol is susceptible to e-check manipulation attack,
- Chen et al. [2010]'s proposal is vulnerable against e-check manipulation attack,
- Chang et al. [2016]'s scheme is susceptible to e-check manipulation attack.

With the exception of Sertkaya and Kalkar [2019, 2021], there are no new e-checkbook propositions. Sertkaya and Kalkar [2019] does not support transferability and meet anonymity property, i.e. eavesdroppers naturally derive payers and payees from the information on e-checks. In Sertkaya and Kalkar [2021], Sertkaya and Kalkar propose an e-checkbook scheme that supports transferable electronic checks and meets anonymity against eavesdroppers. More specifically, they first provide game-based security definitions for e-checkbook unforgeability, e-check unforgeability and non-manipulability, and e-check anonymity. After explaining the details of the proposed scheme, they prove that their scheme meets these properties along with resistance to double payments and replay attacks.

Notation. We use the following notation for the upcoming subsections in addition to the abbreviations.

- Ω_{U}: e-checkbook of U,
- ω_{UV}: e-check with payer U and payee V,
- d: date,
- a: amount,
- σ_{UV}: signcryption of a value by U for V,
- R_{BU}: auxiliary value required for signcryption verification by entities other than the recipient.

3.1. Electronic Checkbook Scheme

An e-checkbook scheme involves four entities:

- Issuer: The bank who issues e-checkbook for its customers/users, performs the actual e-check settlement, and makes the money transfers.
- Payer: An issuing bank customer who wishes to obtain an e-checkbook and make payments by e-checks.
- Payee: An entity that receives an e-check from a payer. Upon receipt of an e-check, payee performs the necessary checks and requests payment of the e-check through her own bank.
- Acquirer: The bank maintaining the payee's bank account.

It is assumed that inter-bank transactions are handled using existing conventional mechanisms. Therefore, for the sake of simplicity, it is further assumed that the Issuer and the Acquirer banks are the same and are denoted by B.

$\mathrm{eChb} = (\mathcal{G}, \mathcal{K}, \mathcal{I}, \mathcal{P}, \mathcal{T}, \mathcal{D})$ is a transferable e-checkbook scheme consisting of the following protocols.

- Setup $\mathcal{G}$ - Given security parameter κ, system environment is constructed and system-wide public parameters are generated and published.
- Keygen $\mathcal{K}$ - On input of the public parameters pp and an identity U, it generates private and public key pair for the entity U.
- $\mathrm{Issuance}$ $\mathcal{I}$ - User U and the bank B follows the e-checkbook issuance protocol and U gets the e-checkbook Ω_{U}.
- $\mathrm{Payment}$ $\mathcal{P}$ - Based on an agreement on the date d and amount a with the payee V, the payer U creates a signed e-check ω_{UV}, sends it to V. V authenticates the issuance of the e-check.
- $\mathrm{Transfer}$ $\mathcal{T}$ - In this optional protocol, instead of cashing out the received e-check, the payee V applies to the bank B to make the e-check transferable. After B makes the check transferable, V creates an e-check ω_{VY} and sends it to the payee Y.
- $\mathrm{Deposit}$ $\mathcal{D}$ - Upon receiving a cash out order for an e-check ω, the bank B first authenticates e-check issuance and checks against double spending, only then deducts the amount from the account of the first issuer of the e-check and adds it to the payee.

3.1.1. Requirements

Just like in the paper check system, banks are assumed to be honest and follow the protocol. So, we assume there is no attack from the bank B. However, both the payer U and the payee V, are thought to be malicious. So, they will try to deviate from the protocol as much as possible.

Known attacks that can be pursued by payers, payees or eavesdroppers against an e-checkbook scheme can be summarized as below.

- E-checkbook forgery: unauthorized creation of a verifiable e-checkbook as if it is issued by B,
- E-check forgery: unauthorized creation of a verifiable new e-check as if it is spent by U,
- E-check manipulation: manipulation (changing the payee, the amount or the date) of a transmitted e-check,
- Double spending: paying with the same indexed e-check more than once, probably with different payee, amount or date,
- Replay attack: depositing same e-check more than once.

To combat these attacks and ensure user privacy, any e-checkbook system must meet the following properties.

- E-checkbook validation: E-checkbook is issued by B for U,
- E-check validation: E-check belongs to an e-checkbook issued for U by B,
- E-check integrity: E-check has not been manipulated since it was created.
- E-check source authentication: E-check is issued and signed by the dedicated owner U.
- E-check anonymity: E-check appears anonymous to anyone who does not possess the payee's secret key, i.e. hides both the payer's and the payee's identities and other than the payer, the payee and the bank, no one can learn the e-check.

3.1.2. Security Definitions

The following definitions follow the security definitions defined for the signcryption schemes that are included in Section 2.5.3.1..

Definition 3.1: An e-checkbook scheme is said to possess the e-checkbook unforgeability property if no polynomially bounded adversary $\mathcal{A}$ has a non-negligible advantage in the following game.

i) The challenger $\mathcal{C}$ creates and gives the system parameters pp *to the adversary $\mathcal{A}$.*

ii) $\mathcal{A}$ performs polynomially bounded number of queries to the $\mathcal{C}$ to get:

- *the public-private key pair associated to an arbitrary identity* U
- *the e-checkbook* Ω_{U} *for* U

iii) Finally, $\mathcal{A}$ produces an e-checkbook Ω_{U^*} and wins the game if

- *the bank's identity* B *was not corrupted and*
- Ω_{U^*} *is a valid e-checkbook that was not queried.*

The adversary's advantage is defined to be

$$Adv(\mathcal{A}) = |Prob(\mathcal{A})\ wins|. \tag{3.1}$$

Definition 3.2: (E-check unforgeability and non-manipulability) An e-checkbook scheme has the e-check unforgeability and non-manipulability property if no polynomially bounded adversary $\mathcal{A}$ has a non-negligible advantage in the following game.

i) The challenger $\mathcal{C}$ creates and gives the system parameters pp *to the adversary $\mathcal{A}$.*

ii) $\mathcal{A}$ performs a polynomially bounded number of queries to the $\mathcal{C}$ to get:

- *the public-private key pair associated to an arbitrary identity* U
- *the e-checkbook* Ω_{U} *for an arbitrary identity* U *with* α, k.
- *the e-check* ω_{UV} *from* U *to* V *for arbitrary identities* U *and* V *along with desired date, amount, e-check index.*

iii) Finally, $\mathcal{A}$ produces an e-check $\omega_{\mathsf{U}^*\mathsf{V}^*}$ and wins the game if

- *the payer's identity* U^* *was not corrupted and*
- $\omega_{\mathsf{U}^*\mathsf{V}^*}$ *is a valid e-check that was not queried.*

The adversary's advantage is defined to be

$$Adv(\mathcal{A}) = |Prob(\mathcal{A})\ wins|. \tag{3.2}$$

Remark: Note that, in order to successfully execute e-check forgery or e-check manipulation attacks, a malicious entity must necessarily generate valid e-checks. Therefore, Definition 3.2 simulates both of these attack types.

Definition 3.3: An e-checkbook scheme has the e-check anonymity property if no polynomially bounded adversary $\mathcal{A}$ has a non-negligible advantage in the following game.

i) The challenger $\mathcal{C}$ creates and gives the system parameters pp *to the adversary $\mathcal{A}$.*

ii) $\mathcal{A}$ performs a polynomially bounded number of queries to the $\mathcal{C}$ to get:

- *the public-private key pair associated to an arbitrary identity* U
- *the e-checkbook* Ω_{U} *for an arbitrary identity* U.
- *the e-check* ω_{UV} *from* U *to* V *for arbitrary identities* U *and* V.

iii) Eventually, $\mathcal{A}$ submits two sender identities $\mathsf{U}_1, \mathsf{U}_2$ *and two recipient identities* $\mathsf{V}_1, \mathsf{V}_2$ *(along with desired date, amount, e-check index) to $\mathcal{C}$.*

iv) $\mathcal{C}$ selects two bits b_1, b_2 *uniformly at random and sends the "challenge" e-check* $\omega_{\mathsf{U}_{b_1}\mathsf{V}_{b_2}}$ *to $\mathcal{A}$.*

v) $\mathcal{A}$ is free to perform a polynomially bounded number of queries to the $\mathcal{C}$ to get

- *the public-private key pair associated to an arbitrary identity* U
- *the e-checkbook* Ω_{U} *of* U.
- *the e-check* ω_{UV} *from* U *to* V *for an arbitrary identity* V *along with date, amount, e-check index.*

under the additional constraint that $\mathcal{A}$ can not ask for the private key of V_1 *and* V_2.

vi) Finally, $\mathcal{A}$ outputs a guess (b_1^*, b_2^*) *and wins the game if*

- *the identities of* $\mathsf{V}_1, \mathsf{V}_2$ *are not corrupted*
- $(b_1^*, b_2^*) = (b_1, b_2)$.

The adversary's advantage is defined to be

$$Adv(\mathcal{A}) = |\textit{Prob}((b_1^*, b_2^*) = (b_1, b_2)) - \frac{1}{4}|. \tag{3.3}$$

3.2. Proposed Scheme

Our e-checkbook scheme eChb consists of six phases; namely Setup $\mathcal{G}$, Keygen $\mathcal{K}$, Issuance $\mathcal{I}$, Payment $\mathcal{P}$, Transfer $\mathcal{T}$ and Deposit $\mathcal{D}$ as defined in Section 3.1.. Setup $\mathcal{G}$ generates system-wide public parameters and publishes them. Keygen $\mathcal{K}$ is used to generate and distribute public/private key pairs for the related entities. Issuance $\mathcal{I}$ describes the process for a registered user of the bank to receive an e-checkbook from the bank. Whenever a registered user with an e-checkbook Ω wants to issue an e-check ω, she follows the Payment $\mathcal{P}$. Anytime a payee receives an e-check ω, she makes it transferable so that she can use for another payment or cashes out the e-check. Here, without loss of generality, making an electronic check transferable and paying with a transferable e-check is described in the Transfer $\mathcal{T}$ phase. Finally, Deposit $\mathcal{D}$ phase is the protocol between the payee and the bank, where, after the necessary checks, an electronic check is transferred from the payer's account to the payee's account.

An e-check ω should include payee's identity V, the date value d, and the amount value a along with the e-checkbook page index i and the source authenticator hash chain value $\mathcal{H}^{r-i}(\alpha)$, in signcrypted form.

Our eChb scheme uses the signcryption scheme described in 2.5.3.1.. This choice can be changed with any cryptographically secure signcryption scheme that has ESUF-IBSC-CMA and ANON-IBSC-CCA properties.

More specifically, the details of each phases of eChb scheme are as follows.

Setup Phase

Given a security parameter κ, Key generation Center (KGC) runs the SignC's setup phase which constructs a bilinear environment, uniformly selects KGC's master key $\mathsf{sk}_{\mathsf{KGC}} = s$, and publishes the public parameters pp as

$$\{r, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, P, Q, e, \mathcal{H}, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3, g, \mathsf{pk}_{\mathsf{KGC}}\},$$

where $g = e(P, Q)$ and $\mathsf{pk}_{\mathsf{KGC}} = ([s]P, [s]Q)$.

Keygen Phase

For any identity U, KGC;

i) hashes its identity to a public element $u \leftarrow \mathcal{H}_1(\mathsf{U}) \in \mathbb{Z}_r^*$.

ii) computes the user's private keys

$$\mathsf{sk}_\mathsf{U} \leftarrow (\mathsf{sk}_\mathsf{U}^\mathsf{P}, \mathsf{sk}_\mathsf{U}^\mathsf{Q}) = ([(s+u)^{-1}]P, [(s+u)^{-1}]Q). \tag{3.4}$$

Corresponding public keys can be publicly computed from u and pk_KGC as

$$\mathsf{pk}_\mathsf{U} \leftarrow (\mathsf{pk}_\mathsf{U}^\mathsf{P}, \mathsf{pk}_\mathsf{U}^\mathsf{Q}) = ([s+u]P, [s+u]Q). \tag{3.5}$$

Issuance Phase

A payer U first registers with the bank B, then opens a check account and acquires an e-checkbook with k e-checks as follows.

i) U selects $\alpha \leftarrow_\$ \{0,1\}^\kappa$ uniformly at random.

ii) U computes $m_\mathsf{UB} = \mathsf{U}||k||\mathcal{H}^k(\alpha)$ and signcrypts m_UB as

$$\sigma_\mathsf{UB} \leftarrow (c_\mathsf{UB}, S_\mathsf{UB}, T_\mathsf{UB}) = \mathcal{S}(\mathsf{sk}_\mathsf{U}, m_\mathsf{UB}, \mathsf{B})\,. \tag{3.6}$$

iii) U sends σ_UB to B for recording and signing.

iv) B runs $\mathcal{V}(\mathsf{sk}_\mathsf{B}, \sigma_\mathsf{UB}, \mathsf{U})$ to verify the signature, extracts $m_\mathsf{UB} = \mathsf{U}||k||\mathcal{H}^k(\alpha)$, records k and $\mathcal{H}^k(\alpha)$ within the U's bank account information.

v) B constructs $m_\mathsf{BU} = \mathsf{U}||\mathcal{H}^k(\alpha)$ and signcrypts m_BU as

$$\sigma_\mathsf{BU} \leftarrow (c_\mathsf{BU}, S_\mathsf{BU}, T_\mathsf{BU}) = \mathcal{S}(\mathsf{sk}_\mathsf{B}, m_\mathsf{BU}, \mathsf{U})\,. \tag{3.7}$$

vi) B sends σ_BU to U for e-checkbook creation acknowledgement and verification.

vii) U runs $\mathcal{V}(\mathsf{sk}_\mathsf{U}, \sigma_\mathsf{BU}, \mathsf{B})$ to verify the signature, extracts $m_\mathsf{BU} = \mathsf{U}||\mathcal{H}^k(\alpha)$, checks $\mathcal{H}^k(\alpha)$.

viii) If all of the controls hold, U records Ω_{U} as his e-checkbook and keeps it secret, i.e. $\Omega_{\mathsf{U}} \leftarrow (\alpha, k, \sigma_{\mathsf{BU}}, R_{\mathsf{BU}})$, where R_{BU} is the auxiliary value computed by U while unsigncrypting σ_{BU}.

Payment Phase

Assume that U has already used $i-1$ ($i < k$) e-checks from his e-checkbook Ω_{U}. In order to create an e-check as a payment to V with face value a and date d as the i-th e-check ω^i_{UV}, U and V follow steps below.

i) U computes

$$m^i_{\mathsf{UV}} \leftarrow \mathsf{d}||\mathsf{a}||i||\mathcal{H}^{k-i}(\alpha)||\mathsf{U}||\mathsf{V}||\sigma_{\mathsf{BU}}||R_{\mathsf{BU}}, \tag{3.8}$$

$$\omega^i_{\mathsf{UV}} \leftarrow (c_{\mathsf{UV}}, S_{\mathsf{UV}}, T_{\mathsf{UV}}) = \mathcal{S}(\mathsf{sk}_{\mathsf{U}}, m^i_{\mathsf{UV}}, \mathsf{V}). \tag{3.9}$$

ii) U sends ω^i_{UV} to V.

iii) V runs $\mathcal{V}(\mathsf{sk}_{\mathsf{V}}, \omega^i_{\mathsf{UV}}, \mathsf{U})$, verifies the signature, extracts

$$m^i_{\mathsf{UV}} \leftarrow \mathsf{d}||\mathsf{a}||i||\mathcal{H}^{k-i}(\alpha)||\mathsf{U}||\mathsf{V}||\sigma_{\mathsf{BU}}||R_{\mathsf{BU}}\,. \tag{3.10}$$

iv) V verifies her identity, the amount a and the date d is correct; computes $\mathcal{H}^i(\mathcal{H}^{k-i}(\alpha))$, constructs $m_{\mathsf{BU}} \leftarrow \mathsf{U}||\mathcal{H}^k(\alpha)$, and by using R_{BU} and m_{BU}, verifies the B's signature σ_{BU}.

Transfer Phase

Whenever a payee receives an e-check, she has two choices, namely transfer and deposit, where both are send to the bank B. In the sequel, this choice is denoted by the Boolean flag t, that is prefixed to plain order message.

Assume that V received an e-check ω^i_{UV} from U and wants to make a payment with it to another payee Y.

i) V sets the flag t to "1" for indicating transfer order, and computes

$$m^i_{\mathsf{VB}} \leftarrow t||\mathsf{d}||\mathsf{a}||i||\mathcal{H}^{k-i}(\alpha)||\mathsf{U}||\mathsf{V}||\sigma_{\mathsf{BU}}||R_{\mathsf{BU}}|| \tag{3.11}$$

$$\omega^i_{\mathsf{UV}}||R^i_{\mathsf{UV}}, \tag{3.12}$$

$$\sigma^i_{\mathsf{VB}} \leftarrow (c_{\mathsf{VB}}, S_{\mathsf{VB}}, T_{\mathsf{VB}}) = \mathcal{S}(\mathsf{sk}_{\mathsf{V}}, m^i_{\mathsf{VB}}, \mathsf{B}), \tag{3.13}$$

where R^i_{UV} is the auxiliary value computed by V while unsigncrypting ω^i_{UV}.

ii) V sends σ^i_{VB} to B for double spending control and transfer order settlement.

iii) B runs $\mathcal{V}(\mathsf{sk}_{\mathsf{B}}, \sigma^i_{\mathsf{VB}}, \mathsf{V})$ to verify the signature, extracts

$$m^i_{\mathsf{VB}} \leftarrow t||\mathsf{d}||\mathsf{a}||i||\mathcal{H}^{k-i}(\alpha)||\mathsf{U}||\mathsf{V}||\sigma_{\mathsf{BU}}||R_{\mathsf{BU}}|| \tag{3.14}$$

$$\omega^i_{\mathsf{UV}}||R^i_{\mathsf{UV}}\,. \tag{3.15}$$

iv) B constructs $m^i_{\mathsf{UV}} \leftarrow \mathsf{d}||\mathsf{a}||i||\mathcal{H}^{k-i}(\alpha)||\mathsf{U}||\mathsf{V}||\sigma_{\mathsf{BU}}||R_{\mathsf{BU}}$, and by using R^i_{UV} and m^i_{UV} verifies the e-check ω^i_{UV}.

v) By computing $\mathcal{H}^i(\mathcal{H}^{k-i}(\alpha))$ and using R_{BU}, B verifies the signature σ_{BU}.

vi) B assures that ω^i_{UV} was not already spent or transferred.

vii) B computes

$$m^i_{\mathsf{BV}} \leftarrow \mathsf{d}||\mathsf{a}||i||\mathcal{H}^{k-i}(\alpha)||\mathsf{U}||\mathsf{V} \tag{3.16}$$

$$\sigma^i_{\mathsf{BV}} \leftarrow (c_{\mathsf{BV}}, S_{\mathsf{BV}}, T_{\mathsf{BV}}) = \mathcal{S}(\mathsf{sk}_{\mathsf{B}}, m^i_{\mathsf{BV}}, \mathsf{V}), \tag{3.17}$$

and records $(i, \mathcal{H}^{k-i}(\alpha), \mathsf{a}, \mathsf{d}, \omega^i_{\mathsf{UV}}, \sigma^i_{\mathsf{VB}})$ as transferred check.

viii) B sends σ^i_{BV} to V.

ix) V runs $\mathcal{V}(\mathsf{sk}_{\mathsf{V}}, \sigma^i_{\mathsf{BV}}, \mathsf{B})$, verifies the signature, extracts

$$m^i_{\mathsf{BV}} \leftarrow \mathsf{d}||\mathsf{a}||i||\mathcal{H}^{k-i}(\alpha)||\mathsf{U}||\mathsf{V}\,. \tag{3.18}$$

x) In order to make a payment to Y with the transferred e-check, V computes

$$m^i_{\mathsf{VY}} \leftarrow \mathsf{d}||\mathsf{a}||i||\mathcal{H}^{k-i}(\alpha)||\mathsf{U}||\mathsf{V}||\mathsf{Y}||\sigma^i_{\mathsf{BV}}||R^i_{\mathsf{BV}} \tag{3.19}$$

$$\omega^i_{\mathsf{VY}} \leftarrow (c_{\mathsf{VY}}, S_{\mathsf{VY}}, T_{\mathsf{VY}}) = \mathcal{S}(\mathsf{sk}_{\mathsf{V}}, m^i_{\mathsf{VY}}, \mathsf{Y}). \tag{3.20}$$

xi) V sends ω^i_{VY} to Y.

xii) Y runs $\mathcal{V}(\mathsf{sk}_{\mathsf{Y}}, \omega^i_{\mathsf{VY}}, \mathsf{V})$ to verify the signature, extracts

$$m^i_{\mathsf{VY}} \leftarrow \mathsf{d}||\mathsf{a}||i||\mathcal{H}^{k-i}(\alpha)||\mathsf{U}||\mathsf{V}||\mathsf{Y}||\sigma^i_{\mathsf{BV}}||R^i_{\mathsf{BV}}\,. \tag{3.21}$$

xiii) Y verifies her identity, the amount a and the date d is correct; constructs

$$m^i_{\mathsf{BV}} \leftarrow \mathsf{d}||\mathsf{a}||i||\mathcal{H}^{k-i}(\alpha)||\mathsf{U}||\mathsf{V}\,, \tag{3.22}$$

and by using R^i_{BV} and m^i_{BV}, verifies the B's signature σ^i_{BV}.

The e-check that is ordered to be made transferable in the Transfer Phase may have already been transferred, i.e. the e-checkbook owner U makes a payment with to U_1 with the e-check $\omega^i_{\mathsf{UU}_1}$, U_1 passes it to U_2 as $\omega^i_{\mathsf{U}_1\mathsf{U}_2}$, and so on, till U_l transfers it to V. Eventually, V transfers it to Y as ω^i_{VY}. In this payment traverse, the bank B always includes the owner U of the e-checkbook and the owner of the last transfer order in the text to be signcrypted. If required by the law, all involved users may be included.

Each time a payee Y receives an e-check ω^i_{VY}, she can deduce if this e-check belongs to the e-checkbook owner V or from a previous payee V at the end of the Payment or Transfer phase.

Deposit Phase

Suppose that a payee Y wants to deposit ω^i_{VY}.

- If V is the owner of the e-checkbook:

  i) Y sets the flag t to "0" for indicating the deposit order, computes

$$m^i_{\mathsf{YB}} \leftarrow t||\mathsf{d}||\mathsf{a}||i||\mathcal{H}^{k-i}(\alpha)||\mathsf{V}||\mathsf{Y}||\sigma_{\mathsf{BV}}||R_{\mathsf{BV}}|| \tag{3.23}$$

$$\omega^i_{\mathsf{VY}}||R^i_{\mathsf{VY}}\,, \tag{3.24}$$

$$\sigma^i_{\mathsf{YB}} \leftarrow (c_{\mathsf{YB}}, S_{\mathsf{YB}}, T_{\mathsf{YB}}) = \mathcal{S}(\mathsf{sk}_{\mathsf{Y}}, m^i_{\mathsf{YB}}, \mathsf{B})\,. \tag{3.25}$$

ii) Y sends σ^i_{YB} to B for double spending control and deposit order settlement.

iii) B runs $\mathcal{V}(\mathsf{sk}_{\mathsf{B}}, \sigma^i_{\mathsf{YB}}, \mathsf{Y})$, verifies the signature, extracts

$$m^i_{\mathsf{YB}} \leftarrow t||\mathsf{d}||\mathsf{a}||i||\mathcal{H}^{k-i}(\alpha)||\mathsf{V}||\mathsf{Y}||\sigma_{\mathsf{BV}}||R_{\mathsf{BV}}|| \tag{3.26}$$

$$\omega^i_{\mathsf{VY}}||R^i_{\mathsf{VY}}. \tag{3.27}$$

iv) B constructs

$$m^i_{\mathsf{VY}} \leftarrow \mathsf{d}||\mathsf{a}||i||\mathcal{H}^{k-i}(\alpha)||\mathsf{V}||\mathsf{Y}||\sigma_{\mathsf{BV}}||R_{\mathsf{BV}}\,, \tag{3.28}$$

and by using R^i_{VY} and m^i_{VY} verifies the e-check ω^i_{VY}.

v) B computes $\mathcal{H}^i(\mathcal{H}^{k-i}(\alpha))$, constructs

$$m_{\mathsf{BV}} \leftarrow \mathsf{V}||\mathcal{H}^k(\alpha)\,,$$

and by using R_{BV} and m_{BV}, verifies the B's signature σ_{BV}.

vi) B now assures that the e-check ω^i_{VY} was not already spent or transferred.

vii) Then, B records

$$(i, \mathcal{H}^{k-i}(\alpha), \mathsf{a}, \mathsf{d}, \omega^i_{\mathsf{VY}}, \sigma^i_{\mathsf{YB}}) \tag{3.29}$$

as spent check, deducts the amount a from V's account, adds it into the Y's account, and informs Y.

- Else (U is the e-checkbook owner):

i) Y sets the flag t to "0" for indicating the deposit order, computes

$$m^i_{\mathsf{YB}} \leftarrow t||\mathsf{d}||\mathsf{a}||i||\mathcal{H}^{k-i}(\alpha)||\mathsf{U}||\mathsf{V}||\mathsf{Y}||\sigma^i_{\mathsf{BV}}||R^i_{\mathsf{BV}}|| \tag{3.30}$$

$$\omega^i_{\mathsf{VY}}||R^i_{\mathsf{VY}}\,, \tag{3.31}$$

$$\sigma^i_{\mathsf{YB}} \leftarrow (c_{\mathsf{YB}}, S_{\mathsf{YB}}, T_{\mathsf{YB}}) = \mathcal{S}(\mathsf{sk}_{\mathsf{Y}}, m^i_{\mathsf{YB}}, \mathsf{B})\,. \tag{3.32}$$

ii) Y sends σ^i_{YB} to B for double spending control and deposit order settlement.

iii) B runs $\mathcal{V}(\mathsf{sk}_{\mathsf{B}}, \sigma^i_{\mathsf{YB}}, \mathsf{Y})$, verifies the signature, extracts

$$m^i_{\mathsf{YB}} \leftarrow t||\mathsf{d}||\mathsf{a}||i||\mathcal{H}^{k-i}(\alpha)||\mathsf{U}||\mathsf{V}||\mathsf{Y}||\sigma^i_{\mathsf{BV}}||R^i_{\mathsf{BV}}|| \tag{3.33}$$

$$\omega^i_{\mathsf{VY}}||R^i_{\mathsf{VY}}. \tag{3.34}$$

iv) B constructs

$$m^i_{\mathsf{VY}} \leftarrow \mathsf{d}||\mathsf{a}||i||\mathcal{H}^{k-i}(\alpha)||\mathsf{U}||\mathsf{V}||\mathsf{Y}||\sigma^i_{\mathsf{BV}}||R^i_{\mathsf{BV}}\,, \tag{3.35}$$

and by using R^i_{VY} and m^i_{VY} verifies the e-check ω^i_{VY}.

v) B constructs

$$m^i_{\mathsf{BV}} \leftarrow \mathsf{d}||\mathsf{a}||i||\mathcal{H}^{k-i}(\alpha)||\mathsf{U}||\mathsf{V}\,, \tag{3.36}$$

and by using R^i_{BV} and m^i_{BV}, verifies the B's signature σ^i_{BV}.

vi) B now assures that the e-check ω^i_{VY} was not already spent or transferred.

vii) Then, B updates the record of

$$(i, \mathcal{H}^{k-i}(\alpha), \mathsf{a}, \mathsf{d}, \omega^i_{\mathsf{UV}}, \sigma^i_{\mathsf{VB}}) \tag{3.37}$$

with σ^i_{YB} as spent check, deducts the amount a from U's account, adds it into the Y's account, and informs Y.

If the payer and the payee honestly follow the protocol, the correctness of the eChb protocol naturally follows from correctness of the underlying signcryption scheme.

3.3. Security & Performance Analysis

Here, we are going to prove that our scheme satisfies e-checkbook unforgeability, e-check unforgeability and non-manipulability, and e-check anonymity properties. In addition to that, we are going to discuss the performance and security of our scheme with respect to the previously mentioned proposals.

Theorem 3.1: If the signcryption scheme SignC given in Section 2.5.3.1. is existentially signature-unforgeable for adaptive chosen messages and ciphertext attacks (ESUF-IBSC-CMA), then EChb scheme has the e-checkbook unforgeability property.

Proof . Let $\mathcal{A}$ be an adversary that can win the e-checkbook unforgeability game, given in Definition 3.1, with some non-negligible probability p. We are going to show that if such an adversary exists, then it can be used to construct an adversary $\mathcal{B}$ that can win the ESUF-IBSC-CMA game (see Definition 2.6) for SignC scheme.

i) The challenger $\mathcal{C}$ runs the Setup algorithm of SignC scheme with a security parameter κ and gives the system parameters pp to the adversary $\mathcal{B}$.

ii) $\mathcal{B}$ passes pp to the adversary $\mathcal{A}$.

iii) $\mathcal{A}$ performs a polynomially bounded number of queries to $\mathcal{B}$ to get:

- *the public-private key pair associated to an arbitrary identity U,*
- *the e-checkbook Ω_{U} of U with α, k.*

iv) For each query of $\mathcal{A}$, $\mathcal{B}$

- *queries Keygen oracle for the public-private key pair associated to U,*
- *queries Signcrypt oracle with $(\mathsf{B}, \mathsf{U}, m_{\mathsf{BU}})$ to get $\sigma_{\mathsf{BU}} = (c_{\mathsf{BU}}, S_{\mathsf{BU}}, T_{\mathsf{BU}})$ on $m_{\mathsf{BU}} = \mathsf{U}||\mathcal{H}^k(\alpha)$,*
- *computes $R_{\mathsf{BU}} = e(T_{\mathsf{BU}}, sk_{\mathsf{U}}^Q)$.*

v) B returns the following query results to $\mathcal{A}$:

- *the public-private key pair associated to U,*
- *the e-checkbook $\Omega_{\mathsf{U}} = (\alpha, k, \sigma_{\mathsf{BU}}, R_{\mathsf{BU}})$ of U.*

vi) Finally, $\mathcal{A}$ produces an e-checkbook $\Omega_{\mathsf{U}^*} = (\alpha^*, k^*, \sigma_{\mathsf{BU}^*}, R_{\mathsf{BU}^*})$, where

- *the bank's identity B was not corrupted,*

- Ω_{U^*} *is a valid e-checkbook that was not queried.*

vii) $\mathcal{B}$ *sends* $(\sigma_{\mathsf{BU}^*}, \mathsf{B}, \mathsf{U}^*)$ *to* $\mathcal{C}$ *and wins the ESUF-IBSC-CMA game with probability* p *since*

- *sender's identity* B *was not corrupted,*
- *the result of Unsigncrypt oracle on* σ_{BU^*} *under the private key associated to* U^* *is a valid message-signature pair* $(m^*, (h^*, S^*))$*, where* $m^* = \mathsf{U}^* || \mathcal{H}^{k^*}(\alpha^*)$ *such that no Signcrypt query*
  - *involved* m^**,* B *and some receiver* U' *(possibly different from* U^**) and*
  - *resulted in a ciphertext* σ' *whose decryption under the private key* $\mathsf{sk}_{\mathsf{U}}{}'$ *is alleged forgery* $(m^*, (h^*, S^*), \mathsf{U}^*)$.

However, as it was already proved, (see [Barreto et al., 2005, Theorem 3]), an adversary can only win the ESUF-IBSC-CMA game for SignC with negligible probability. Thus, EChb scheme has the e-checkbook unforgeability property.

■

Theorem 3.2: (E-check unforgeability and non-manipulability) If the signcryption scheme SignC described in 2.5.3.1. satisfies ESUF-IBSC-CMA property, then EChb scheme has the e-check unforgeability and non-manipulabity property, that is EChb scheme is resistant against e-check forgery and e-check manipulation attacks.

Proof . Let $\mathcal{A}$ *be an adversary that can win the e-check unforgeability game, given in Definition 3.2, with some non-negligible probability* p*. We will show that if such an adversary exists, then it can be used to construct an adversary* $\mathcal{B}$ *that can win the ESUF-IBSC-CMA game (see Definition 2.6) for SignC scheme.*

i) The challenger $\mathcal{C}$ *runs the SignC scheme's Setup algorithm with a security parameter* κ *and gives the system parameters* pp *to the adversary* $\mathcal{B}$*.*

ii) $\mathcal{B}$ *passes* pp *to the adversary* $\mathcal{A}$*.*

iii) $\mathcal{A}$ *performs a polynomially bounded number of queries to* $\mathcal{B}$ *to get:*

- *the public-private key pair associated to an arbitrary identity* U
- *the e-checkbook* Ω_{U} *of* U *with* α, k*.*
- *the e-check* ω_{UV} *from* U *to* V *for an arbitrary identity* V *and* d, a, i*.*

iv) For each query of $\mathcal{A}$*,* $\mathcal{B}$

- *queries Keygen oracle for the public-private key pair associated to* U*,*
- *queries Signcrypt oracle with* $(\mathsf{B}, \mathsf{U}, m_{\mathsf{BU}})$ *to get* $\sigma_{\mathsf{BU}} = (c_{\mathsf{BU}}, S_{\mathsf{BU}}, T_{\mathsf{BU}})$ *on* $m_{\mathsf{BU}} = \mathsf{U}||\mathcal{H}^k(\alpha)$*,*
- *computes* $R_{\mathsf{BU}} = e(T_{\mathsf{BU}}, sk_{\mathsf{U}}^Q)$*.*
- *computes* $\omega_{\mathsf{UV}}^i = \sigma_{\mathsf{UV}}^i = \mathcal{S}(sk_U, m_{\mathsf{UV}}^i, V)$*, where*

$$m_{\mathsf{UV}}^i = \mathsf{d}||\mathsf{a}||i||\mathcal{H}^{k-i}(\alpha)||\mathsf{U}||\mathsf{V}||\sigma_{\mathsf{BU}}||R_{\mathsf{BU}}. \tag{3.38}$$

v) B *returns the following query results to* $\mathcal{A}$*:*

- *the public-private key pair associated to* U*,*
- *the e-checkbook* $\Omega_{\mathsf{U}} = (\alpha, k, \sigma_{\mathsf{BU}}, R_{\mathsf{BU}})$ *of* U*,*
- *the e-check* ω_{UV}^i *from* U *to* V *with* d, a, i*.*

vi) Finally, $\mathcal{A}$ *produces an e-check* $\omega_{\mathsf{U}^*\mathsf{V}^*}^j$ *with* d^*, a^*, j *where*

- *the payer's identity* U^* *was not corrupted and*
- $\omega_{\mathsf{U}^*\mathsf{V}^*}^j$ *is a valid e-check that was not already queried.*

vii) $\mathcal{B}$ *sends* $(\sigma_{\mathsf{U}^*\mathsf{V}^*}^j, \mathsf{U}^*, \mathsf{V}^*)$ *to* $\mathcal{C}$ *and wins the ESUF-IBSC-CMA game with probability* p *since*

- *sender's identity* U^* *was not corrupted,*
- *the result of Unsigncrypt oracle on* $(\sigma_{\mathsf{U}^*\mathsf{V}^*}^j$ *under the private key associated to* V^* *is a valid message-signature pair* $(m^*, (h^*, S^*))$*, where*

$$m^* = \mathsf{d}^*||\mathsf{a}^*||j||\mathcal{H}^{k^*-j}(\alpha^*)||\mathsf{U}^*||\mathsf{V}^*||\sigma_{\mathsf{BU}^*}||R_{\mathsf{BU}^*} \tag{3.39}$$

such that no Signcrypt query

  - *involved* m^**,* U^* *and some receiver* V' *(possibly different from* V^**) and*
  - *resulted in a ciphertext* σ' *whose decryption under the private key* $\mathsf{sk}_{\mathsf{V}}{}'$ *is alleged forgery* $(m^*, (h^*, S^*), \mathsf{V}^*)$*.*

However, as it is already proved, (see [Barreto et al., 2005, Theorem 3]), any adversary can win the ESUF-IBSC-CMA game for SignC with only negligible probability. Thus one can conclude that EChb scheme has the e-check unforgeability property. ■

Theorem 3.3: If the signcryption scheme SignC given in 2.5.3.1. is ciphertext anonymous against adaptive chosen-ciphertext insider attacks (ANON-IBSC-CCA), then EChb scheme has the e-check anonymity property.

Proof . Let A be an adversary that has a non-negligible advantage β in the e-check anonymity game, given in Definition 3.3, and let p be the probability of A correctly guessing the sender and the receiver of the e-check. We are going to show that if such an adversary exists, then it can be used to construct an adversary B who has a non-negligible advantage in the ANON-IBSC-CCA game (see Definition 2.7) for SignC scheme.

i) The challenger $\mathcal{C}$ runs the SignC scheme's Setup algorithm with a security parameter κ and gives the system parameters pp *to the adversary $\mathcal{B}$.*

ii) $\mathcal{B}$ passes pp *to the adversary $\mathcal{A}$.*

iii) $\mathcal{A}$ performs a polynomially bounded number of queries to $\mathcal{B}$ to get:

- *the public-private key pair associated to an arbitrary identity* U
- *the e-checkbook Ω_{U} of* U *with α, k.*
- *the e-check ω_{UV} from* U *to* V *for an arbitrary identity* V *and d, a, i.*

iv) For each query of $\mathcal{A}$, $\mathcal{B}$

- *queries Keygen oracle for the public-private key pair associated to* U*,*
- *queries Signcrypt oracle with $(\mathsf{B}, \mathsf{U}, m_{\mathsf{BU}})$ to get $\sigma_{\mathsf{BU}} = (c_{\mathsf{BU}}, S_{\mathsf{BU}}, T_{\mathsf{BU}})$ on $m_{\mathsf{BU}} = \mathsf{U}||\mathcal{H}^k(\alpha)$,*
- *computes $R_{\mathsf{BU}} = e(T_{\mathsf{BU}}, sk_{\mathsf{U}}^Q)$.*
- *computes $\omega_{\mathsf{UV}}^i = \sigma_{\mathsf{UV}}^i = \mathcal{S}(sk_U, m_{\mathsf{UV}}^i, V)$, where*

$$m_{\mathsf{UV}}^i = \mathsf{d}||\mathsf{a}||i||\mathcal{H}^{k-i}(\alpha)||\mathsf{U}||\mathsf{V}||\sigma_{\mathsf{BU}}||R_{\mathsf{BU}} \tag{3.40}$$

v) B *returns the following query results to $\mathcal{A}$:*

- *the public-private key pair associated to* U*,*
- *the e-checkbook $\Omega_{\mathsf{U}} = (\alpha, k, \sigma_{\mathsf{BU}}, R_{\mathsf{BU}})$ of* U*,*
- *the e-check ω_{UV}^i from* U *to* V *with d, a, i.*

vi) *Eventually, $\mathcal{A}$ submits two sender identities $\mathsf{U}_1, \mathsf{U}_2$ and two recipient identities $\mathsf{V}_1, \mathsf{V}_2$ along with d, a, i to $\mathcal{B}$, where $\mathcal{A}$ has queried $\mathcal{B}$ for Ω_{U_1} and Ω_{U_2}.*

vii) *$\mathcal{B}$ guesses $b'_1, b'_2 \in_R \{0, 1\}$, creates*

$$m_{b'_1 b'_2} = \mathsf{d}||\mathsf{a}||i||\mathcal{H}^{k_{\mathsf{U}_{b'_1}} - i}(\alpha_{\mathsf{U}_{b'_1}})||\mathsf{U}_{b'_1}||\mathsf{V}_{b'_2}||\sigma_{\mathsf{BU}_{b'_1}}||R_{\mathsf{BU}_{b'_1}} \tag{3.41}$$

viii) *$\mathcal{B}$ submits $m_{b'_1 b'_2}$ and $\mathsf{U}_1, \mathsf{U}_2, \mathsf{V}_1, \mathsf{V}_2$ to $\mathcal{C}$.*

ix) *$\mathcal{C}$ selects two bits b_1, b_2 uniformly at random, computes $\sigma_{\mathsf{U}_{b_1}\mathsf{V}_{b_2}}$ on $m_{b'_1 b'_2}$ for identities U_{b_1} and V_{b_2}, and sends $\sigma_{\mathsf{U}_{b_1}\mathsf{V}_{b_2}}$ to $\mathcal{B}$.*

x) *$\mathcal{B}$ sends $\omega_{\mathsf{U}_{b_1}\mathsf{V}_{b_2}} = \sigma_{\mathsf{U}_{b_1}\mathsf{V}_{b_2}}$ to $\mathcal{A}$.*

xi) *$\mathcal{A}$ and $\mathcal{B}$ may adaptively repeat find phases (Steps 3, 4 and 5), without querying the private key and e-checkbook of U_1 and U_2, and e-checks to V_1 and V_2.*

xii) *if B correctly guessed (b_1, b_2), i.e. $(b'_1, b'_2) = (b_1, b_2)$, $\mathcal{A}$ outputs (b_1, b_2) with probability p.*

xiii) *if B correctly guessed (b_1, b_2), then B sends $(b_1^*, b_2^*) = (b_1, b_2)$ to C. Otherwise, B passes random (b_1^*, b_2^*) to C. So, B wins the ANON-IBSC-CCA game with probability $\frac{p}{4} + \frac{3}{16}$ since*

- *neither sender's nor recipient's identity was not corrupted,*
- *probability of $(b_1^*, b_2^*) = (b_1, b_2)$ is $\frac{1}{4}p + \frac{3}{4}\frac{1}{4}$.*

A's advantage is

$$Adv(\mathcal{A}) = |Prob((b_1^*, b_2^*) = (b_1, b_2)) - \frac{1}{4}| \tag{3.42}$$

$$= |\frac{p}{4} + \frac{3}{16} - \frac{1}{4}| \tag{3.43}$$

$$= |\frac{p}{4} - \frac{1}{16}|, \tag{3.44}$$

which is non-negligible given that $p - \frac{1}{4}$ is non-negligible. On the other hand, as it is already stated in Barreto et al. [2005], any adversary can win the ANON-IBSC-CCA game for SignC with only negligible probability. Thus one can conclude that eChb scheme has the e-check anonymity property. ■

Furthermore, naturally eChb scheme is resistant to double spending and replay attacks, since each deposit and transfer order is validated by the bank B and spent / transferred checks are recorded.

Comparison

Current standing of e-checkbook proposals can be summarized as in Table 3.1.

Table 3.1: Feature and Security Comparisons of E-checkbook Schemes

	Correctness	E-check Anonymity	Mutual Authentication	Transferability	Forgery Attack	Manipulation Attack
Pasupathinathan et al. [2005]	✗*	✗*	✓	✗	–	–
Chen et al. [2009]	✓	✗	✗	✗	✓*	✓*
Chang et al. [2009]	✓	✗	✓	✗	–	✓*
Chen et al. [2010]	✓	✗	✓	✗	–	✓*
Chang et al. [2016]	✓	✗	✓	✗	–	✓*
Sertkaya and Kalkar [2019]	✓	✗	✓	✗	–	–
Sertkaya and Kalkar [2021]	✓	✓	✓	✓	–	–

Attacks presented in Sertkaya and Kalkar [2020].

Unfortunately, with the exception of Sertkaya and Kalkar [2019], the previous schemes have some flaws. Additionally, the protocol given in Sertkaya and Kalkar [2019] does not satisfy e-check anonymity and transferability properties. Thus, the eChb protocol proposed by Sertkaya and Kalkar [2021] is the first protocol that enables e-check anonymity and transferability. This protocol utilizes the signcryption scheme SignC given in 2.5.3.1., which requires

- 1 exponentiation in $\mathbb{G}_T$ and 2 scalar multiplications in $\mathbb{G}_1$ for Signcrypt, and
- 1 multiplication and exponentiation in $\mathbb{G}_T$ and 2 pairing computations for Unsigncrypt.

Our e-checkbook protocol requires

- 2 Signcrypt and 2 Unsigncrypt for Issuance phase,
- 1 Signcrypt and 2 Unsigncrypt for Payment phase,
- 3 Signcrypt and 6 Unsigncrypt for Transfer phase,
- 1 Signcrypt and 3 Unsigncrypt for Deposit phase.

4. ELECTRONIC EXAM

Generally, traditional exams involve three entities and four phases. Firstly, during the *registration* phase, the *exam authority* organizes the exams and the *candidates* sign up for it. Then at the *testing* phase, candidates receive the exam and submit their answers. Next, during the *marking* phase, the *examiners* receive and evaluate the candidates' answers. Finally, the exam authority registers the candidates' marks and the candidates learn their marks at the *notification* phase.

Multiple-choice, truth or false, matching, arrangement, fill in the blank, essays, and other types of questions make up an e-exam. There is a distinction to be made between a remote electronic exam (or shortly e-exam) and a computer-based assessment, which relies on the use of specialized software that is not linked to the internet. Network-based assessment, on the other hand, is based on the use of internet techniques such as a remote exam network, Ahmed et al. [2021]. Here we consider network based assessments.

E-exams are organized following the same principles of traditional exams and require the same main security and privacy features. Naturally, in traditional face-to-face exams, candidate authentication and exam rules compliance can be checked more easily. However, e-exam schemes require specific security and privacy concerns on authentication due to the increased attack surface. Besides authentication, candidates that take the exam should be proctored for cheating.

There are various e-exam proposals that aim to assure security and privacy as summarized in the sequel. Here, we revisit secure and privacy preserving e-exam protocol proposals and propose an e-exam protocol Kalkar and Sertkaya [2022] that utilizes decentralized identity-based verifiable credentials for proof of authentication and public-permissioned blockchain for immutably storing records. In regard to the previously proposed e-exam schemes, our scheme offers both privacy enhancement and better efficiency. More concretely, the proposed solution satisfies test answer authentication, examiner authentication, anonymous marking, anonymous examiner, question secrecy, question privacy, mark privacy, test verifiability, and mark verifiability properties.

Since our scheme is based on a permissioned blockchain, here we briefly explain. A blockchain is an immutable transaction ledger that is maintained by a distributed network of nodes, with each node maintaining a copy of the ledger by appending new transactions that have been checked by a consensus protocol. The data on the ledger is organized into blocks, each of which has a hash that links it to the one before it. While public permissionless blockchains allow anyone to write on the ledger, permissioned blockchains follow a governance model that allows only approved nodes to write on it.

Although this system can be realized on existing public blockchains, students and examiners would need to pay fees to the miners to get their transactions written to the ledger. In order to avoid changing costs and being dependent on systems' performance to process transactions, we believe it would be better if this system uses its own blockchain. Permissioned blockchain is chosen over permissionless since it is faster, more scalable, energy-efficient, and participants(schools) are known beforehand.

One example of permissioned blockchain is Hyperledger Fabric that has been developed under the Linux Foundation as an open source project, Androulaki et al. [2018]. Our scheme utilizes a permissioned blockchain as an append-only immutable bulletin board.

Related Work. Online exam proposals dates back to 2006. Castella-Roca et al. [2006] introduced a secure e-exam management system with trusted exam authority who is fully trusted for assuring candidate's privacy. Later, Huszti and Petho [2010] proposed to reduce Castella-Roca et al. [2006]'s trust assumptions by utilizing pseudonyms for the candidates. However, Dreier et al. [2014] showed that Huszti and Petho [2010]'s scheme suffers from several security flaws. Next, Giustolisi et al. [2014] proposed an e-exam protocol called Remark! that utilizes a mix net with at least one honest entity and uses a bulletin board. Bella et al. [2017] proposed a secure exam protocol that does not rely on any trusted party by relying on oblivious transfer and visual cryptography schemes. Traoré et al. [2017] proposed an e-exam system called ExamShield that again fully trusts the exam authorities and focus on biometric authentication that further involves mouse dynamics, keystroke dynamics, and face biometrics.

Islam et al. designed a blockchain based exam system named BSSSQS Islam et al. [2018]. They address the problem of question leaking and their solution depends

on encryption and randomization of the questions. However, they only deal with the delivery of the questions to the candidates.

Mitchell et al. [2019] proposed a decentralized application for an examination review called "dAppER". In fact, dAppER implements existing procedures and documents them on a permissioned blockchain to ensure irreversibility, immutability and auditability. Deborah et al. Deborah L et al. [2019], focuses on a simple e-exam scheme for mutual authentication between the candidate and the server and secure delivery of question paper from the server. However, both of these proposals unfortunately assume the exam authorities are trusted and will not act maliciously. Therefore, they can not determine the misconducts, summarized in Catalano and Gatti [2017], caused by the exam authorities.

For a recent survey on online exams, reader may refer to Muzaffar et al. [2021]. Of all the examined papers, there are three that focus on security. Kausar et al. [2020] focus on authentication and authorization of the candidates and aim to protect security and integrity of the questions. Mathapati et al. [2017] propose to use graphical own image password to eliminate the modifications in the result and generation of fake question papers. Sukadarmika et al. [2018] aims to improve availability of e-exam which is implemented in wireless network. In case of connection problem, this model automatically provides additional time. None of these papers handles marking and notification. In addition to that, they do not fulfill the e-exam security and privacy requirements that are summarized in Section 4.1.1..

Even if there exists many computer-based exam management systems, most of these system designs rely on trusted exam authorities, please refer to Ahmed et al. [2021] for a recent e-exam survey. There are only a few proposals that rely solely on cryptographic primitives to assure security and privacy.

In Kalkar and Sertkaya [2022], we give an online exam proposal based on blockchain which eliminates the need of bulletin boards. As illustrated in Figure 4.1, Our proposal also uses a blind signature scheme which helps to achieve anonymity of the users without using mix nets or trusting a third party. Moreover, due to the nature of the blockchain, the data remains indefinitely and unchanged which also makes audit procedures much easier.

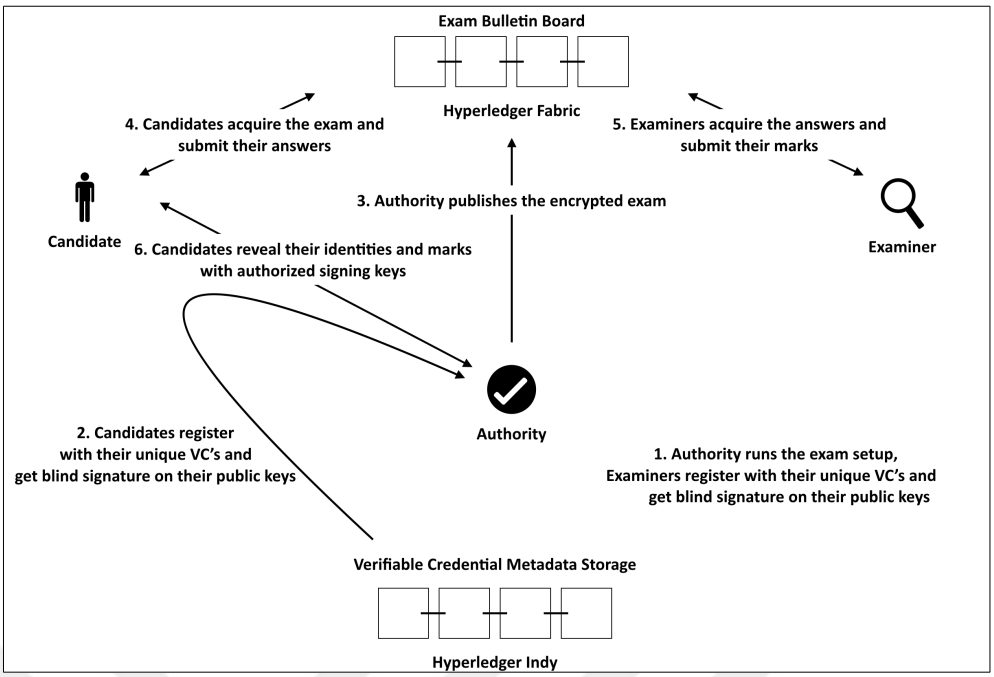

Figure 4.1: Architectural overview of the proposed secure e-exam scheme.

4.1. Electronic Checkbook Scheme

As illustrated in Figure 4.1, typically, there are three type of entities involved in an exam: candidates (c), examiners who grade the exams (e), and an exam authority (EA) that takes part in the organization and execution of the exam. E-exams and traditional exams consist four phases: registration, testing, marking, and notification. The exam authority arranges the exam and candidates enroll for the exam during the registration phase. Testing phase is the phase where candidates receive and take the exam, and submit their answers. Examiners receive and grade the answers during marking phase. Finally, candidates learn their marks in the notification phase.

Exam= {Setup, Registration, Testing, Marking, Notification} is an online exam, where each phase is defined as follows:

- Setup: Secret keys of the EA and public parameters of the scheme is output as pp_{ex}.
- Registration: Candidates and examiners follow the registration phase together with the exam authority in order to register their pseudonyms.
- Testing: Exam authority and the candidates perform this phase. EA publish the test questions for the candidates, candidates solve the questions, append their answers to the questions, and publish.

- Marking: Examiners get the answers and questions of the candidates, mark them and publish the marked answers.
- Notification: During this phase, candidates' marks are registered by EA.

e-Exam Threats

In our scheme, we aim to rule out the following threats same as Remark! scheme proposed in Giustolisi et al. [2014].

i) An intruder impersonating a candidate during the testing.

ii) An intruder tampering with a candidate's test answer or mark.

iii) A candidate seeking to get higher mark than she deserved.

iv) A candidate seeking to coerce the examiner who evaluates her test.

v) The manager tampering with the marks.

vi) An examiner seeking to assign a biased mark to a specific candidate's test.

4.1.1. Requirements

In this subsection, we give the security requirements that a secure e-exam should satisfy. These requirements follow from the previous studies Giustolisi et al. [2013] and Giustolisi et al. [2014].

- Test Answer Authentication. The exam authority only accepts test answers submitted by registered candidates.
- Examiner Authentication. The exam authority only accepts evaluations provided by a registered examiner.
- Anonymous Marking. No one learns the author of a test answer before the test is marked.
- Anonymous Examiner. No candidate learns the identity of the examiner who evaluates their test answers.
- Question Secrecy. No candidate learns the test question before the testing phase begins.
- Question Privacy. The exam authority does not learn which test question is assigned to a specific candidate.

- Mark Privacy. The candidate learns only her mark and not those of other candidates.
- Test Verifiability. The candidate can verify that her test is considered for evaluation.
- Mark Verifiability. The candidate can verify that the exam authority registers the mark she was assigned to by the examiner and the exam authority can verify that the candidate gets the mark she was given by the examiner.

e-Exam Assumptions

Design and analysis of the proposed online exam scheme rely on the following assumptions.

i) We assume that there exists an authentication mechanism for the exam authority to authenticate the candidates and the examiners at the beginning of the registration phase. This can be realized for example anonymous credential Camenisch et al. [2016] based self-sovereign identity solutions Khovratovich and Law [2016].

ii) To mitigate cheating, candidates are invigilated during the testing phase.

iii) A private, permissioned blockchain is already established and ready to use.

4.2. Proposed Scheme

Our scheme assumes existence of an SSI mechanism and each entity holds a set of verifiable credentials to prove her identity and use for authentication. For concrete SSI primitives, please refer to Camenisch et al. [2016] and Sovrin's self-sovereign identity Khovratovich and Law [2016]. Proposed protocol is illustrated in Figure 4.1. In a nutshell, the protocol is pursued as follows.

- The exam authority EA determines Pec and Sig primitives, runs Setup phase for generating system-wide public parameters and her key pairs for blind Schnorr signature scheme,
- At the Registration phase, the candidate C proves her identity using her verifiable credentials to EA and requests EA to blindly sign her public keys,
- Following EA's randomly pairing candidates and examiners and generating test questions, C takes the test during the Testing phase. After answering the questions, C signs her answers and encrypts for the examiner.

- During the Marking phase, the examiner decrypts the answers, marks the test, appends her mark, signs and encrypts for the candidate.
- The candidate decrypts the examiner's mark, learns her mark, adds her identifying information to bind the public keys to her real identity, and encrypts for the exam authority. The exam authority decrypts that information, matches the public keys of the candidate to the candidate's real identity, and saves the candidate's mark at the end of Notification.

For concrete instantiation of the proposed protocol,

- verifiable credentials and corresponding metadata can be stored on Hyperledger Indy Foundation [2018].
- ElGamal encryption schemeElGamal [1985] can be utilized for Pec scheme.
- Schnorr signaturesSchnorr [1989] can be utilized for Sig scheme.
- Blind Schnorr signature scheme will be used for blindly signing the public keys of the examiners and candidates.
- the permissioned blockchain that will be utilized as bulletin board can be Hyperledger Fabric Androulaki et al. [2018].

There is a famous chart in NIST's technical report (Page 42, Figure 6 Yaga et al. [2018]) that guides someone to decide if they really need a blockchain for their problem. Referring to the mentioned figure; online-exam system needs a shared, consistent data store in order to archive exams, there are more than one entity(schools, students, teachers) that contributes to the data, data records are never updated or deleted, sensitive identifiers are not stored on the blockchain(exams, answers, and marks are encrypted; identities of students and examiners are anonymous), since the exam system is nation-wide or even global, entities can not decide who should be in control of the data store, and finally a tamper-proof log of all writes to the data store are required for any dispute resolution.

In this thesis, we treat Pec and Sig algorithms as black-box since any CCA-secure encryption algorithm and existentially-unforgeable signature algorithm can be utilized as Pec and Sig, respectively. However, since the blind signature algorithm is crucial for the protocol, we give it in detail in the Setup and Registration phases even though we just follow blind Schnorr signature scheme. More concretely, steps 6-9 and steps

2-5 of Setup and Registration phases correspond to the blind Schnorr signature scheme.

Setup

During the Setup phase, the exam authority EA publishes the exam public parameters $\mathsf{pp}_{\mathsf{ex}}$ and examiners register themselves for the exam.

i) EA determines a public key encryption scheme Pec that is going to be used for encrypting and decrypting questions, answers, and marks and generates its public parameters $\mathsf{pp}_{\mathsf{pec}}$.

ii) EA determines a digital signature scheme Sig that is going to be used to sign questions, answers, and marks and generates its public parameters $\mathsf{pp}_{\mathsf{sig}}$ along with its signing key pair $(\mathsf{sk}^{\mathsf{EA}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{EA}}_{\mathsf{sig}})$.

iii) EA generates the public parameters for blind Schnorr signature scheme $\mathsf{pp}_{\mathsf{Sch}} = (p, \mathbb{G}, P, \mathcal{H})$ along with its signing key pair $(\mathsf{sk}^{\mathsf{EA}}_{\mathsf{Sch}}, \mathsf{pk}^{\mathsf{EA}}_{\mathsf{Sch}})$, where $\mathsf{sk}^{\mathsf{EA}}_{\mathsf{Sch}} \leftarrow_{\mathsf{r}} \mathbb{Z}_{\mathsf{p}}, \mathsf{pk}^{\mathsf{EA}}_{\mathsf{Sch}} = \mathsf{xP}$.

iv) EA publishes the exam public parameters $\mathsf{pp}_{\mathsf{ex}}$ as

$$\mathsf{pp}_{\mathsf{ex}} \leftarrow (\mathsf{pp}_{\mathsf{pec}}, \mathsf{pp}_{\mathsf{sig}}, \mathsf{pp}_{\mathsf{Sch}}, \mathsf{pk}^{\mathsf{EA}}_{\mathsf{sig}}). \tag{4.1}$$

v) e proves herself to EA using verifiable credentials .

vi) EA chooses $r \leftarrow_r \mathbb{Z}_p$, computes $R' \leftarrow [r]P$ and sends R' to e.

vii) e creates key pairs for encryption and signature algorithms,

$$(\mathsf{sk}^{\mathsf{e}}_{\mathsf{pec}}, \mathsf{pk}^{\mathsf{e}}_{\mathsf{pec}}) \leftarrow \mathsf{K}_{\mathsf{pec}}(\mathsf{pp}_{\mathsf{pec}}), \tag{4.2}$$

$$(\mathsf{sk}^{\mathsf{e}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{e}}_{\mathsf{sig}}) \leftarrow \mathsf{K}_{\mathsf{sig}}(\mathsf{pp}_{\mathsf{sig}}), \tag{4.3}$$

chooses $\alpha, \beta \leftarrow_r \mathbb{Z}_p$, computes

$$R \leftarrow R' + [\alpha]P + [\beta]\mathsf{pk}^{\mathsf{EA}}_{\mathsf{Sch}} \tag{4.4}$$

$$c \leftarrow \mathcal{H}(R, \mathsf{pk}^{\mathsf{e}}_{\mathsf{pec}} || \mathsf{pk}^{\mathsf{e}}_{\mathsf{sig}}) \tag{4.5}$$

$$c' \leftarrow c + \beta \mod p, \tag{4.6}$$

and sends c' to EA.

viii) EA computes $s' \leftarrow r + c'x \mod p$ and sends s' to e.

ix) e checks if

$$[s']P = R' + [c']\mathsf{pk}^{\mathsf{EA}}_{\mathsf{Sch}}, \tag{4.7}$$

computes $s = s' + \alpha \mod p$ and EA's signature on $\mathsf{pk}^{\mathsf{e}}_{\mathsf{pec}}||\mathsf{pk}^{\mathsf{e}}_{\mathsf{sig}}$ is

$$\sigma_{\mathsf{e}} \leftarrow (R, s). \tag{4.8}$$

x) e sends $(\mathsf{pk}^{\mathsf{e}}_{\mathsf{pec}}||\mathsf{pk}^{\mathsf{e}}_{\mathsf{sig}}, \sigma_{\mathsf{e}})$ to the blockchain along with a signature on it signed using $\mathsf{sk}^{\mathsf{e}}_{\mathsf{sig}}$.

xi) Nodes first verify the outer signature by $\mathsf{pk}^{\mathsf{e}}_{\mathsf{sig}}$, then checks if

$$[s]P = R + [c]\mathsf{pk}^{\mathsf{EA}}_{\mathsf{Sch}}, \tag{4.9}$$

where

$$c = \mathcal{H}(R, \mathsf{pk}^{\mathsf{e}}_{\mathsf{pec}}||\mathsf{pk}^{\mathsf{e}}_{\mathsf{sig}}). \tag{4.10}$$

If only all the checks holds, the transaction is added to the ledger.

Registration

After the examiner registration is completed, an eligible candidate c and the exam authority EA performs the registration phase together.

i) c proves herself to EA using verifiable credentials .

ii) EA chooses $r \leftarrow_r \mathbb{Z}_p$, computes $R' \leftarrow [r]P$ and sends R' to c.

iii) c creates key pairs for encryption and signature algorithms,

$$(\mathsf{sk}^{\mathsf{c}}_{\mathsf{pec}}, \mathsf{pk}^{\mathsf{c}}_{\mathsf{pec}}) \leftarrow \mathsf{K}_{\mathsf{pec}}(\mathsf{pp}_{\mathsf{pec}}), \tag{4.11}$$

$$(\mathsf{sk}^{\mathsf{c}}_{\mathsf{sig}}, \mathsf{pk}^{\mathsf{c}}_{\mathsf{sig}}) \leftarrow \mathsf{K}_{\mathsf{sig}}(\mathsf{pp}_{\mathsf{sig}}), \tag{4.12}$$

$$\tag{4.13}$$

chooses $\alpha, \beta \leftarrow_r \mathbb{Z}_p$, records β, computes

$$R \leftarrow R' + [\alpha]P + [\beta]\mathsf{pk}^{\mathsf{EA}}_{\mathsf{Sch}} \tag{4.14}$$

$$c \leftarrow \mathcal{H}(R, \mathsf{pk}^{\mathsf{c}}_{\mathsf{pec}} || \mathsf{pk}^{\mathsf{c}}_{\mathsf{sig}}) \tag{4.15}$$

$$c' \leftarrow c + \beta \mod p, \tag{4.16}$$

and sends c' to EA.

iv) EA records c' for c, computes $s' \leftarrow r + c'x \mod p$ and sends s' to c.

v) c checks if

$$[s']P = R' + [c']\mathsf{pk}^{\mathsf{EA}}_{\mathsf{Sch}}, \tag{4.17}$$

computes $s = s' + \alpha \mod p$ and EA's signature on $\mathsf{pk}^{\mathsf{c}}_{\mathsf{pec}} || \mathsf{pk}^{\mathsf{c}}_{\mathsf{sig}}$ is $\sigma_{\mathsf{c}} \leftarrow (R, s)$.

vi) c sends $(\mathsf{pk}^{\mathsf{c}}_{\mathsf{pec}} || \mathsf{pk}^{\mathsf{c}}_{\mathsf{sig}}, \sigma_{\mathsf{c}})$ to the blockchain along with a signature on it signed using $\mathsf{sk}^{\mathsf{c}}_{\mathsf{sig}}$.

vii) Nodes first verify the outer signature by $\mathsf{pk}^{\mathsf{c}}_{\mathsf{sig}}$, then checks if

$$[s]P = R + [c]\mathsf{pk}^{\mathsf{EA}}_{\mathsf{Sch}}, \tag{4.18}$$

where

$$c = \mathcal{H}(R, \mathsf{pk}^{\mathsf{c}}_{\mathsf{pec}} || \mathsf{pk}^{\mathsf{c}}_{\mathsf{sig}}). \tag{4.19}$$

If only all of the checks are successful, nodes append the transaction to their ledger.

Testing

Before the testing stars, the exam authority randomly selects an examiner e_c for each candidate c and generates the test questions q_c.

i) EA signs (q_c, pk_{e_c}), $S_{q_c} \leftarrow S_{sig}(sk_{ex}, (q_c, pk_c, pk_{e_c}))$.

ii) EA encrypts $(pk_{e_c}, S_{q_c}, q_c, pk_{e_c})$ under pk_c, i.e.

$$E_{q_c} \leftarrow E_{pec}(pk_c, (pk_{e_c}, S_{q_c}, q_c, pk_{e_c}))). \tag{4.20}$$

iii) EA sends E_{q_c} to the blockchain along with a signature on it signed using sk_{sig}^{EA}.

iv) Nodes verify the outer signature by pk_{sig}^{EA}. If successful, writes the transaction to the ledger.

v) c decrypts E_{q_c}, i.e $(pk_{e_c}, S_{q_c}, pk_c, pk_{e_c}) \leftarrow D_{pec}(sk_c, E_{q_c})$.

vi) c verifies S_{q_c} by checking that

$$1 = V_{sig}(pk_{sig}^{EA}, (q_c, pk_{e_c}), S_{q_c}). \tag{4.21}$$

vii) When c finishes her test, she appends her answers and her public key to the questions,

$$T_c = (q_c, pk_{e_c}, a_c, pk_c). \tag{4.22}$$

viii) c signs the filled test T_c,

$$\sigma_c \leftarrow S_{sig}(sk_{sig}^c, T_c). \tag{4.23}$$

ix) c encrypts (σ_c, T_c) under the public key of the stated examiner e_c and sends E_{T_c} to the blockchain along with a signature on it signed using sk_{sig}^c.

$$E_{T_c} \leftarrow E_{pec}(pk_{pec}^{e_c}, (\sigma_c, T_c)) \tag{4.24}$$

x) Nodes verify the outer signature by pk_{sig}^c. Then, they also verify that c completed the registration step. (The candidate can send Transaction ID belonging to the

registration transaction). If both checks are successful, writes the transaction to the ledger.

Marking

i) $\mathsf{e_c}$ decrypts $\mathsf{E_{T_c}}$ and gets

$$(\sigma_\mathsf{c}, \mathsf{T_c}) \leftarrow \mathsf{D_{pec}}(\mathsf{sk}^{\mathsf{e_c}}_{\mathsf{pec}}, \mathsf{E_{T_c}}). \tag{4.25}$$

ii) $\mathsf{e_c}$ verifies the signature σ_c.

iii) After marking the exam, the examiner $\mathsf{e_c}$ appends the mark $\mathsf{m_c}$ and generates

$$\mathsf{M_c} \leftarrow (\mathsf{T_c}, \mathsf{m_c}). \tag{4.26}$$

iv) $\mathsf{e_c}$ signs $\mathsf{M_c}$ with his private key $\mathsf{sk}^{\mathsf{e_c}}_{\mathsf{sig}}$,

$$\sigma_{\mathsf{e,c}} \leftarrow \mathsf{S_{sig}}(\mathsf{sk}^{\mathsf{e_c}}_{\mathsf{sig}}, \mathsf{M_c}). \tag{4.27}$$

v) $\mathsf{e_c}$ encrypts $(\sigma_{\mathsf{e,c}}, \mathsf{M_c})$ for c and sends $\mathsf{E_{M_c}}$ to the blockchain along with a signature on it signed using $\mathsf{sk}^{\mathsf{e_c}}_{\mathsf{sig}}$.

$$\mathsf{E_{M_c}} \leftarrow \mathsf{E_{pec}}(\mathsf{pk}^{\mathsf{c}}_{\mathsf{pec}}, (\sigma_{\mathsf{e,c}}, \mathsf{M_c})). \tag{4.28}$$

vi) Nodes verify the outer signature by $\mathsf{pk}^{\mathsf{e_c}}_{\mathsf{sig}}$. Then, they also verify that c completed the registration step (The examiner can send Transaction ID belonging to the setup transaction.). If both checks are successful, writes the transaction to the ledger.

Notification

An eligible candidate c and the exam authority performs the notification phase together.

i) c decrypts $\mathsf{E_{M_c}}$ and gets $(\sigma_{\mathsf{e,c}}, \mathsf{M_c}) \leftarrow \mathsf{D_{pec}}(\mathsf{sk}^{\mathsf{c}}_{\mathsf{pec}}, \mathsf{E_{M_c}})$.

ii) c verifies the signature $\sigma_{\mathsf{e,c}}$.

iii) c creates a message $\mathsf{M} \leftarrow (\mathsf{M_c}, \sigma_{\mathsf{e,c}}, \mathsf{pk^c_{pec}}||\mathsf{pk^c_{sig}}, (\mathsf{R}, \mathsf{s}), \beta)$, encrypts M for EA, and sends $\mathsf{E_M}$ to the blockchain along with a signature on it signed using $\mathsf{sk^c_{sig}}$,

$$\mathsf{E_M} \leftarrow \mathsf{E_{pec}}(\mathsf{pk^{EA}_{pec}}, M). \tag{4.29}$$

iv) Nodes verify the outer signature by $\mathsf{pk^{e_c}_{sig}}$. Then, they also verify that c completed the registration step(The candidate can send Transaction ID belonging to the registration transaction.). If both checks are successful, writes the transaction to the ledger. If successful, write the transaction to the ledger.

v) EA decrypts $\mathsf{E_M}$ and gets

$$(\mathsf{M_c}, \sigma_{\mathsf{e,c}}, \mathsf{pk^c_{pec}}||\mathsf{pk^c_{sig}}, (\mathsf{R}, \mathsf{s}), \beta) \leftarrow \mathsf{D_{pec}}(\mathsf{sk^{EA}_{pec}}, \mathsf{E_M}). \tag{4.30}$$

vi) EA verifies the signature $\sigma_{\mathsf{e,c}}$ on $\mathsf{M_c}$ and also verifies (R, s) on $\mathsf{pk^c_{pec}}||\mathsf{pk^c_{sig}}$.

vii) EA finds the candidate's identity using β and assigns the candidate's mark.

4.3. Security & Performance Analysis

Below, we give reasons why the properties of an online exam are satisfied assuming invigilation is in place.

- Test Answer Authentication. The exam authority encrypts the questions for the candidates who have pseudonyms signed by EA and only accepts marks whose signature can be verified by pseudonyms that have exam authority's signature on it. Since this signature can not be forged, test answer authentication property is satisfied.

- Examiner Authentication. Candidates encrypt their answers using the public key of the assigned examiner and this public key is signed by EA and EA only accepts marks whose signature can be verified by pseudonyms that have exam authority's signature on it. Since this signature can not be forged, examiner authentication property is also satisfied.

- Anonymous Marking. Since the examiner and the exam authority only knows the pseudonym of the candidate, anonymous marking property is satisfied. The exam

authority can not associate the pseudonym of a candidate with the candidate's real identity since blind signature is used.

- Anonymous Examiner. Since the candidate only knows the pseudonym of the examiner that is going to mark that candidate's answers, anonymous examiner property is satisfied.

- Question Secrecy. The exam authority publishes the test questions after the candidates are under invigilation, so no candidate learns the test questions before the testing phase starts, hence question secrecy holds.

- Question Privacy. Since test questions are encrypted with the candidate's pseudonym which does not link to the candidate's real identity, EA can not learn which test questions are assigned to a particular candidate. This guarantees question privacy.

- Mark Privacy. Mark privacy property requires that a candidate only learns her mark not the other candidates'. Since each mark is only carried encrypted under either the candidate's or the exam authority's public key, no one other than the candidate, the examiner that marked the exam, and the exam authority learns that candidate's mark.

- Test Verifiability. This property requires that both the exam authority and the candidate can verify that the mark given by the examiner is not changed. Since the mark is sent to the candidate by the examiner, candidate part is satisfied. Then, candidate sends the mark to the exam authority. Since this mark has signature of the examiner which is assigned to mark the test, the exam authority also is able to verify.

Please note that this scheme is not resistant to the situation where the exam authority and the candidate collude in any way. The exam authority may behave maliciously and give the exam questions to a specific candidate before the testing phase starts using other channels. Similar to the previously proposed schemes Castella-Roca et al. [2006]; Huszti and Petho [2010]; Giustolisi et al. [2014, 2017], our scheme would suffer from this.

Castella-Roca et al. [2006]; Deborah L et al. [2019]; Traoré et al. [2017]; Mitchell et al. [2019] require full trust on exam authority. BSSSQS Islam et al. [2018] only focuses on delivery of the test questions and Kausar et al. [2020]; Mathapati et al. [2017]; Sukadarmika et al. [2018] do not have marking and notification phases. There are four full e-exam proposals that do not rely on a trusted exam authority Bella

et al. [2017]; Giustolisi et al. [2014, 2017]; Huszti and Petho [2010], aside from this thesis. Bella et al. [2017] requires an administrator and candidates to physically meet since they use visual cryptography in order to jointly generate candidates' pseudonyms. Giustolisi et al. [2017] is slightly modified version of Giustolisi et al. [2014] and both of them relies on exponential mix-nets for student and examiner anonymization and assume that "an authenticated, append-only, bulletin board is available. On it, everyone is guaranteed to see the same data. Write access will be restricted however to the appropriate entities ." Using blind signatures eliminates costly mix-net computations and the described bulletin is exactly a permissioned blockchain. Huszti and Petho [2010] does not satisfy test answer authentication, anonymous marking, anonymous examiner, and mark privacy properties as shown by Dreier et al. [2014].

Please note that from the five proposals that do not require a trusted exam authority; Bella et al. [2017]; Giustolisi et al. [2014], and ours do not provide full security proofs while Giustolisi et al. [2017]; Huszti and Petho [2010] verify their proposals using ProVerif Blanchet [2014].

Although we do not provide any implementation, Hyperledger Indy already supports verifiable credentials and can be used in our system for authentication purposes. Other than that, our system can work on Hyperledger Fabric.

5. OUTSOURCING PAIRING COMPUTATIONS

Pairing computation is one of the most expensive tasks among cryptographic primitives. Reducing the computational cost of pairings is a widely explored area Barreto et al. [2007]; Beuchat et al. [2010]; Hess et al. [2006]; Koblitz and Menezes [2005]; Scott et al. [2006]. However, results are not practical enough to be deployed within resource constrained and energy limited devices. Furthermore, there has been advances Barbulescu and Duquesne [2019]; Kiyomura et al. [2017] on the complexity of solving the discrete logarithm problem on the image group of pairing function, and these advances require to increase key sizes which results in increased computations for calculating a pairing. Thus, delegation of pairing computation to more powerful and possibly untrusted servers is a cost-effective way of realizing pairing-based cryptography for resource constrained devices.

Outsourcing a pairing computation to a more resourcefull but potentially malicious server is first mentioned by Girault and Lefranc [2005] as a part of speeding up the verification step of an authentication/signature scheme. The first verifiable pairing outsource scheme was proposed by Chevallier-Mames et al. [2005] and later improved by Kang et al. [2005] and Canard et al. [2014]. However their proposal were not applicable since their outsourcing computations were more costly than calculating a pairing. Chen et al. [2015] proposed the first efficient pairing outsourcing method which is secure under the One-Malicious Version of a Two-Untrusted Program (OMTUP) model. This model assumes that there exists two untrusted programs performing the delegated computation but exactly one of them may behave maliciously.

After Chen et al. [2015]'s efficient proposition, Tian et al. [2015] and Arabacı et al. [2015] further improved the efficiency of delegating pairing computation. However, schemes were only 1/2-checkable, i.e. servers can cheat the outsourcer with 1/2 probability. Tian et al. [2015] also proposed another solution with adjustable checkability, i.e. one can achieve higher checkability with the cost of more computations. Then many people including Kalkar *et al.* claimed that they proposed fully checkable outsourcing schemes Ren et al. [2016]; Luo et al. [2016]; Ren et al. [2017]; Kalkar et al. [2018]; Luo et al. [2018].

Following the lines of Chevallier-Mames et al. [2005]; Canard et al. [2014], a secure fully verifiable delegation protocol for pairing computation is expected to satisfy informally the following main properties:

- Completeness: After completion of the protocol with an honest program U, the delegator T obtains $e(A, B)$ on the inputs $A \in \mathbb{G}_1$ and $B \in \mathbb{G}_2$, except with negligible probability.
- Secrecy & Privacy: An untrusted program should not learn any information about the input points $A \in \mathbb{G}_1$ and $B \in \mathbb{G}_2$. More formally, for any malicious program U, there exists a simulator $\mathcal{S}$ such that for any $A \in \mathbb{G}_1$ and $B \in \mathbb{G}_2$, the output of $\mathcal{S}$, to which the points A are B are not given, is computationally indistinguishable from the program's view:

$$\mathcal{S} \stackrel{c}{\equiv} \mathrm{View}_U(A, B). \tag{5.1}$$

- Verifiability: The delegator should be able to detect a cheating program, except with negligible probability. More formally, for any cheating program U and for any input values $A \in \mathbb{G}_1$ and $B \in \mathbb{G}_2$, the delegator outputs either $\perp$ or $e(A, B) \in \mathbb{G}_3$, except with negligible probability.

5.1. Security Model

For secure and verifiable outsourcing of cryptographic computations in the presence of potentially malicious servers, the first simulation-based security notions were provided by Hohenberger and Lysyanskaya [2005]. Their model first defines three models:

- One-Untrusted Program (OUP): The delegated computation is performed by a single program which is malicious.
- One-Malicious version of a Two-Untrusted Program (OMTUP): The delegated computation is performed by two programs. Both of them are curious and untrusted however, only one of them is malicious.
- Two-Untrusted Program (TUP): The delegated computations are performed by two programs. Both of the programs are untrusted and malicious however they are not allowed to maliciously collude.

Different protocols chose to follow different security models. However, most of the protocols follow Hohenberger and Lysyanskaya [2005]'s security model and we give its definition here.

Informally, a *trusted* honest but resource-component part T securely delegates some work to a potentially untrusted component U, and (T, U) forms a delegated-secure implementation of a cryptographic scheme Alg if

- T and U jointly implements $Alg = T^U$,
- if T is given oracle access to a malicious U', $U \neq U'$, then despite the assumption that U' acts maliciously every time it is invoked by recording its own computation over time, it cannot obtain any information about both the input and the output of $T^{U'}$.

Since U' is not the single entity acting maliciously and interacting with Alg, the adversary $\mathcal{A}$ consists of two parts:

- the adversarial component U' operating in place of U,
- an adversarial component *environment* E submitting adversarially chosen inputs to Alg.

The fundamental assumption in Hohenberger and Lysyanskaya [2005] is that E and U' are allowed to collude to create a strategy but only until interacting with T. Then, first logical divisions of inputs to Alg are

- *secret* information solely available to T,
- environmentally *protected* information available to both T and E but nor available to U',
- *protected* information available to both T and U' but nor available to E,
- *unprotected* information available to T, E, and U'.

This division includes in particular the cases where E may have access something about the protected inputs to Alg which is either not available to U' or not available to E. More concretely, T might hide some of these from U' whereas E can clearly see all of its own adversarial inputs to Alg. Likewise, T might hide some information from E whereas U' can see some of protected random inputs generated solely by T which E cannot see as E and U' can only communicate through T. Throughout the thesis

both environmentally protected and protected information are called protected if there is no need to distinguish the cases from which adversarial component the information is protected.

Moreover, the above divisions have additional subdivisions depending on whether the inputs were generated *honestly* or *adversarially*. Note that there cannot however exist a adversarial secret input.
Similarly, the outputs of Alg are logically divided into *secret, protected,* and *unprotected* outputs. The simplified formal definition is given as follows (i.e. by neglecting possible relations of the inputs and outputs to each other):

Definition 5.1: Hohenberger and Lysyanskaya [2005] (Algorithm with delegated-IO) An algorithm Alg is said to obey the delegation input/output specification if it takes five inputs, and produces three outputs. The first three inputs are generated by an honest party T, and are classified by how much information about them is available to the adversary $\mathcal{A} = (E, U')$, where E is the adversarial environment submitting adversarially chosen inputs to Alg, and U' is the adversarial component operating in place of oracle U. The first input is called the honest, secret input, which is unknown to both E and U; the second is called the honest, protected input, which may either be known to E, but is protected from U, or known to U, but is protected from E; and the third is called the honest, unprotected input, which may be known by both E and U. In addition, there are two adversarially-chosen inputs generated by the environment E; the adversarial, protected input, which is known to E, but protected from U; and the adversarial, which may be known by both E and U. Similarly, the first output called secret is unknown to both E and U; the second is protected, which may be known to E, but it is protected from U; and the third is unprotected, which may be known by both E and U.

Definition 5.2: Hohenberger and Lysyanskaya [2005] (Delegated Security)
Let Alg$(\cdot, \cdot, \cdot, \cdot, \cdot)$ be an algorithm with delegated-IO. A pair of algorithms (T, U) is said to be a delegated-secure implementation of an algorithm Alg if:
Completeness. T^U is a correct implementation of Alg.
Security. For all probabilistic polynomial-time adversaries $A = (E, U')$, there exist probabilistic expected polynomial-time simulators (S_1, S_2) such that the following pairs of random variables are computationally indistinguishable. We assume that the honestly-generated inputs are chosen by a process I.

- *Pair One:* $EVIEW_{real}$~$EVIEW_{ideal}$*:*
  - *The view that the adversarial environment* E *obtains by participating in the following REAL process:*

$$EVIEW^i_{real} = \{(istate^i, x^i_{hs}, x^i_{hp}, x^i_{hu}) \leftarrow I(1^k, istate^{i-1});$$
$$(estate^i, j^i, x^i_{ap}, x^i_{au}, stop^i) \leftarrow E(1^k, EVIEW^{i-1}_{real}, x^i,_{hp}, x^i_{hu});$$
$$(tstate^i, ustate^i, y^i_s, y^i_p, y^i_u) \leftarrow$$
$$T^{U'(ustate^{i-1})}(tstate^{i-1}, x^{j^i}_{hs}, x^{j^i}_{hp}, x^{j^i}_{hu}, x^i_{ap}, x^i_{au}) :$$
$$(estate^i, y^i_p, y^i_u)\}$$

    $EVIEW_{real} = EVIEW^i_{real}$ *if* $stop^i = TRUE$.
    The real process proceeds in rounds. In round i*, the honest (secret, protected, and unprotected) inputs* $(x^i_{hs}, x^i_{hp}, x^i_{hu})$ *are picked using an honest, stateful process* I *to which the environment does not have access. Then the environment, based on its view from the last round, chooses* (0) *the value of its* $estate_i$ *variable as a way of remembering what it did next time it is invoked;* (1) *which previously generated honest inputs*

$$(x^{j^i}_{hs}, x^{j^i}_{hp}, x^{j^i}_{hu})$$

    to give to $T^{U'}$ *(note that the environment can specify the index* j^i *of these inputs, but not their values);* (2) *the adversarial, protected input* x^i_{ap}*;* (3) *the adversarial, unprotected input* x^i_{au}*;* (4) *the Boolean variable* $stop^i$ *that determines whether* $round\ i$ *is the last round in this process. Next, the algorithm* $T^{U'}$ *is run on the inputs*

$$(tstate^{i-1}, x^{j^i}_{hs}, x^{j^i}_{hp}, x^{j^i}_{hu}, x^i_{ap}, x^i_{au}),$$

    where $tstate^{i-1}$ *is* T*'s previously saved state, and produces a new state* $tstate^i$ *for* T*, as well as the secret* y^i_s*, protected* y^i_p*, and unprotected* y^i_u *outputs. The oracle* U' *is given its previously saved state,* $ustate^{i-1}$*, as input, and the current state of* U' *is saved in the variable* $ustate^i$*. The view of the real process in* $roundi$ *consists of* $estate^i$*, and the values* y^i_p *and* y^i_u*. The overall view of the environment in the real process is just its view in the last round, i.e. for* i *with* $stop^i = TRUE$.

  - *The IDEAL process:*

$$EVIEW^i_{ideal} = \{(istate^i, x^i_{hs}, x^i_{hp}, x^i_{hu}) \leftarrow I(1^k, istate^{i-1});$$
$$(estate^i, j^i, x^i_{ap}, x^i_{au}, stop^i) \leftarrow E(1^k, EVIEW^{i-1}_{ideal}, x^i,_{hp}, x^i_{hu});$$
$$(astate^i, y^i_s, y^i_p, y^i_u) \leftarrow Alg(astate^{i-1}, x^{j^i}_{hs}, x^{j^i}_{hp}, x^{j^i}_{hu}, x^i_{ap}, x^i_{au});$$

$$(sstate^i, ustate^i, Y_p^i, Y_u^i, replace^i) \leftarrow S_1^{U'(ustate^{i-1})}(sstate^{i-1}, \cdots, x_{hp}^{j^i}, x_{hu}^{j^i}, x_{ap}^i, x_{au}^i, y_p^i, y_u^i);$$
$$(z_p^i, z_u^i) = replace^i(Y_p^i, Y_u^i) + (1 - replace^i)(y_p^i, y_u^i) : (estate^i, z_p^i, z_u^i)\}$$

$EVIEW_{ideal} = EVIEW_{ideal}^i$ *if* $stop^i = TRUE$.
The ideal process also proceeds in rounds. In the ideal process, we have a stateful simulator S_1 *who, shielded from the secret input* x_{hs}^I*, but given the non-secret outputs that Alg produces when run all the inputs for round* i*, decides to either output the values* (y_p^i, y_u^i) *generated by Alg, or replace them with some other values* (Y_p^i, Y_u^i) *Note that this process is captured by having the indicator variable* $replace^i$ *be a bit determining whether* y_p^i *will be replaced with* Y_p^i*. In doing so, it is allowed to query the oracle* U'*; moreover,* U' *saves its state as in the real experiment.*

- *Pair Two:* $UVIEW_{real}$~$UVIEW_{ideal}$*:*
  - *The view that the untrusted software* U' *obtains by participating in the REAL process described in Pair One.* $UVIEW_{real} = ustate^i$ *if* $stop^i = TRUE$.
  - *The* $IDEAL$ *process:*

$$UVIEW_{ideal}^i = \{(istate^i, x_{hs}^i, x_{hp}^i, x_{hu}^i) \leftarrow I(1^k, istate^{i-1});$$
$$(estate^i, j^i, x_{ap}^i, x_{au}^i, stop^i) \leftarrow E(1^k, estate^{i-1}, x_{hp}^i, x_{hu}^i, y_p^{i-1}, y_u^{i-1});$$
$$(astate^i, y_s^i, y_p^i, y_u^i) \leftarrow Alg(astate^{i-1}, x_{hs}^{j^i}, x_{hp}^{j^i}, x_{hu}^{j^i}, x_{ap}^i, x_{au}^i);$$
$$(sstate^i, ustate^i) \leftarrow S_2^{U'(ustate^{i-1})}(sstate^{i-1}, x_{hu}^{j^i}, x_{au}^i) : (ustate^i)\}$$

$UVIEW_{ideal} = UVIEW_{ideal}^i$ *if* $stop^i = TRUE$.
In the ideal process, we have a stateful simulator S_2 *who, equipped with only the unprotected inputs* (x_{hu}^i, x_{au}^i)*, queries* U'*. As before,* U' *may maintain state.*

Definition 5.3: (α*-Efficiency)*

A pair of algorithms (T, U_1, U_2) *are an* α*-efficient delegated-implementation of an algorithm Alg if* (1) T^{U_1,U_2} *is a complete implementation of Alg, and* (2) $\forall$ *inputs* x*, the running time of* T *is smaller than an* α*-multiplicative factor of the running time of* $Alg(x)$.

Definition 5.4: (β-Verifiability)
A pair of algorithms (T, U_1, U_2) are a β-verifiable delegated implementation of an algorithm Alg if (1) T^{U_1,U_2} is a complete implementation of Alg, and (2) $\forall$ inputs x, if $U_i', i = 1, 2$ deviates from its advertised functionality during the execution of $T^{(U_1',U_2')}(x), T$ will detect the error with probability larger than β. In particular, if T will always detect the error, except with negligible probability, i.e. $1 - \beta$ is negligibly small, then a pair of algorithms (T, U_1, U_2) are a fully verifiable delegated implementation of an algorithm Alg.

Definition 5.5: (α, β-Delegated Secure Implementation)
A pair of algorithms (T, U_1, U_2) are an (α, β)-delegated secure implementation of an algorithm Alg if they are both α-efficient and β-verifiable. In particular, a pair of algorithms (T, U_1, U_2) are a fully verifiable $(\alpha, 1)$-delegated secure implementation of an algorithm Alg if they are a fully verifiable delegated implementation of an algorithm Alg.

5.2. Protocols

Pairing outsource protocols can be divided into two depending on the number of the servers that are used to outsource the computation; single server, multiple servers. Then, these categories are further divided into 8 depending on the inputs to the pairing function being secret/public, constant/variable.

5.2.1. Single Server

In this subsection we present the outsourcing protocols that delegates a single pairing computation to a single server.

For the verification step of an authentication or a signature scheme, Girault and Lefranc [2005] give a method for checking the equation

$$e(\sigma, f(\mathcal{I}, m, r)) = e(g, g), \tag{5.2}$$

where $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ is a bilinear map, $\mathbb{G}$ and $\mathbb{G}_1$ are cyclic groups of prime order p, $e(g, g) \neq 1$, f is a public function specific to the scheme, $\mathcal{I}$ is the public parameters including the public key, and (r, σ) is the signature of a message. In order to verify

$e(\sigma, f(\mathcal{I}, m, r)) = e(g, g), C$

i) randomly picks an integer $t \in \mathbb{Z}_p$ and precomputes $\gamma = e(g, g)^t$,

ii) computes $\alpha = (f(\mathcal{I}, m, r))^t$ and sends (σ, α) to S to get $\beta = e(\sigma, \alpha)$,

iii) checks if $\beta = \gamma$.

So, instead of calculating a pairing, the client now computes two exponentiations; one offline and one online.

The scheme of Girault and Lefranc is only able to verify a pairing equation, so it may not be seen as a pairing delegation protocol. In this sense, the first scheme which is able to calculate a pairing $e(A, B)$ directly is given by Chevallier-Mames et al. [2005]. In their scheme, a computationally limited device outsources the computation of $e(A, B)$ to a more powerful device while the powerful device does not learn the points A, B and the limited device is able to detect if powerful device cheats. Now we present their algorithms in the following.

- Generic case with secret A, B
  In order to compute $e(A, B)$ with secret $A \in \mathbb{G}_1, B \in \mathbb{G}_2$, C

  i) generates random $g_1, g_2, a_1, a_2, r_1, r_2 \in \mathbb{Z}_p$ and queries S to get

$$\alpha_1 = e(A + g_1 P, Q), \tag{5.3}$$

$$\alpha_2 = e(P, B + g_2 Q), \tag{5.4}$$

$$\alpha_3 = e(A + g_1 P, B + g_2 Q), \tag{5.5}$$

$$\alpha_4 = e(a_1 A + r_1 P, a_2 B + r_2 Q). \tag{5.6}$$

  ii) checks $\alpha_1, \alpha_2, \alpha_3 \in \mathbb{G}_T$, by checking that $(\alpha_i)^p = 1$ for $i = 1, 2, 3$. Otherwise, the card outputs $\perp$ and halts. Then, computes

$$e_{AB} = \alpha_1^{-g_1} \alpha_2^{-g_2} \alpha_3 e(P, Q)^{g_1 g_2}, \tag{5.7}$$

$$\alpha_4' = e_{AB}^{a_1 a_2} \alpha_1^{a_1 r_2} \alpha_2^{a_2 r_1} e(P, Q)^{r_1 r_2 - a_1 g_1 r_2 - a_2 g_2 r_1} \tag{5.8}$$

and checks that $\alpha'_4 = \alpha_4$. In this case, outputs e_{AB}, otherwise outputs $\perp$.

- secret A, public B
  In order to compute $e(A, B)$ with secret $A \in \mathbb{G}_1$ and public $B \in \mathbb{G}_2$, the protocol is the same as the generic case, except that $g_2 = 0$. C

  i) generates random $g_1, a_1, a_2, r_1, r_2 \in \mathbb{Z}_p$ and queries S to get

$$\alpha_1 = e(A + g_1 P, Q), \tag{5.9}$$

$$\alpha_2 = e(P, B), \tag{5.10}$$

$$\alpha_3 = e(A + g_1 P, B), \tag{5.11}$$

$$\alpha_4 = e(a_1 A + r_1 P, a_2 B + r_2 Q). \tag{5.12}$$

  ii) checks $\alpha_1, \alpha_2, \alpha_3 \in \mathbb{G}_T$, by checking that $(\alpha_i)^p = 1$ for $i = 1, 2, 3$. Otherwise, the card outputs $\perp$ and halts. Computes

$$e_{AB} = \alpha_2^{-g_1} \alpha_3, \tag{5.13}$$

$$\alpha'_4 = e_{AB}^{a_1 a_2} \alpha_1^{a_1 r_2} \alpha_2^{a_2 r_1} e(P, Q)^{r_1 r_2 - a_1 g_1 r_2}, \tag{5.14}$$

  and checks that $\alpha'_4 = \alpha_4$. In this case, outputs e_{AB}, otherwise outputs $\perp$.

- public A, B
  In order to compute $e(A, B)$ with public $A \in \mathbb{G}_1, B \in \mathbb{G}_2$), the protocol is the same as the generic case, except that $g_1 = 0$ and $g_2 = 0$. C

  i) generates random $a_1, a_2, r_1, r_2 \in \mathbb{Z}_p$ and queries S to get

$$\alpha_1 = e(A, Q), \tag{5.15}$$

$$\alpha_2 = e(P, B), \tag{5.16}$$

$$\alpha_3 = e(A, B), \tag{5.17}$$

$$\alpha_4 = e(a_1 A + r_1 P, a_2 B + r_2 Q). \tag{5.18}$$

ii) checks $\alpha_1, \alpha_2, \alpha_3 \in \mathbb{G}_T$, by checking that $(\alpha_i)^p = 1$ for $i = 1, 2, 3$. Otherwise, the card outputs $\perp$ and halts. Computes

$$\alpha_4' = \alpha_3^{a_1 a_2} \alpha_1^{a_1 r_2} \alpha_2^{a_2 r_1} e(P, Q)^{r_1 r_2} \tag{5.19}$$

and checks that $\alpha_4' = \alpha_4$. In this case, outputs α_3, otherwise outputs $\perp$.

- public constant A, public B
  C is given Q_1 and $e(A, Q_1)$ which are kept secret. In order to compute $e(A, B)$ with constant public $A \in \mathbb{G}_1$ and public $B \in \mathbb{G}_2$), C

  i) generates random $r \in \mathbb{Z}_p$ and queries S to get

  $$\alpha_1 = e(A, B), \tag{5.20}$$

  $$\alpha_2 = e(A, rB + Q_1). \tag{5.21}$$

  ii) checks $\alpha_1, \alpha_2 \in \mathbb{G}_T$ by checking that $(\alpha_i)^p = 1$ for $i = 1, 2$ and

  $$\alpha_1^r e(A, Q_1) = \alpha_2. \tag{5.22}$$

  In this case, outputs α_1, otherwise outputs $\perp$.

- secret constant A, public B
  C is given Q_1 and $e(A, Q_1)$ which are kept secret. In order to compute $e(A, B)$ with constant secret $A \in \mathbb{G}_1$ and public $B \in \mathbb{G}_2$), C

  i) generates random $x, y \in \mathbb{Z}_p$ and queries S to get

  $$\alpha_1 = e(xA, B), \tag{5.23}$$

  $$\alpha_2 = e(yA, z(B + Q_1)). \tag{5.24}$$

  ii) computes $e_{AB} = \alpha_1^{x^{-1}}, \quad \alpha_3 = \alpha_2^{(yz)^{-1}}$.

  iii) checks $\alpha_1, \alpha_2 \in \mathbb{G}_T$ by checking that $(\alpha_i)^p = 1$ for $i = 1, 2$ and $e_{AB} e(A, Q_1) = \alpha_3$. In this case, outputs e_{AB}, otherwise outputs $\perp$.

Following Chevallier-Mames et al. [2005], Kang et al. [2005] proposed more efficient solutions. Like Chevallier-Mames et al. [2005], Kang *et al.* give a generic solution than modify it for different cases.

- Generic case with secret A, B
  In order to compute $e(A, B)$ with secret $A \in \mathbb{G}_1, B \in \mathbb{G}_2$), C

  i) generates random $g_1, g_2, r_1, r_2 \in \mathbb{Z}_p$ and queries S to get

$$\alpha_1 = e(g_1 A, Q), \tag{5.25}$$

$$\alpha_2 = e(P, g_2 Q), \tag{5.26}$$

$$\alpha_3 = e(g_1 A, g_2 B), \tag{5.27}$$

$$\alpha_4 = e(A + r_1 P, B + r_2 Q), \tag{5.28}$$

  ii) checks that $\alpha_1, \alpha_2, \alpha_3 \in \mathbb{G}_T$ by checking $\alpha_i^p = 1$. Otherwise, outputs $\perp$ and halts. Computes

$$\alpha_4' = \alpha_3^{g_1 {g_2}^{-1}} \alpha_1^{{g_1}^{-1} r_2} \alpha_2^{{g_2}^{-1} r_1} e(P, Q)^{r_1 r_2}, \tag{5.29}$$

  and checks that $\alpha_4 = \alpha_4'$. In this case, outputs $\alpha_3^{g_1 {g_2}^{-1}}$ as $e(A, B)$; otherwise outputs $\perp$.

- secret A, public B
  In order to compute $e(A, B)$ with private $A \in \mathbb{G}_1$ and public $B \in \mathbb{G}_2$, the protocol is the same as the generic case, except that $g_2 = 1$. C

  i) generates random $g_1, a_2, r_1, r_2 \in \mathbb{Z}_p$ and queries S to get

$$\alpha_1 = e(g_1 A, Q), \tag{5.30}$$

$$\alpha_2 = e(P, Q), \tag{5.31}$$

$$\alpha_3 = e(g_1 A, B), \tag{5.32}$$

$$\alpha_4 = e(A + r_1 P, a_2 B + r_2 Q). \tag{5.33}$$

ii) checks that $\alpha_1, \alpha_2, \alpha_3 \in \mathbb{G}_T$ by checking $\alpha_i^p = 1$. Otherwise, outputs $\perp$ and halts. Computes

$$\alpha_4' = \alpha_3^{g_1^{-1} a_2} \alpha_1^{g_1^{-1} r_2} \alpha_2^{a_2 r_1} e(P, Q)^{r_1 r_2}, \tag{5.34}$$

and checks that $\alpha_4 = \alpha_4'$. In this case, outputs $\alpha_3^{g_2^{-1}}$ as $e(A, B)$; otherwise outputs $\perp$.

- secret constant A, secret B
  C is given $P_1 \in \mathbb{G}_1$, and $e(P_1, B)$ which are kept secret. In order to compute $e(A, B)$ with private $A \in \mathbb{G}_1$ and constant private $B \in \mathbb{G}_2$, C

  i) generates random $r_1, r_2, g_1, g_2 \in \mathbb{Z}_p$ and queries S to get

  $$\alpha_1 = e(A + r_1 P_1, r_2 B), \tag{5.35}$$

  $$\alpha_2 = e(g_1 A, g_2 B). \tag{5.36}$$

  ii) checks $\alpha_1 = \alpha_2^{(g_1 g_2)^{-1} r_2} e(P_1, B)^{r_1 r_2}$ and $\alpha_1 \in \mathbb{G}_T$. If it is satisfied, outputs $\alpha_2^{(g_1 g_2)^{-1}}$ as $e(A, B)$, otherwise outputs $\perp$.

- secret A, public constant B
  C is given $P_1 \in \mathbb{G}_1$, and $e(P_1, B)$ which are kept secret. In order to compute $e(A, B)$ with private $A \in \mathbb{G}_1$ and constant public $B \in \mathbb{G}_2$, C

  i) generates random $r_1, g_1 \in \mathbb{Z}_p$ and queries S to get

  $$\alpha_1 = e(A + r_1 P_1, B), \tag{5.37}$$

  $$\alpha_2 = e(g_1 A, B). \tag{5.38}$$

  ii) checks $\alpha_1 = \alpha_2^{g_1^{-1}} e(P_1, B)^{r_1}$ and $\alpha_2 \in \mathbb{G}_T$. If it is satisfied, outputs $\alpha_2^{g_1^{-1}}$ as $e(A, B)$, otherwise outputs $\perp$.

Canard et al. [2014] argue that the previous solutions Chevallier-Mames et al. [2005] and Kang et al. [2005] for the generic case are more costly than calculating the pairing itself. Then, they propose more efficient solutions and introduce the idea of offline computations for the computations of the values that do not depend on the inputs.

- Generic case with secret A, B
  In order to compute $e(A, B)$ with secret $A \in \mathbb{G}_1$ and secret $B \in \mathbb{G}_2$, C

  i) generates random $x_1, x_2 \in \mathbb{Z}_p$ and pre-computes

$$X_1 = x_1 P, \quad X_2 = x_2 Q, \tag{5.39}$$

$$\chi = e(P, Q)^{x_1 x_2}. \tag{5.40}$$

  ii) generates random $u, v \in \mathbb{Z}_p$, computes

$$A' = uA, \quad B' = vB \tag{5.41}$$

$$T_1 = x_2^{-1} A + X_1, \quad T_2 = x_1^{-1} B + X_2, \tag{5.42}$$

  and queries S to get

$$\alpha_1 = e(T_1, T_2)[e(G_1, B')e(A', G_2)]^{-1}, \tag{5.43}$$

$$\alpha_2 = e(A', B'). \tag{5.44}$$

  iii) checks $\alpha_2 \in \mathbb{G}_T$ and $\alpha_1 = \chi \alpha_2^{(x_1 x_2)^{-1}}$. In this case, outputs $\alpha_2^{(uv)^{-1}}$ as $e(A, B)$. Otherwise, the card outputs $\perp$ and halts.

- public A, public B
  In order to compute $e(A, B)$ with public $A \in \mathbb{G}_1$ and public $B \in \mathbb{G}_2$, the protocol is the same as the generic case, except that $u = 0$ and $v = 0$.

  i) generates random $x_1, x_2 \in \mathbb{Z}_p$ and pre-computes

$$X_1 = x_1 P, \quad X_2 = x_2 Q, \tag{5.45}$$

$$\chi = e(P, Q)^{x_1 x_2}. \tag{5.46}$$

  ii) computes $T_1 = x_2^{-1} A + X_1, \quad T_2 = x_1^{-1} B + X_2$, and queries S to get

$$\alpha_1 = e(T_1, T_2)[e(G_1, B)e(A, G_2)]^{-1}, \tag{5.47}$$

$$\alpha_2 = e(A, B). \tag{5.48}$$

iii) checks $\alpha_2 \in \mathbb{G}_T$ and $\alpha_1 = \chi\alpha_2^{(x_1x_2)^{-1}}$. In this case, outputs α_2 as $e(A, B)$. Otherwise, the card outputs $\perp$ and halts.

- secret A, public constant B
  Canard *et al.* assumes that the client knows $\gamma = e(P, B)$ and this assumption helps them to give much more efficient solution than Kang et al. [2005]. In order to compute $e(A, B)$ with secret $A \in \mathbb{G}_1$ and public constant $B \in \mathbb{G}_2$, C

  i) generates random $x, y \in \mathbb{Z}_p$ and pre-computes

$$X = xP, \quad Y = yP, \tag{5.49}$$

$$\chi_1 = \gamma^x, \quad \chi_2 = \gamma^y. \tag{5.50}$$

  ii) generates random $u \in \mathbb{Z}_p$, computes $T_1 = A + X, \quad T_2 = uA + Y$, and queries S to get

$$\alpha_1 = e(T_1, B), \tag{5.51}$$

$$\alpha_2 = e(T_2, B). \tag{5.52}$$

  iii) checks $\alpha_1 \in \mathbb{G}_T$ and $\alpha_2 = \chi_2(\alpha_1\chi_1^{-1})^u$. In this case, outputs $\alpha_1\chi^{-1}$ as $e(A, B)$. Otherwise, the card outputs $\perp$ and halts.

Protocols given so far, namely Chevallier-Mames et al. [2005]; Kang et al. [2005]; Canard et al. [2014], are verifiable, i.e. outsourcer is able to detect a cheating verifier. Guillevic and Vergnaud argue that this is not always required for encryption primitives where verifiability can be achieved by other means. As a result, they give two protocols which are not verifiable. One of them uses a knapsack-based approach, and the other one delegates only the non-critical steps in the pairing algorithm.

- Secret constant A, public B
  The protocol to outsource $e(A, B)$ with secret constant $A \in \mathbb{G}_1$ and public $B \in \mathbb{G}_2$ is given in 3 stages.

i) C generates random $P_1, P_2, \ldots, P_{n-1} \in \mathbb{G}_1$, chooses random $\sigma_i \in S$ and computes $Q_i = \sigma_i(P_i)$, where S is the set of efficient endomorphisms on the curve. Then, for each Q_i, chooses random $\alpha_i \in \{0, 1, \ldots, 2^{\ell-1}\}$ and pre-computes

$$P_n = A - (\alpha_1 Q_1 + \alpha_2 Q_2 + \ldots + \alpha_{n-1} Q_{n-1}) \tag{5.53}$$

$$= A - \sum_{i=1}^{n-1} \alpha_i Q_i. \tag{5.54}$$

ii) C sends $P_1, P_2, \ldots, P_n, B$ to S to get $f_i = e(Q_i, B)$.

iii) C outputs $e(A, B)$ as

$$(f_1^{\sigma_1})^{\alpha_1} (f_2^{\sigma_2})^{\alpha_2} \ldots (f_{n-1}^{\sigma_{n-1}})^{\alpha_{n-1}} f_n. \tag{5.55}$$

Luo et al. [2018] proposed the following scheme for delegation of pairing computation under OUP model. In order to compute $e(A, B)$, C

i) calls $Rand$ to get

$$a, b, k_1, k_2, aP, bQ, ak_2^{-1}P, bk_1^{-1}Q, e(P, Q)^{-ab(k_1 k_2)^{-1}},$$

where $a, b, k_1, k_2 \in_R \mathbb{Z}_p^*$.

ii) queries S in random order to get:

$$V_1 = e(A + aP, B + bQ), \tag{5.56}$$

$$V_2 = e(k_1 A + ak_2^{-1}P, k_2 B + bk_1^{-1}Q), \tag{5.57}$$

$$V_3 = e(A, B + bQ), \tag{5.58}$$

$$V_4 = e(A + aP, B), \tag{5.59}$$

$$V_5 = e(k_1 A, k_2 B + bk_1^{-1}Q), \tag{5.60}$$

$$V_6 = e(k_1 A + ak_2^{-1}P, k_2 B). \tag{5.61}$$

iii) verifies

$$V_3 \cdot V_4^{-1} \stackrel{?}{=} V_5 \cdot V_6^{-1}. \tag{5.62}$$

iv) If the verification step fails T outputs $\perp$.

v) Else, T outputs

$$e(A,B) = \left(\frac{V_2 \cdot e(P_1,P_2)^{-ab(k_1k_2)^{-1}}}{V_1} \right)^{(k_1k_2-1)^{-1}}. \tag{5.63}$$

Attacks on The Security and Verifiability of Luo et al. [2018]. When we look at the values send to S, we see that they are of the form:

$$V_1 = e(X,K), \quad V_2 = e(Y,L), \tag{5.64}$$

$$V_3 = e(\mathcal{A},K), \quad V_4 = e(X,\mathcal{B}), \tag{5.65}$$

$$V_5 = e(Z,L), \quad V_6 = e(Y,M). \tag{5.66}$$

By looking at the first part of the inputs, S has two possibilities for A either $\mathcal{A}$ or Z. Similarly, B is either $\mathcal{B}$ or M. Hence, $e(A,B)$ is also known.

On the other side, the authors' claim for full verifiability of the scheme in Luo et al. [2018] does not hold either. Since S is able to distinguish the queries for $V_3, V_4, V_5,$ and V_6 as illustrated above, S sends correct values for $V_3, V_4, V_5,$ and V_6 while sending any random value for V_1 and V_2. These V_1 and V_2 are not checked at any step, so the client can not detect that S misbehaves.

Uzunkol et al. [2017] show that the schemeLuo et al. [2018] given by Luo *et al* is 0-checkable and not secure.

5.2.1.1. Comparison

The efficiency comparison of single server pairing outsourcing algorithms presented so far are given in Table 5.1. Since the scheme of Luo et al. [2018] is shown to be broken by Uzunkol et al. [2017], we do not include it in the comparison. The

following values are calculated by computational cost of operations on a BN Curve that satisfies 128-bit level of security.

Table 5.1: Comparison of single server single pairing outsource algorithms.

	Chevallier-Mames et al. [2005]	Kang et al. [2005]	Canard et al. [2014]	Guillevic and Vergnaud [2015]
SVSV	167081	112158	66902	-
SVPV	121904	116906	-	-
PVPV	100310	-	33525	-
PCPV	28635	-	-	-
SCPV	55097	-	-	(n-1)(62+36ℓ)
SVSC	-	83469	-	-
SVPC	-	43177	21767	-

5.2.2. Multiple Servers

All algorithms presented in Section 5.2.2. require scalar multiplication on $\mathbb{G}_1, \mathbb{G}_2$ and/or exponentiation on $\mathbb{G}_T$. These operations are computationally heavy operations, especially exponentiation. By utilizing multiple servers, it is possible to outsource pairing computations without requiring the client to compute scalar multiplication and exponentiation. However, eliminating exponentiation introduces checkability issues. So, one needs to decide when to use single/multiple server depending on the particular situation. It should be also noted that, algorithms given in multi-server setting are all generic algorithms, they work for all type of inputs(public/private, constant/variable).

Chen et al. [2015] proposed the first efficient and secure outsourcing algorithm of bilinear pairings in the one-malicious version of two untrusted program model. However, their scheme is $1/2$-checkable, i.e. a malicious server is able to cheat the client with probability $1/2$.

The algorithm given in Chen et al. [2015] uses a sub-routine called $Rand$ which is responsible for some offline calculations that do not depend on the inputs. We omit the details of $Rand$.

In order to outsource $e(A, B)$, C

i) runs $Rand$ to create three six-tuple

$$(V_1, V_2, v_1V_1, v_2V_2, \lambda = e(v_1V_1, v_2V_2)), \tag{5.67}$$

$$(x_1, X_2, x_1X_1, x_2X_2, e(x_1X_1, x_2X_2)), \tag{5.68}$$

$$(y_1, Y_2, y_1Y_1, y_2Y_2, e(y_1Y_1, y_2Y_2)). \tag{5.69}$$

ii) queries S_1 in random order to get

$$\alpha_1 = e(A + v_1V_1, B + v_2V_2), \tag{5.70}$$

$$\alpha_4 = e(v_1V_1 + v_2V_1, V_2), \tag{5.71}$$

$$\beta_{11} = e(x_1X_1, x_2X_2), \tag{5.72}$$

$$\beta_{12} = e(y_1Y_1, y_2Y_2). \tag{5.73}$$

iii) queries S_2 in random order to get

$$\alpha_2 = e(A + V_1, v_2V_2), \tag{5.74}$$

$$\alpha_3 = e(v_1V_1, B + V_2), \tag{5.75}$$

$$\beta_{21} = e(x_1X_1, x_2X_2), \tag{5.76}$$

$$\beta_{22} = e(y_1Y_1, y_2Y_2). \tag{5.77}$$

iv) checks that $\beta_{11} = \beta_{21}$ and $\beta_{12} = \beta_{22}$. If not, outputs $\perp$ and halts. Otherwise outputs $\alpha_1\alpha_2^{-1}\alpha_3^{-1}\lambda^{-1}\alpha_4$ as $e(A, B)$.

Later Tian et al. [2015] proposed two algorithms. First one is more efficient than Chen et al. [2015] while providing the same checkability, $1/2$. Second one offers verifiable checkability which is inversely proportional to efficiency. Both algorithms use subroutines to compute values that do not depend on the inputs and we omit the details.

- First Algorithm. In order to outsource $e(A, B)$, C

i) calls $RandA$ to get

$$(x_1P, x_3P, x_1x_2^{-1}x_5P, x_7P, x_1^{-1}x_2Q, x_4Q, x_1^{-1}x_6Q, x_8Q, e(P,Q)^{x_7x_8}, e(P,Q)^{x_5+x_6-x_2}).$$

ii) queries S_1 in random orders to get

$$\alpha_1 = e(A + x_1P, B + x_1^{-1}x_2Q), \tag{5.78}$$

$$\alpha_2 = e(x_3P, x_4Q). \tag{5.79}$$

iii) queries S_2 in random orders to get

$$\alpha_1' = e(A + x_1x_2^{-1}x_5P, -x_1^{-1}x_2Q), \tag{5.80}$$

$$\alpha_2' = e(-x_1P, B + x_1^{-1}x_6Q), \tag{5.81}$$

$$\alpha_3' = e(x_3P, x_4Q), \tag{5.82}$$

$$\alpha_4' = e(x_7P, x_8Q). \tag{5.83}$$

iv) checks $\alpha_2 = \alpha_3'$ and $e(P,Q)^{x_7x_8} = \alpha_4'$. If both equations hold, it computes $e(A,B)$ as $\alpha_1\alpha_1'\alpha_2'e(P,Q)^{x_5+x_6-x_2}$. Otherwise it rejects and outputs $\perp$.

- Flexible Version.

  In order to compute $e(A,B)$, C

  i) calls $RandB$ two times to get

  $$(x_1P, x_1x_2^{-1}x_3P, x_1^{-1}x_2Q, x_1^{-1}x_4Q, x_8Q, e(P,Q)^{x_3+x_4-x_2}),$$

  and

  $$(x_1'P, x_1'x_2'^{-1}x_3'P, x_1'^{-1}x_2'Q, x_1'^{-1}x_4'Q, x_8Q, e(P,Q)^{x_3'+x_4'-x_2'})$$

  ii) randomly selects a small integer $t \in \{1, \ldots, s\}$

  iii) queries S_1 in random orders to get

$$\alpha_1 = e(A + x_1 P, B + x_1^{-1} x_2 Q) \tag{5.84}$$

$$\alpha_2 = e(tA + x_1' {x_2'}^{-1} x_3' P, -{x_1'}^{-1} x_2' Q) \tag{5.85}$$

$$\alpha_3 = e(-x_1' P, B + {x_1'}^{-1} x_4' Q) \tag{5.86}$$

iv) queries S_2 in random orders to get

$$\alpha_1' = e(tA + x_1 {x_2}^{-1} x_3 P, -{x_1'}^{-1} x_2 Q) \tag{5.87}$$

$$\alpha_2' = e(A + x_1 {x_2}^{-1} x_3 P, -x_1^{-1} x_2 Q) \tag{5.88}$$

$$\alpha_3' = e(-x_1 P, B + x_1^{-1} x_4 Q) \tag{5.89}$$

v) computes $\beta = \alpha_1 \alpha_2' \alpha_3' e(P, Q)^{x_3 + x_4 - x_2}$ and $\beta' = \alpha_1' \alpha_2 \alpha_3 e(P, Q)^{x_3' + x_4' - x_2'}$

vi) checks $\beta^t = \beta'$ and $\beta \in \mathbb{G}_T$, $\alpha_2 = \alpha_3'$ and $e(P, Q)^{x_7 x_8} = \alpha_4'$. If the equations hold, it computes $e(A, B)$ as $\alpha_1 \alpha_1' \alpha_2' e(P, Q)^{x_5 + x_6 - x_2}$. Otherwise it rejects and outputs $\perp$.

Arabacı et al. [2015] improves the algorithms given by Chen et al. [2015] and Tian et al. [2015] efficieny-wise while decreasing communication cost and server's workload while still offering the same checkability, $1/2$. They propose two algorithms and both use subroutines for offline computations. Again, we omit the details of $Rand1$ and $Rand2$.

- Algorithm 1. In order to outsource $e(A, B)$, C

  i) calls $Rand1$ to get

  $$(x_1 P, 2x_1 P, x_3 P, x_2 Q, -2x_2 Q, x_4 Q, \lambda_1 = e(P, Q)^{2x_1 x_2}, \lambda_2 = e(P, Q)^{x_3 x_4})$$

  ii) queries S_1 in random orders to get

  $$\alpha_1 = e(A + 2x_1 P, -B - 2x_2 Q) \tag{5.90}$$

  $$\alpha_1' = e(x_3 P, x_4 Q) \tag{5.91}$$

  iii) queries S_2 in random orders to get

$$\alpha_2 = e(A + x_1P, B + x_2Q) \tag{5.92}$$

$$\alpha_2' = e(x_3P, x_4Q) \tag{5.93}$$

iv) checks $\alpha_1' = \alpha_2' = \lambda_2$. If the verifications are successful then it outputs $\alpha_1\alpha_2^2\lambda$ as $e(A, B)$. Otherwise, it outputs $\perp$ and halts.

- Algorithm 2. In order to outsource $e(A, B)$, C

i) calls $Rand2$ to get

$$(x_1P, x_3P, x_4P, x_2Q, x_5Q, (x_2 - x_1^{-1}x_2x_3)Q, \lambda = e(P, Q)^{x_4x_5})$$

ii) queries S_1 in random orders to get

$$\alpha_1 = e(A + x_1P, B + x_2Q), \tag{5.94}$$

$$\alpha_1' = e(x_4P, x_5Q). \tag{5.95}$$

iii) queries S_2 in random orders to get

$$\alpha_2 = e(A + x_3P, -x_2Q), \tag{5.96}$$

$$\alpha - 4 = e(-x_1P, B + (x_2 - x_1^{-1}x_2x_3)Q), \tag{5.97}$$

$$\alpha_2' = e(x_4P, x_5Q). \tag{5.98}$$

iv) checks $\alpha_1' = \alpha_2' = \lambda$. If the verifications are successful then it outputs $\alpha_1\alpha_2\alpha_3$ as $e(A, B)$. Otherwise, it outputs $\perp$ and halts.

In 2016, Ren et al. [2016] proposed two algorithms for bilinear pairing outsource. One is for single pairing outsource and the other one aims batch outsource. Here, we give the first one. Ren et al. [2016] claim that their algorithm is 1-checkable, i.e. there is no way a malicious server is able to cheat the client without getting caught. However, it is shown by Uzunkol et al. [2017] that they fail to satisfy their claim. A server is able to cheat the outsourcer with probability at least $1/6$.
In order to outsource $e(A, B)$, C

i) calls $Rand$ and gets

$$a^{-1}, b^{-1}, aP, bQ, a_2P, b_2Q, e(aP, bQ), \tag{5.99}$$

$$e(a_2P, bQ)^{-1}, e(aP, b_2Q)^{-1}, \tag{5.100}$$

$$a_1^{-1}, b_1^{-1}, a_1P, b_1Q, a_3P, b_3Q, e(a_1P, b_1Q), \tag{5.101}$$

$$e(a_3P, b_1Q)^{-1}, e(a_1P, b_3Q)^{-1}, \tag{5.102}$$

where $a, b, a_i, b_i \in_R \mathbb{Z}_p^*$, $1 \leq i \leq 3$.

ii) queries S_1 in random order to get:

$$\alpha_{11} = e(A - aP, B - bQ), \tag{5.103}$$

$$\alpha_{12} = e(A - aP + a_3P, b_1Q), \tag{5.104}$$

$$\alpha_{13} = e(a_1P, B - bQ + b_3Q). \tag{5.105}$$

iii) similarly, queries S_2 in random order to get:

$$\alpha_{21} = e(A - aP + a_1P, B - bQ + b_1Q), \tag{5.106}$$

$$\alpha_{22} = e(A - aP + a_2P, bQ), \tag{5.107}$$

$$\alpha_{23} = e(aP, B - bQ + b_2Q). \tag{5.108}$$

iv) computes

$$\alpha_{12} \cdot e(a_3P, b_1Q)^{-1} = e(A - aP, b_1Q), \tag{5.109}$$

$$\alpha_{13} \cdot e(a_1P, b_3Q)^{-1} = e(a_1P, B - bQ), \tag{5.110}$$

$$\alpha_{22} \cdot e(a_2P, bQ)^{-1} = e(A - aP, bQ), \tag{5.111}$$

$$\alpha_{23} \cdot e(aP, b_2Q)^{-1} = e(aP, B - bQ). \tag{5.112}$$

$$\tag{5.113}$$

v) chooses $t_1, t_2 \in_R \mathbb{Z}_p^*$ and queries S_1 in random order to get:

$$\alpha_{14} = e(A - aP, t_1Q), \tag{5.114}$$

$$\alpha_{15} = e(t_2P, B - bQ). \tag{5.115}$$

vi) similarly, queries S_2 in random order to get:

$$\alpha_{24} = e(A - aP, t_1Q), \tag{5.116}$$

$$\alpha_{25} = e(t_2P, B - bQ). \tag{5.117}$$

vii) verifies

$$\alpha_{14} \stackrel{?}{=} \alpha_{24}, \tag{5.118}$$

$$\alpha_{15} \stackrel{?}{=} \alpha_{25}, \tag{5.119}$$

$$\alpha_{21} \stackrel{?}{=} \alpha_{11} \cdot e(A - aP, b_1Q) \cdot e(a_1P, B - bQ) \cdot e(a_1P, b_1Q) \tag{5.120}$$

If the verification step fails, outputs $\perp$. Else, outputs

$$\alpha_{11} \cdot e(A - aP, bQ), e(aP, B - bQ) \cdot e(aP, bQ) \tag{5.121}$$

as $e(A, B)$.

The above algorithm is shown to be broken by Uzunkol et al. [2017]. Suppose S_1 is a malicious and S_2 is an honest server. Firstly, S_1 could successfully guess the correct positions of α_{1i}, $i = 1, 2, 3$ with probability $1/6$. After a successful guess, U_1 knows a_1P and b_1Q, and could easily compute a_3P by subtracting the first component of α_{12} from the first component of α_{11} as well. Similarly, by subtracting the second component of α_{13} from the second component of α_{11}, the point b_3Q could be computed by S_1. Then, S_1 could compute $e(a_3P, b_1Q)$ and $e(a_1P, b_3Q)$. Moreover, it could compute the values

$$\left(\alpha_{12} \cdot e(a_3P, b_1Q)^{-1}\right)^{-1} = e(A - aP, b_1Q)^{-1}, \tag{5.122}$$

$$\left(\alpha_{13} \cdot e(a_1P, b_3Q)^{-1}\right)^{-1} = e(a_1P, B - bQ)^{-1}. \tag{5.123}$$

Then, S_1 could simply send to C the following bogus values instead of α_{1i}, $i = 1, 2, 3$:

$$\theta_{11} = \alpha_{11} \cdot e(A - aP, b_1Q)^{-1} \cdot e(a_1P, b_3Q)^{-1}, \tag{5.124}$$

$$\theta_{12} = \alpha_{12}^2 \cdot e(a_3P, b_1Q)^{-1}, \tag{5.125}$$

$$\theta_{13} = \alpha_{13}^2 \cdot e(a_1P, b_3Q)^{-1}. \tag{5.126}$$

After receiving the values θ_{1i}, α_{2i}, $i = 1, 2, 3$, the delegator C computes the following values following the scheme specification:

$$\theta_{12} \cdot e(a_3P, b_1Q)^{-1} = e(A - aP, 2b_1Q), \tag{5.127}$$

$$\theta_{13} \cdot e(a_1P, b_3Q)^{-1} = e(2a_1P, B - bQ), \tag{5.128}$$

$$\alpha_{22} \cdot e(a_2P, bQ)^{-1} = e(A - aP, bQ), \tag{5.129}$$

$$\alpha_{23} \cdot e(aP, b_2Q)^{-1} = e(aP, B - bQ). \tag{5.130}$$

In the second round, the malicious server S_1 could send

$$\theta_{14} = \alpha_{14}^2 = e(A - aP, 2t_1Q), \tag{5.131}$$

$$\theta_{15} = \alpha_{15}^2 = e(2t_2P, B - bQ), \tag{5.132}$$

instead of α_{14} and α_{14}, respectively. Note that S_1 could manipulate α_{14} and α_{15} with θ_{14} and θ_{15} with probability 1 since only squares are taken which are independent of the correct positions of α_{14} and α_{15}. Then, following the protocol honestly, the second server S_2 would compute

$$U_2(e(A - aP, 2b_1Q), t_1b_1^{-1}) \leftarrow \alpha_{24} = e(A - aP, 2t_1Q), \tag{5.133}$$

$$U_2(e(2a_1P, B - bQ), t_2a_1^{-1}) \leftarrow \alpha_{25} = e(2t_2P, B - bQ), \tag{5.134}$$

implying that

$$\theta_{14} = \alpha_{24}, \tag{5.135}$$

$$\alpha_{21} = \theta_{11} \cdot e(A - aP, 2b_1Q) \cdot e(2a_1P, B - bQ) \cdot e(a_1P, b_1Q). \tag{5.136}$$

After passing the verification step with these bogus values, the output would be the bogus value

$$e(A, B)e(A-aP, b_1Q)^{-1}e(a_1P, b_3Q)^{-1} = \theta_{11}\cdot e(A-aP, bQ), e(aP, B-bQ)\cdot e(aP, bQ).$$

instead of $e(A, B)$. Note that the values $e(A - aP, b_1Q)$ and $e(a_1P, B - bQ)$ are computed by the honest server S_2, hence these values remain unchanged.

This attack shows that the scheme in Ren et al. [2016] does not satisfy the full verifiability claim and it is a scheme in which a malicious S_1 could pass the verification step with bogus values with probability at least $1/6$. Hence, S_1 could manipulate the output with probability at least $1/6$.

In 2017, Ren et al. [2017] proposed a delegation protocol utilizing multiple servers and claimed that their scheme's checkability is almost 1. However, as shown by Uzunkol et al. [2017], their scheme fails to satisfy this claim and Tian et al. [2015]'s second scheme is much more efficient for the same checkability level. The scheme is given below.

i) C calls $Rand$ to get

$$t_1,\ t_2,\ a_1P + a_2P,\ a_3P,\ a_4P,\ b_1P + b_2P,\ b_3P, \tag{5.137}$$

$$-(a_1P + a_2P + b_3P),\ -(t_2a_1P + a_2P),\ -(a_1P + t_2a_2P), \tag{5.138}$$

$$a_1Q + a_2Q,\ a_3Q,\ b_1Q + b_2Q,\ b_3Q,\ b_4Q,\ -(b_1Q + b_2Q + a_3Q), \tag{5.139}$$

$$-(t_1b_1Q + b_2Q),\ -(b_1Q + t_1b_2Q),\ e(a_3P, a_3Q), \tag{5.140}$$

$$e(b_3P, b_3Q),\ e(a_4P, b_1Q + b_2Q)^{t_1+1}, \tag{5.141}$$

$$e(a_1P + a_2P, b_4Q)^{t_2+1},\ e(a_1P + a_2P, b_1Q + b_2Q)^{-1}, \tag{5.142}$$

where $a_i, b_i \in_R \mathbb{Z}_q^*$, $1 \leq i \leq 4$, $t_j \in \{2, 3, \cdots, s\}$, and $s \in \mathbb{N}$ is a small number.

ii) C queries S_1 in random order to get:

$$\alpha_{11} = e(A + a_1P + a_2P, B + b_1Q + b_2Q), \tag{5.143}$$

$$\alpha_{12} = e(A + b_1P + b_2P, a_3Q), \tag{5.144}$$

$$\alpha_{13} = e(-(a_1P + a_2P + b_3P), B + b_3Q), \tag{5.145}$$

$$\alpha_{14} = e(A + a_4P, -(t_1b_1Q + b_2Q)), \tag{5.146}$$

$$\alpha_{15} = e(-(t_2a_1P + a_2P), B + b_4Q). \tag{5.147}$$

iii) Then similarly, C queries S_2 in random order to get:

$$\alpha_{21} = e(A + a_1P + a_2P, B + b_1Q + b_2Q), \tag{5.148}$$

$$\alpha_{22} = e(A - a_3P, -(b_1Q + b_2Q + a_3Q)), \tag{5.149}$$

$$\alpha_{23} = e(b_3P, B + a_1Q + a_2Q), \tag{5.150}$$

$$\alpha_{24} = e(A + a_4P, -(b_1Q + t_1b_2Q)), \tag{5.151}$$

$$\alpha_{25} = e(-(a_1P + t_2a_2P)), B + b_4Q). \tag{5.152}$$

iv) Upon receiving computation results from both servers, C checks if

$$\alpha_{11} \stackrel{?}{=} \alpha_{21}, \tag{5.153}$$

$$(\alpha_{12}\alpha_{22}e(a_3P, a_3Q))^{t_1+1} \stackrel{?}{=} \alpha_{14}\alpha_{24}e(a_4P, b_1Q + b_2Q)^{t_1+1}, \tag{5.154}$$

$$(\alpha_{13}\alpha_{23}e(b_3P, b_3Q))^{t_2+1} \stackrel{?}{=} \alpha_{15}\alpha_{25}e(a_1P + a_2P, b_4Q)^{t_2+1}. \tag{5.155}$$

If the check is not successful, outputs $\perp$. Otherwise, outputs

$\alpha_{11}\alpha_{22}e(a_3P_3, a_3Q)\alpha_{13}\alpha_{23}e(b_3P, b_3Q)e(a_1P + a_2P, b_1Q + b_2Q)^{-1}$ as $e(A, B)$.

Despite the claims of Ren et al. [2017], Uzunkol et al. [2017] showed that their scheme failed to satisfy this claim. Assume that the server S_1 is malicious. The probability that S_1 chooses one of the pairs $(\alpha_{12}, \alpha_{14})$ or $(\alpha_{13}, \alpha_{15})$ out of 10 pairs from 5 random queries is at least $1/5$. Then, S_1 can guess the right position of α_{12} or α_{13} with probability at least $1/2$. Moreover, S_1 (resp. S_2) could correctly guess the value of the right exponent t_i with probability $1/(s-1)$, $i = 1, 2$. Hence, a malicious server S_1 (resp. S_2) could correctly guess $(\alpha_{12}, \alpha_{14})$ or $(\alpha_{13}, \alpha_{15})$ with the correct exponent t_i with probability at least $1/10(s-1)$. Assume without loss of generality that $(\alpha_{12}, \alpha_{14})$ is the correctly guessed pair with the exponent t_1. Then, the server S_1 could send the bogus values with using an arbitrary element $\theta \in \mathbb{G}_3$, $\gamma_{12} = \alpha_{12} \cdot \theta, \gamma_{14} = \alpha_{14} \cdot \theta^{t_1+1}$, to the delegator C which successfully enable S_1 to cheat the delegator C, and pass the verification step with probability at least $1/10(s-1)$. Moreover, since after the verification step, the value α_{12} (resp. α_{13}) is also used to recover $e(A, B)$, the output yields to a bogus value instead of $e(A, B)$ with probability at least $1/10(s-1)$. It

is obvious that the same simple attack strategy could be used to manipulate α_{22} or α_{23} and α_{24} or α_{25} for S_2. For $s = 4$, the verification step in Ren et al. [2017] is successful with probability $1 - 1/10 \cdot 3 \approx 0.967$ instead of the author's claim with probability ≈ 0.999. Now, if the verification probabilities of Tian et al. [2015] and Ren et al. [2017] are chosen to be the same (i.e. by choosing appropriate values for the adjustable verification probabilities in both schemes), then it can be seen that the proposed scheme in Ren et al. [2017] has almost no efficiency benefit when carefully compared with the Tian et al.'s scheme Tian et al. [2015]. Additionally, it has worse communication overhead than Tian et al. [2015] (with 10 calls to the servers instead of 6 calls in Tian et al. [2015]). Hence, the scheme in Ren et al. [2017] is less practical than the scheme in Tian et al. [2015] with almost no computational advantages and requirement of additional bandwidth.

In 2017, Dong et al. [2017] proposed a scheme and claimed that it is the first fully verifiable secure delegation scheme. However, Uzunkol et al. [2017] show that their scheme is not secure and a malicious server is able to learn A, B, and $(e(A, B))$. In order to outsource $e(A, B)$, C

i) calls $Rand$ and gets

$$a_1P,\ a_2P,\ a_3P,\ a_4P,\ a_5P,\ a_6P,\ a_7P, \tag{5.156}$$

$$b_1Q,\ b_2Q,\ b_3Q,\ b_4Q,\ b_5Q,\ b_6Q,\ b_7Q, \tag{5.157}$$

$$e(a_1P, b_1Q)^{-1},\ e(a_2P, (b_4 + b_6)Q)^{-1},\ r_i, r'_i, i \in \{4, 5, 6, 7\}, \tag{5.158}$$

$$e(a_3P, (b_5 + b_7)Q)^{-1},\ e((a_4 + a_6)P, b_2Q)^{-1},\ e((a_5 + a_7)P, b_3Q)^{-1}, \tag{5.159}$$

where $a_j, b_j \in_R \mathbb{Z}_q^*$, $1 \leq j \leq 7$, $r_i, r'_i \in \{\pm 1, \pm 2, \pm 4\}$, and

$$r_4b_4 = r_6b_6, \quad r_5b_5 = r_7b_7, \tag{5.160}$$

$$r'_4a_4 = r'_6a_6, \quad r'_5a_5 = r'_7a_7, \tag{5.161}$$

$$\sum_{j=4}^{7} b_j = -b_1, \quad \sum_{j=4}^{7} a_j = -a_1. \tag{5.162}$$

ii) queries S_1 in random order to get:

$$\theta_{11} = e(A + a_1P, B + b_1Q), \tag{5.163}$$

$$\alpha_{11} = e(A + a_2P, b_4Q), \tag{5.164}$$

$$\alpha_{12} = e(A + a_3P, b_5Q), \tag{5.165}$$

$$\beta_{11} = e(a_4P, B + b_2Q), \tag{5.166}$$

$$\beta_{12} = e(a_5P, B + b_3Q). \tag{5.167}$$

iii) similarly, queries S_2 in random order as follows:

$$\theta_{21} = e(A + a_1P, B + b_1Q), \tag{5.168}$$

$$\alpha_{21} = e(A + a_2P, b_6Q), \tag{5.169}$$

$$\alpha_{22} = e(A + a_3P, b_7Q), \tag{5.170}$$

$$\beta_{21} = e(a_6P, B + b_2Q), \tag{5.171}$$

$$\beta_{22} = e(a_7P, B + b_3Q). \tag{5.172}$$

iv) performs the verification step using θ_{11}, θ_{12}, α_{ij}, β_{ij}, $1 \leq i, j \leq 2$, and after successful verification computes $e(A, B)$ by multiplying these values with some precomputed values. We refer to Dong et al. [2017] for the details of the verification step and the computation of $e(A, B)$.

The following attack is given by Uzunkol et al. [2017] which shows that the presented algorithm is not secure and the server is able to learn both the inputs and output of the pairing function with probability 1. Suppose that S_1 is malicious. We will show that S_1 is able to narrow down the possible A values to 300. S_1 is given $A + a_1P$, a_4P, and a_5P. We also know that

$$-a_1 = a_4 + a_5 + a_6 + a_7 + \tag{5.173}$$

$$a_4 + a_5 + (r'_4/r'_6)a_4 + (r'_5/r'_7)a_5. \tag{5.174}$$

So, $-a_1P = a_4P + a_5P + (r'_4/r'_6)a_4P + (r'_5/r'_7)a_5P$. (r'_4/r'_6) and (r'_5/r'_7) can only take values in $\{\pm 1/4, \pm 1/2, \pm 1, \pm 2, \pm 4\}$. We have five possible values for $A + a_1P$, 4 for a_4P, and 3 for a_5P. We further have 10 possibilities for both $(r'_4/r'_6)a_4P$ and $(r'_5/r'_7)a_5P$. So, in total S_1 has 6000 possible values that A can take. Similarly, S_1 has 6000 possible values for B, and is able to guess $e(A, B)$ with probability $1/36.10^6$. S_2 is also able to mount the same attack. Hence, the scheme in Dong et al. [2017] is completely insecure in its current form.

The only possible way of having a secure version of the scheme in Dong et al. [2017] seems to choose r_i and r'_i such that bit-lengths of r_i and r'_i are long enough, e.g. at least 76-bits for 80-bits security level. On the other side, this would make the scheme totally inefficient. In other words, the scheme would have a huge computational overhead when compared to the direct computation of $e(A, B)$ by the client C.

In 2017, Kalkar et al. [2018] falsely claimed that they provide a verifiable and efficient scheme which is given in the following.

In order to outsource $e(A, B)$, C

i) calls $Rand$ to get

$$\alpha, \beta, x, y, m, n, \alpha P, xP, yP, \alpha P - xP, \alpha P - yP, mQ, nQ, \beta Q, e(\alpha P, \beta Q),$$

where $\alpha, \beta, x, y, m, n \in_R \mathbb{Z}_p^*$.

ii) queries S_1 in random order to get

$$A_{11} = e(A - \alpha P, mQ), \tag{5.175}$$

$$A_{12} = e(A - \alpha P, B - nQ), \tag{5.176}$$

$$A_{13} = e(\alpha P - xP, B - \beta Q), \tag{5.177}$$

$$A_{14} = e(yP, B - \beta Q). \tag{5.178}$$

iii) Similarly, queries S_2 in random order to get:

$$A_{21} = e(A - \alpha P, nQ), \tag{5.179}$$

$$A_{22} = e(A - \alpha P, B - mQ), \tag{5.180}$$

$$A_{23} = e(\alpha P - yP, B - \beta Q), \tag{5.181}$$

$$A_{24} = e(xP, B - \beta Q). \tag{5.182}$$

iv) verifies

$$A_{11}A_{22} \stackrel{?}{=} A_{21}A_{12}, \tag{5.183}$$

$$A_{13}A_{24} \stackrel{?}{=} A_{23}A_{14}. \tag{5.184}$$

v) If the verification step fails, outputs $\perp$. Else, C outputs $e(A, B)$ as

$$A_{11}A_{22}A_{13}A_{24}e(P_1, P_2)^{\alpha\beta}. \tag{5.185}$$

Again, as shown by Uzunkol et al. [2017] the scheme fails to fulfill its verifiability claim and a server is able to cheat the client with probability 1. A malicious server S_1 could send the bogus values CA_{1i} instead of A_{1i}, $i = 1, 2, 3, 4$, and could successfully pass the verification step always, i.e. with probability 1. Then, the delegator computes the bogus output $C^2e(A, B)$ instead of $e(A, B)$. Obviously, a malicious server S_2 could also mount a similar attack and could always pass the verification step with bogus values. This fairly simple attack shows that the the claim of having a fully verifiable scheme is unfortunately false. In particular, no verifiability is provided in Kalkar et al. [2018].

5.3. Proposed Scheme

In this section, we first propose an efficient fully verifiable secure partial delegation scheme for the precomputation step Rand. Secondly, we introduce VerPair which is a fully verifiable efficient secure delegation scheme for general pairing

computation under the OMTUP assumption.

5.3.1. Rand: Proposed Scheme's Precomputation Step

The precomputation scheme Rand consists of a precomputation step realized by one of the existing techniques Brickell et al. [1993]; Boyko et al. [1998]; Wang et al. [2014]. The other part consists of a delegation scheme. Before initializing the subroutine Rand, a global security parameter κ is chosen which outputs the *global parameters*

i) the prime number q,

ii) the groups $(\mathbb{G}_1, +)$, $(\mathbb{G}_2, +)$, and $(\mathbb{G}_3, \cdot)$ of order q,

iii) the pairing map $e: \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_3$,

iv) the generators P_1, P_2, and $g := e(P_1, P_2)$ of $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$, respectively.

Together with these global parameters, the *static* tuples

$$(\alpha_1, \alpha_1 P_1, \alpha_1 P_2), (\alpha_2, \alpha_2 P_1, \alpha_2 P_2) \tag{5.186}$$

are computed at the initialization of the subroutine Rand, and loaded to the delegator T (by a trusted party, e.g. by using HSM, TPM, etc.), where α_1 and α_2 are random elements in $\mathbb{Z}_q^*$. Note that $(\alpha_1, \alpha_1 P_1, \alpha_1 P_2)$ is *secret* and *protected* from U_2, but not necessarily from U_1, and $(\alpha_2, \alpha_2 P_1, \alpha_2 P_2)$ is *secret* and *protected* from U_1, but not necessarily from U_2. Rand takes no input except global parameters and static tuples (5.186).

After calling Rand, the first part chooses random values $a, b, s, , t \in_R \mathbb{Z}_q^*$ and outputs (a, aP_1), (b, bP_2), (s, sP_1, sP_2) and (t, g^t).

In the delegated second part of Rand, the delegator T chooses first randomly $t_1, t_2, c_1, c_2 \in_R \mathbb{Z}_q^*$. Then,

i) T queries U_1 in random order as follows:

$$U_1(a \cdot b - t, g) \leftarrow \gamma_1 = g^{ab-t}, \quad U_1(t_1 \cdot s - t, g) \leftarrow \gamma_2 = g^{t_1 s - t}, \tag{5.187}$$

$$U_1(t_2 \cdot s - t, g) \leftarrow \gamma_3 = g^{t_2 s - t}, \quad U_1(t_1 - \alpha_1, P_1) \leftarrow \gamma_4 = (t_1 - \alpha_1)P_1, \tag{5.188}$$

$$U_1(t_1 - \alpha_1, P_2) \leftarrow \gamma_5 = (t_1 - \alpha_1)P_2, \quad U_1(t_2 - \alpha_2, P_1) \leftarrow \gamma_6 = (t_2 - \alpha_2)P_1, \tag{5.189}$$

$$U_1(t_2 - \alpha_2, P_2) \leftarrow \gamma_7 = (t_2 - \alpha_2)P_2, \quad U_1(c_1 \cdot t_1, \cdot^{-1}) \leftarrow \theta_{11} = c_1^{-1} t_1^{-1}, \tag{5.190}$$

$$U_1(c_2 \cdot t_2, \cdot^{-1}) \leftarrow \theta_{12} = c_2^{-1} t_2^{-1}. \tag{5.191}$$

ii) T queries U_2 in random order as follows:

$$U_2(ab - t, g) \leftarrow \beta_1 = g^{ab-t}, \quad U_2(t_1 s - t, g) \leftarrow \beta_2 = g^{t_1 s - t}, \tag{5.192}$$

$$U_2(t_2 s - t, g) \leftarrow \beta_3 = g^{t_2 s - t}, \quad U_2(t_1 - \alpha_1, P_1) \leftarrow \beta_4 = (t_1 - \alpha_1)P_1, \tag{5.193}$$

$$U_2(t_1 - \alpha_1, P_2) \leftarrow \beta_5 = (t_1 - \alpha_1)P_2, \quad U_2(t_2 - \alpha_2, P_1) \leftarrow \beta_6 = (t_2 - \alpha_2)P_1, \tag{5.194}$$

$$U_2(t_2 - \alpha_2, P_2) \leftarrow \beta_7 = (t_2 - \alpha_2)P_2, \quad U_2(c_1 t_1, \cdot^{-1}) \leftarrow \theta_{21} = c_1^{-1} t_1^{-1}, \tag{5.195}$$

$$U_2(c_2 t_2, \cdot^{-1}) \leftarrow \theta_{22} = c_2^{-1} t_2^{-1}. \tag{5.196}$$

iii) After receiving γ_i, θ_{1j} from U_1 and β_i, θ_{2j} from U_2, $1 \leq i \leq 7, \overset{\leq}{} j \leq 2$, T verifies

$$\gamma_i \overset{?}{=} \beta_i \text{ and } \theta_{1j} \overset{?}{=} \theta_{2j}. \tag{5.197}$$

iv) If Equations (5.197) do not hold simultaneously, then T outputs $\perp$.

v) Else, T outputs

$$((a, aP_1),\ (b, bP_2),\ (s, sP_1, sP_2),\ (t, g^t), (t_1,\ t_1P_1 = \gamma_4 + \alpha_1 P_1,\ t_1P_2 = \gamma_5 + \alpha_1 P_2),$$

$$(t_2,\ t_2P_1 = \gamma_6 + \alpha_2 P_1, t_2P_2 = \gamma_7 + \alpha_2 P_2), g^{ab} = \gamma_1 \cdot g^t, g^{t_1 s} = \gamma_2 \cdot g^t, g^{t_2 s} = \gamma_3 \cdot g^t,$$

$$t_1^{-1} = c_1 \cdot \theta_{11}, t_2^{-1} = c_2 \cdot \theta_{12}, a \cdot t_1^{-1}, a \cdot t_2^{-1}, b \cdot t_1^{-1}, b \cdot t_2^{-1}).$$

Remark 5.1: We note that the outputs of the delegated part of Rand is always secret or (honest/adversarial) protected except possibly (t_1, t_1P_1, t_1P_2) *from* U_1 *and* (t_2, t_2P_1, t_2P_2) *from* U_2*. We refer to Section 5 for the further details. Furthermore, we here give a simple but more efficient two-server version for secure delegation of the modular inversion* $t^{-1} \bmod q$ *which is first introduced in Cavallo et al. [2015].*

5.3.2. VerPair: A Fully Verifiable Secure Delegation Scheme

The Rand scheme outputs the following values

$$((a, aP_1),\ (b, bP_2),\ (s_1, s_1P_1),\ (s_2, s_2P_2),\ (t, g^t),$$

$$(t_1,\ t_1P_1,\ t_1P_2), (t_2,\ t_2P_1,\ t_2P_2), g^{ab},\ g^{t_1 s},\ g^{t_2 s},$$

$$t_1^{-1},\ t_2^{-1},\ at_1^{-1},\ at_2^{-1},\ bt_1^{-1},\ bt_2^{-1}),$$

where P_1 is the generator of $\mathbb{G}_1$ (of prime order q) and P_2 is the generator of $\mathbb{G}_2$ (of prime order q) and $a, b, t_1, t_2, s \in_R \mathbb{Z}_q^*$.

Let T denote the delegator and U_1 and U_2 be the two untrusted servers. The inputs of VerPair are the outputs of Rand scheme and the secret, private inputs $A \in \mathbb{G}_1$ and $B \in \mathbb{G}_2$.

The steps of VerPair are given as follows:

i) T queries U_1 in random order as follows:

$$U_1(t_1P_1, B - bP_2 - sP_2) \leftarrow D_{11} = e(t_1P_1, B - bP_2 - sP_2), \tag{5.198}$$

$$U_1(A - aP_1 - sP_1, t_1P_2) \leftarrow D_{12} = e(A - aP_1 - sP_1, t_1P_2). \tag{5.199}$$

ii) Then similarly, T queries U_2 in random order as follows:

$$U_2(t_2P_1, B - bP_2 - sP_2) \leftarrow D_{21} = e(t_2P_1, B - bP_2 - sP_2), \tag{5.200}$$

$$U_2(A - aP_1 - sP_1, t_2P_2) \leftarrow D_{22} = e(A - aP_1 - sP_1, t_2P_2). \tag{5.201}$$

iii) After receiving D_{11}, D_{12} from U_1 and D_{21}, D_{22} from U_2, T computes

$$\beta_1 = D_{11} \cdot g^{t_1 s} = e(t_1P_1, B - bP_2), \tag{5.202}$$

$$\beta_2 = D_{12} \cdot g^{t_1 s} = e(A - aP_1, t_1P_2), \tag{5.203}$$

$$\beta_3 = D_{21} \cdot g^{t_2 s} = e(t_2P_1, B - bP_2), \tag{5.204}$$

$$\beta_4 = D_{22} \cdot g^{t_2 s} = e(A - aP_1, t_2P_2). \tag{5.205}$$

iv) Then, T queries U_1 in random order as follows:

$$U_1(\beta_3, at_2^{-1}) \leftarrow D_{13} = \beta_3^{at_2^{-1}} = e(aP_1, B - bP_2), \tag{5.206}$$

$$U_1(\beta_4, bt_2^{-1}) \leftarrow D_{14} = \beta_4^{bt_2^{-1}} = e(A - aP_1, bP_2), \tag{5.207}$$

$$U_1(A - aP_1, B - bP_2) \leftarrow D_{15} = e(A - aP_1, B - bP_2). \tag{5.208}$$

v) Similarly, T queries U_2 in random order as follows:

$$U_2(\beta_1, at_1^{-1}) \leftarrow D_{23} = \beta_1^{at_1^{-1}} = e(aP_1, B - bP_2), \tag{5.209}$$

$$U_2(\beta_2, bt_1^{-1}) \leftarrow D_{24} = \beta_2^{bt_1^{-1}} = e(A - aP_1, bP_2), \tag{5.210}$$

$$U_2(A - aP_1, B - bP_2) \leftarrow D_{25} = e(A - aP_1, B - bP_2). \tag{5.211}$$

vi) T verifies

$$D_{13} \stackrel{?}{=} D_{23},\ D_{14} \stackrel{?}{=} D_{24},\ D_{15} \stackrel{?}{=} D_{25}. \tag{5.212}$$

vii) If Equations (5.212) do not hold simultaneously, then T outputs $\perp$.

viii) Else, T outputs $e(A, B) = D_{13} \cdot D_{14} \cdot D_{15} \cdot g^{ab}$.

5.4. Security & Performance Analysis

In this section, we analyze the security, verifiability and efficiency properties of the delegated part of $Rand$ and VerPair. Moreover, we compare VerPair with the previous proposals with respect to its computational and communication complexities for both the delegator T and the services U_i, $i = 1, 2$, i.e. the overall communication overhead, number of rounds, memory requirements of the delegator T, and the computational complexity for T.

Theorem 5.1: In the one-malicious version of a two-untrusted program model (OMTUP), the algorithms (T, U_1, U_2) are a fully verifiable $\mathcal{O}(\frac{1}{\log q})$-efficient delegated-secure implementations of the delegated part of Rand scheme, where the static inputs

$$(\alpha_1, \alpha_1 P_1, \alpha_1 P_2), (\alpha_2, \alpha_2 P_1, \alpha_2 P_2)$$

of Rand maybe honest, secret; or $(\alpha_1, \alpha_1 P_1, \alpha_1 P_2)$ honest, unprotected from U_1; or $(\alpha_2, \alpha_2 P_1, \alpha_2 P_2)$ honest, unprotected from U_2.

Proof . We prove completeness, security, full verifiability and efficiency of Rand as follows:

Completeness. Assume that the servers U_1 and U_2 run Rand honestly. Since we delegate the same computations to both U_1 and U_2, we clearly have

$$\gamma_i = \beta_i,\ \theta_{1j} = \theta_{2j}, \textit{for } 1 \leq i \leq 7, j = 1, 2. \tag{5.213}$$

Now, we show that the following equalities hold:

$$\gamma_1 \cdot g^t = g^{ab-t} g^t = g^{ab} \tag{5.214}$$

$$, \gamma_2 \cdot g^t = g^{t_1 s - t} g^t = g^{t_1 s}, \tag{5.215}$$

$$\gamma_3 \cdot g^t = g^{t_2 s - t} g^t = g^{t_2 s}, \tag{5.216}$$

$$\gamma_4 + \alpha_1 P_1 = (t_1 - \alpha_1) P_1 + \alpha_1 P_1 = t_1 P_1, \tag{5.217}$$

$$\gamma_5 + \alpha_1 P_2 = (t_1 - \alpha_1) P_2 + \alpha_1 P_2 = t_1 P_2, \tag{5.218}$$

$$\gamma_6 + \alpha_2 P_1 = (t_2 - \alpha_2) P_1 + \alpha_2 P_1 = t_2 P_1, \tag{5.219}$$

$$\gamma_7 + \alpha_2 P_2 = (t_2 - \alpha_2) P_2 + \alpha_2 P_2 = t_2 P_2, \tag{5.220}$$

$$c_1 \theta_{11} = c_1 (c_1^{-1} t_1^{-1}) = t_1^{-1}, \tag{5.221}$$

$$c_2 \theta_{12} = c_2 (c_2^{-1} t_2^{-1}) = t_2^{-1}. \tag{5.222}$$

Since the values $a, b, t \in \mathbb{Z}_q^*$ with $(aP_1, bP_2, sP_1, sP_1, g^t)$ are computed in the first part of Rand using a precomputation subroutine, and the outputs at_i^{-1}, bt_i^{-1}, $i = 1, 2$, are solely computed by T itself, we are also done with completeness of Rand.

Security & Full verifiability. The proof is similar to Hohenberger and Lysyanskaya [2005]. We assume now that $\mathcal{A} = (E, U_1', U_2')$ is a probabilistic polynomial-time (PPT) adversary interacting with a PPT-based algorithm T in the delegated-security model of Section (2). Our fist claim is $EVIEW_{real}$~$EVIEW_{ideal}$, Note that all static inputs $(\alpha_1, \alpha_1 P_1, \alpha_1 P_2)$, $(\alpha_2, \alpha_2 P_1, \alpha_2 P_2)$ are assumed to be honest, secret for an environmental adversary since neither E and U_1' nor E and U_2' cannot communicate directly to develop a joint strategy after interacting with T. Then, ignoring the ith round, a simulator S_1 first chooses elements x_i, $1 \leq i \leq 9$ randomly, and makes 18 random queries to U_1' and U_2'

$$U_1'(x_i, P_1) \leftarrow \gamma_i, \quad U_2'(x_i, P_1) \leftarrow \beta_i \textit{for} i = 4, 6, \tag{5.223}$$

$$U_1'(x_i, P_2) \leftarrow \gamma_i, \quad U_2'(x_i, P_2) \leftarrow \beta_i \textit{for} i = 5, 7, \tag{5.224}$$

$$U_1'(x_i, g) \leftarrow \gamma_i, \quad U_2'(x_i, g) \leftarrow \beta_i \textit{for} i = 1, 2, 3, \tag{5.225}$$

$$U_1'(x_8, \cdot^{-1}) \leftarrow \theta_{11}, \quad U_2'(x_8, \cdot^{-1}) \leftarrow \theta_{21}, \tag{5.226}$$

$$U_1'(x_9, \cdot^{-1}) \leftarrow \theta_{12}, \quad U_2'(x_9, \cdot^{-1}) \leftarrow \theta_{22}. \tag{5.227}$$

Note that we do not consider the outputs at_i^{-1}, bt_i^{-1}, $i = 1, 2$, to prove the result since it is solely computed by T without any interaction with E, U_1' or U_2'. Then, S_1 behaves

- *if the outputs of U_1' and U_2' are not equal for a randomly selected i, $1 \leq i \leq 9$, then the values $Y_p^i =" error"$, $Y_u^i = \emptyset$, and $replace^i = 1$ (corresponding to the output $(estate^i, " error", \emptyset)$ in the ideal process) are produced by S_1,*
- *if no $"error"$ is detected , then the values $Y_p^i = \emptyset$, $Y_u^i = \emptyset$, and $replace^i = 0$ (corresponding to the output $(estate^i, Y_p^i, Y_u^i)$ in the ideal process) are produced by S_1,*
- *otherwise, S_1 selects a random element r and outputs $Y_p^i = r$, $Y_u^i = \emptyset$, and $replace^i = 1$ (corresponding to the output $(estate^i, r, Y_u^i)$ in the ideal process).*

In either cases, S_1 saves the appropriate states.

The distributions of inputs in the real and ideal experiments are computationally indistinguishable. In the ideal experiment, the inputs are chosen uniformly at random. In the real experiment, all inputs of the delegated part of Rand are independently randomized by the choice of uniformly distributed random elements $t_1, t_2, c_1, c_2 \in_R \mathbb{Z}_q^*$. Note that, by each invocation of Rand, new random values are generated which are different from other invocations. Then, there are two cases

- *if U_1' and U_2' behave honestly both in the real and the ideal experiments in the round i, then we have $EVIEW_{real}^i$~$EVIEW_{ideal}^i$ since in the real execution $T^{U_1', U_2'}$ perfectly runs Rand and in the ideal execution S_1 does not change the output,*
- *If one of U_1' or U_2' behaves dishonestly in the round i, than this can be detected by both T and S_1 with probability 1. The reason is that one server is always honest under OMTUP, and only the equality of the same delegated inputs are compared coming from an honest and a potentially dishonest server. Then, any misbehavior*

could always be detected, and this will result an output of an "$error$". This argument also shows that Rand is fully verifiable.

Note that it is impossible that Rand could be corrupted implying that S_1 never executes the case of selecting a random element r and returning $Y_p^i = r$, $Y_u^i = \emptyset$, and $replace^i = 1$ in the ideal experiment since Rand is fully verifiable, hence it is impossible for both U_1' and U_2' to deviate from their functionalities. Thus, we have

$$EVIEW_{real}^i \sim EVIEW_{ideal}^i$$

even in the case of a dishonest server U_1' or U_2'. By the hybrid argument, we conclude that

$$EVIEW_{real} \sim EVIEW_{ideal}.$$

Secondly, we claim that $UVIEW_{real} \sim UVIEW_{ideal}$, i.e. Pair Two of the delegated-security model that the untrusted server U_i, $i = 1$ or $i = 2$, learns nothing useful. By ignoring the ith round, a simulator S_2 produces random queries for both U_1' and U_2', behaving exactly like S_1, and saves its states. Furthermore, it saves the states of (U_1', U_2'). Due to OMTUP assumption, an external environment adversary cannot tell U_1' or U_2' that the simulator S_2 produces bogus outputs since neither E and U_1' nor E and U_2' can communicate directly to develop a joint strategy after interacting with T. Similarly, U_1' and U_2' cannot communicate directly to collaborate to test the random inputs. Now, we have the following possibilities:

- *$(\alpha_1, \alpha_1 P_1, \alpha_1 P_2), (\alpha_2, \alpha_2 P_1, \alpha_2 P_2)$ are honest, secret for both U_1' and U_2',*
- *$(\alpha_1, \alpha_1 P_1, \alpha_1 P_2)$ is honest, unprotected from U_1',*
  and/or $(\alpha_2, \alpha_2 P_1, \alpha_2 P_2)$ is honest, unprotected from U_2'.

If $(\alpha_1, \alpha_1 P_1, \alpha_1 P_2), (\alpha_2, \alpha_2 P_1, \alpha_2 P_2)$ are honest, secret for both U_1' and U_2', then U_1' and U_2' cannot distinguish the real queries from the random ones due to the exactly same reason when interacting with S_1, whence $UVIEW_{real}^i \sim UVIEW_{ideal}^i$. Hence, by a hybrid argument

$$UVIEW_{real} \sim UVIEW_{ideal}.$$

If $(\alpha_1, \alpha_1 P_1, \alpha_1 P_2)$ is honest, unprotected from U_1', then $t_1, t_1 P_1, t_1 P_2$ are unprotected from U_1' but honest, secret for U_2'. If further $(\alpha_2, \alpha_2 P_1, \alpha_2 P_2)$ is honest, unprotected from U_2', then $t_1, t_1 P_1, t_1 P_2$ are unprotected from U_2' but honest, secret for U_1'. Then,

the simulation S_2 is trivial for unprotected values, i.e. S_2 behaves the same way as in the real execution. In this case, the rest of the proof follows exactly as above for honest, secret static inputs, whence

$$UVIEW_{real} \sim UVIEW_{ideal}.$$

Efficiency. Since Rand needs

- *two modular multiplications (MM's) to prepare $c_1 t_1$ and $c_2 t_2$,*
- *four elliptic curve point additions (PA's) to compute $t_1 P_1, t_1 P_2, t_2 P_1, t_2 P_2$,*
- *three MM's to compute $g^{ab}, g^{t_1 s}, g^{t_2 s}$,*
- *two MM's to compute t_1^{-1} and t_2^{-1}, and*
- *four MM's to compute at_1^{-1}, bt_1^{-1}, at_2^{-1}, and bt_2^{-1}.*

Moreover, computation of modular exponents and elliptic curve scalar multiplications take $\mathcal{O}(\log q)$ steps (e.g. by square-and-multiply and double-and-add methods or their variants). Therefore, (T, U_1, U_2) is an $\mathcal{O}(1/\log q)$-efficient implementation of $Rand$.

■

Theorem 5.2: In the one-malicious version of a two-untrusted program model, the algorithms $(\mathcal{C}, \mathcal{U}_1, \mathcal{U}_2)$ are a fully verifiable $\mathcal{O}(\frac{1}{\log q})$-efficient delegated-secure implementations of VerPair*, where the inputs (A, B) may be honest, secret; or honest, protected; or adversarial protected.*

Proof . We prove completeness, security, full verifiability and efficiency of VerPair *as follows:*
Completeness. Assume that the servers U_1 and U_2 run VerPair *honestly. It is not difficult to see that*

$$D_{13} = \beta_3^{at_2^{-1}} = e(aP_1, B - bP_2) \tag{5.228}$$

$$= \beta_1^{at_1^{-1}} = D_{23}, \tag{5.229}$$

and

$$D_{14} = \beta_4^{bt_2^{-1}} = e(A - aP_1, bP_2) \tag{5.230}$$

$$= \beta_2^{bt_1^{-1}} = D_{24}. \tag{5.231}$$

Similarly,

$$D_{15} = e(A - aP_1, B - bP_2) = D_{25}. \tag{5.232}$$

Hence, the verification step of VerPair *is complete. Now,*

$$D_{13} \cdot D_{14} \cdot D_{15} \cdot g^{ab} = \beta_3^{at_2^{-1}} \cdot \beta_4^{bt_2^{-1}} \cdot e(A - aP_1, B - bP_2) \cdot g^{ab} \tag{5.233}$$

$$= e(aP_1, B - bP_2) \cdot e(A - aP_1, bP_2) \cdot e(A - aP_1, B - bP_2) \cdot e(aP_1, bP_2) \tag{5.234}$$

$$= e(aP_1, B - bP_2) \cdot e(aP_1, bP_2) \cdot e(A - aP_1, bP_2) \cdot e(A - aP_1, B - bP_2) \tag{5.235}$$

$$= e(aP_1, B) \cdot e(A - aP_1, B) \tag{5.236}$$

$$= e(A, B). \tag{5.237}$$

Full Verifiability. Assume without loss of generality that U_1 is a malicious server capable of cheating the delegator T with non-negligible probability. Let $h = g^\omega \in \mathbb{G}_3$ be given. Now, we consider the algorithms T^{U_1,U_2} implementing VerPair *for which $aP_1 = \omega P_1$ is chosen. The delegator T verifies at the end of the scheme*

$$D_{13} = \beta_3^{\omega t_2^{-1}} = e(\omega P_1, B - bP_2) = \beta_1^{\omega t_1^{-1}} = D_{23}, \tag{5.238}$$

$$D_{14} = \beta_4^{bt_2^{-1}} = e(A - \omega P_1, bP_2) = \beta_2^{bt_1^{-1}} = D_{24}, \tag{5.239}$$

and

$$D_{15} = e(A - \omega P_1, B - bP_2) = D_{25}. \tag{5.240}$$

Since both D_{15} and D_{25} are used to delegate $e(A-\omega P_1, B-bP_2)$ and U_2 is honest, U_1 can only cheat T during the verification formulas (5.238) and (5.239). Let $x, y \in \mathbb{Z}_q^*$ with $A-\omega P_1 - sP_1 = xP_1,\ B-bP_2-sP_2 = yP_2$ be given. Instead of sending $D_{11} = g^{xt_1}$ (resp. $D_{12} = g^{yt_1}$), U_1 chooses bogus values $\theta_1, \theta_2 \in \mathbb{Z}_q^*$ and send $\Gamma_{11} = g^{\theta_1}$ and $\Gamma_{12} = g^{\theta_2}$ to the delegator. Then, T computes

$$\varphi_1 = \Gamma_{11} \cdot g^{t_1 s} = g^{\theta_1 + t_1 s}, \tag{5.241}$$

$$\varphi_2 = \Gamma_{12} \cdot g^{t_1 s} = g^{\theta_2 + t_1 s}, \tag{5.242}$$

$$\beta_3 = D_{21} \cdot g^{t_2 s} = e(t_2 P_1, B - bP_2), \tag{5.243}$$

$$\beta_4 = D_{22} \cdot g^{t_2 s} = e(A - \omega P_1, t_2 P_2). \tag{5.244}$$

instead of β_1 and β_2. Note that β_3 and β_4 are correct values since U_2 is honest. Then, U_2 computes in the second round

$$\phi_{23} = (g^{\theta_1 + t_1 s})^{\omega t_1^{-1}} = g^{\theta_1 \omega t_1^{-1} + s\omega}, \tag{5.245}$$

$$\phi_{24} = (g^{\theta_2 + t_1 s})^{b t_1^{-1}} = g^{\theta_2 b t_1^{-1} + sb} \tag{5.246}$$

instead of D_{23} and D_{24} following T^{U_1,U_2} honestly. Hence, in order to pass the verification steps (5.238) and (5.239), U_1 must know exactly the values of ϕ_{23} and ϕ_{24}. Note that if $B-bP_2 = y_2 P_2$ and $A-\omega P_1 = x_1 P_1$, then U_1 knows further $\beta_3 = g^{x_1 t_2}$ and $\beta_4 = g^{y_1 t_2}$, and the values ωt_2^{-1} and $b t_2^{-1}$ from the scheme specification. Furthermore, sP_1 (resp. sP_2) is also known by U_1 in the second round; since by subtracting $A-\omega P_1$ (resp. $B - bP_2$) from the first component of D_{12} (resp. from the second component of D_{11}), U_1 can easily obtain sP_1 (resp. sP_2). Then, in order to compute the values

$$\phi_{23} = g^{\theta_1 \omega t_1^{-1} + sa} = g^{\omega(\theta_1 t_1^{-1} + s)} = (g^{\theta_1 t_1^{-1} + s})^{\omega}, \tag{5.247}$$

$$\phi_{24} = g^{\theta_2 b t_1^{-1} + sb} = g^{b(\theta_2 t_1^{-1} + s)} = (g^{\theta_2 t_1^{-1} + s})^{b}. \tag{5.248}$$

U_1 needs to know the exponents ω and b from h_1^{ω} and h_2^{b} with non-negligible probability due the fact that $h_1 = g^{\theta_1 t_1^{-1} + s},\ h_2 = g^{\theta_2 t_1^{-1} + s} \in \mathbb{G}_3$ are known to U_1. Notice that, ω, b

and ωb cannot also be computed from ωt_2^{-1}, bt_2^{-1} and $\omega b - t$ by the proof of secrecy of the delegated part of Rand*, i.e. t_2 is only available to U_2. Therefore, if U_1 can compute ω from $h^{\theta_1 t_1^{-1}+s} = h_1^{\omega}$, thence solves the discrete logarithm problem (DLP) to the base g, with non-negligible probability. Since, U_1 is a polynomially bounded adversary, this gives a contradiction.*

Security. The proof is similar to the proof of Theorem (5.1). We assume now that $\mathcal{A} = (E, U_1', U_2')$ is a probabilistic polynomial-time (PPT) adversary interacting with a PPT-based algorithm T in the delegated-security model of Section (2). Our fist claim is

$$EVIEW_{real} \sim EVIEW_{ideal},$$

e.g. Pair One in the security model that the external adversary environment E learns nothing useful. If inputs (A, B) are either honest, protected or adversarial protected, then a simulator S_1 behaves exactly as in the real execution, i.e. it never requires to access (A, B) since both of them are not secret to the adversary E. We now assume that (A, B) are honest, secret inputs. Then, ignoring the ith round, S_1 first chooses elements $\ell_i \in \mathbb{Z}_q^*$, $1 \leq i \leq 8$ randomly, computes $(\ell_1 P_1, \ell_2 P_2, \ell_3 P_1, \ell_4 P_2)$ for U_1' and $(\ell_4 P_1, \ell_6 P_2, \ell_7 P_1, \ell_8 P_2)$, and makes 2 random queries to U_1'

$$U_1'(\ell_1 P_1, \ell_2 P_2) \leftarrow D_{11}, \tag{5.249}$$

$$U_1'(\ell_3 P_1, \ell_4 P_2) \leftarrow D_{12}, \tag{5.250}$$

and 2 random queries to U_2'

$$U_2'(\ell_5 P_1, \ell_6 P_2) \leftarrow D_{21}, \tag{5.251}$$

$$U_2'(\ell_7 P_1, \ell_8 P_2) \leftarrow D_{22}. \tag{5.252}$$

After receiving the outputs of U_1' and U_2', the simulator S_1 chooses random elements (g_1, γ_1), $(g_2, \gamma_2) \in \mathbb{G}_3 \times \mathbb{Z}_q^*$ and random elements $\ell_9, \ell_{10} \in \mathbb{Z}_q^*$, compute $(\ell_9 P_1, \ell_{10} P_2)$, and queries randomly to U_1'

$$U_1'(g_1, \gamma_1) \leftarrow D_{13}, \tag{5.253}$$

$$U_1'(g_2, \gamma_2) \leftarrow D_{14}, \tag{5.254}$$

$$U_1'(\ell_9 P_1, \ell_{10} P_2) \leftarrow D_{15}, \tag{5.255}$$

similarly, S_1 *chooses random elements* $(g_3, \gamma_3), (g_4, \gamma_4) \in \mathbb{G}_3 \times \mathbb{Z}_q^*$ *and random elements* $\ell_{11}, \ell_{12} \in \mathbb{Z}_q^*$*, compute* $(\ell_{11} P_1, \ell_{12} P_2)$*, and queries randomly to* U_2'

$$U_2'(g_3, \gamma_3) \leftarrow D_{23}, \tag{5.256}$$

$$U_1'(g_4, \gamma_4) \leftarrow D_{24}, \tag{5.257}$$

$$U_1'(\ell_{11} P_1, \ell_{12} P_2) \leftarrow D_{25}. \tag{5.258}$$

Then, S_1 *behaves*

- *if the outputs* D_{1i} *of* U_1' *and* D_{2i} U_2' *are not equal for a randomly selected* i*,* $3 \leq i \leq 5$*, then the values* $Y_p^i ='' error''$*,* $Y_u^i = \emptyset$*, and* $replace^i = 1$ *(corresponding to the output* $(estate^i, '' error'', \emptyset)$ *in the ideal process) are produced by* S_1*,*
- *if no* $''error''$ *is detected , then then the values* $Y_p^i = \emptyset$*,* $Y_u^i = \emptyset$*, and* $replace^i = 0$ *(corresponding to the output* $(estate^i, Y_p^i, Y_u^i)$ *in the ideal process) are produced by* S_1*,*
- *otherwise,* S_1 *selects a random element* r *and outputs* $Y_p^i = r$*,* $Y_u^i = \emptyset$*, and* $replace^i = 1$ *(corresponding to the output* $(estate^i, r, Y_u^i)$ *in the ideal process).*

In either cases, S_1 *saves the appropriate states.*

The distributions of inputs in the real and ideal experiments are computationally indistinguishable. In the ideal experiment, the inputs are chosen uniformly at random. In the real experiment, all inputs of VerPair *are independently randomized by the choice of uniformly distributed random elements. Note that, by each invocation of* VerPair*, new random values are generated by* Rand *which are different from other invocations, and computationally indistinguishable from random elements. Since* VerPair *is a fully verifiable secure-delegated scheme, we only have two cases*

- *if* U_1' *and* U_2' *behave honestly both in the real and the ideal experiments in the round* i*, then we have* $EVIEW_{real}^i$~$EVIEW_{ideal}^i$ *since in the real execution* $T^{U_1', U_2'}$ *perfectly runs* VerPair*, and in the ideal execution* S_1 *does not change the output,*

. *If one of U_1' or U_2' behaves dishonestly in the round i, than this can be detected by both T and S_1 with probability* 1*, see full verifiability.*

In particular, it is impossible that VerPair could be corrupted implying that S_1 never executes the case of selecting a random element r and returning $Y_p^i = r$, $Y_u^i = \emptyset$, and $replace^i = 1$ in the ideal experiment. for further details. This implies that it is impossible for both U_1' and U_2' to deviate from their functionalities. Thus, we have

$$EVIEW^i_{real} \sim EVIEW^i_{ideal}$$

even in the case that one of U_i', $i = 1, 2$, misbehaves. By the hybrid argument, we conclude that

$$EVIEW_{real} \sim EVIEW_{ideal}.$$

It is clear that this argument only works if only one server misbehaves (under OMTUP model), i.e. if both U_1 and U_2 are malicious simultaneously, then the misbehavior in this case is not independent of the inputs (A, B) whereas the misbehavior of only one of U_i, i=1,2, is independent of the inputs (A, B).
Secondly, we claim that

$$UVIEW_{real} \sim UVIEW_{ideal},$$

i.e. Pair Two of the delegated-security model that the untrusted server U_i, $i = 1$ or $i = 2$, learns nothing useful. For a round i, a simulator S_2 behaves exactly like S_1 to produce random queries by ignoring the ith round for both U_1' and U_2', and saves its states. Furthermore, it saves the states of (U_1', U_2'). Due to OMTUP assumption, an external environment adversary can tell neither to U_1' nor to U_2' that the simulator S_2 produces bogus outputs since the output in the real experiment is not corrupted, and neither E and U_1' nor E and U_2' can communicate directly in order to develop a joint strategy after interacting with T. Hence, honest, secret; honest, protected; or adversarial protected inputs are all private for both U_1' and U_2', although E could easily distinguish between these real and ideal experiments. The reason, exactly as in the case of interacting with S_1, is that in the ith round of the real experiment, the values given to either U_1' or U_2' are completely re-randomized by Rand*, and S_2 generates random, independent queries for both U_1' and U_2' in the ideal experiment. Thus, we have*

$$UVIEW^i_{real} \sim UVIEW^i_{ideal}$$

for each round i. It follows then by a hybrid argument

$$UVIEW_{real} \sim UVIEW_{ideal}.$$

■

5.4.1. Comparison

We give the comparison of single pairing outsourcing mechanisms that utilizes multiple servers in Table 5.2.

Table 5.2: Comparison of the Computational Costs and Communication Complexities.

Delegation Scheme	Secrecy (output)	Verifiability (real)	Client's workload	Servers' workload	#Rounds
Chevallier-Mames et al [2005] SVSV	yes	1	$2\text{PA}_1, 2PA_2, 6MM_T, 3SM_1, 3SM_2, 10ME_T$	4P	1
Chevallier-Mames et al [2005] SVPV	NA	1	$2\text{PA}_1, 1PA_2, 4MM_T, 3SM_1, 2SM_2, 8ME_T$	4P	1
Chevallier-Mames et al [2005] PVPV	NA	1	$1\text{PA}_1, 1PA_2, 3MM_T, 2SM_1, 2SM_2, 7ME_T$	4P	1
Chevallier-Mames et al [2005] PCPV	NA	1	$1\text{PA}_2, 1MM_T, 1SM_2, 3ME_T$	2P	1
Chevallier-Mames et al [2005] SCPV	NA	1	$1\text{PA}_2, 1MM_T, 2SM_1, 1SM_2, 4ME_T$	2P	1
Kang et al [2005] SVSV	yes	1	$1\text{PA}_1, 1PA_2, 3MM_T, 2SM_1, 3SM_2, 7ME_T$	4P	1
Kang et al [2005] SVPV	NA	1	$1\text{PA}_1, 1PA_2, 3MM_T, 2SM_1, 2SM_2, 8ME_T$	4P	1
Kang et al [2005] SCSV	NA	1	$1\text{PA}_1, 1MM_T, 2SM_1, 2SM_2, 4ME_T$	2P	1
Kang et al [2005] SVPC	NA	1	$1\text{PA}_1, 1MM_T, 2SM_1, 3ME_T$	2P	1
Canard et al [2014] SVSV	yes	1	$1\text{PA}_1, 1PA_2, 1MM_T, 2SM_1, 2SM_2, 2ME_T, 1MT_T$	2P	1
Canard et al [2014] PVPV	yes	1	$1\text{PA}_1, 1PA_2, 1MM_T, 1SM_1, 1SM_2, 1ME_T, 1MT_T$	2P	1
Luo et al [2018]	no	0	4PA, 4MM, 2SM, 3MI, 1ME	6P	1
Dong et al [2017]	no	1	6PA, 19MM	10P	1
Kalkar et al [2018]	yes	0	4PA, 6MM	8P	1
Luo et al [2016]	yes	1/2	8PA, 6MM	4P	1
Ren et al [2016]	yes	5/6	8PA, 14MM	6P, 4ME	2
Ren et al [2017] ($s = 4$)	yes	0.967	8PA, 19MM	10P	1
Canard et al [2014]	yes	1	2PA, 1MM, 1TM, 4SM, 2ME	4P	1
Uzunkol et al [2017]	yes	1	4PA, 7MM	6P, 4ME	2

6. BATCH OUTSOURCING PAIRING COMPUTATIONS

There are schemes that require multiple pairing computations, or systems that runs multiple schemes involving pairing computations. In these cases, considering batch outsource may be efficient. Essentially, one should use batch outsource when outsourcing n computations at once is cheaper than outsourcing computations n times. In blockchain, every transaction is signed with user's secret key and nodes validate these transactions by verifying signature on the transaction. Since there are multiple transactions in a certain time period, batch signature verification is a desired property for signature schemes. When there is not a batch verification algorithm in hand, outsourcing these signature verifications can be useful.

6.1. Protocols

Pairings $e(A, B)$ can be classified into 16 types depending each of points A, and B is public/secret and constant/variable. Half of them are duplicates, and there is no need for batch outsourcing when both points are public constant. This leads to a taxonomy of pairings into 7 types, as given by Tsang et al. [2007]. These types are denotes as SVSV, SVSC, SVPV, SVPC, PVPV, PVPC, PVSC; where P/S denotes public/secret and V/C denotes variable/constant. There are 5 papersTsang et al. [2007]; Luo et al. [2016]; Mefenza and Vergnaud [2018]; Shao and Wei [2018]; Kalkar et al. [2020] on batch pairing outsource.

- Tsang et al. [2007] give a solution for SVSC type pairings which can be modified to work for SVPC, PVSC, and PVPC types.
- Mefenza and Vergnaud [2018] propose more efficient solutions for PVPC and PVSC types, and also give the first solution for PVPV type.
- Shao and Wei [2018] proposes a more efficient solution for SVPC type.
- This thesis proposes solutions for all types. In detail, they propose solutions for SVSV and SVPV types for the first time and also give more efficient algorithms for other types.

Comparison of the proposed mechanisms are given in Table 6.1.

6.1.1. Single Server

In this section, we present batch pairing outsource mechanisms utilizing only one server. Public Variable, Public Variable. PVPV type pairings have public variable input on both sides. They are commonly used in cryptographic constructions, and can be seen in verification of signature schemes and ciphertext validity checks of some encryption schemes. Some schemes that need this type of pairing calculations are Boldyreva [2002]; Boneh and Boyen [2008]; Boneh et al. [2003]; Baek and Zheng [2004]; Chow et al. [2006]; Chow [2005]. First solution is proposed by Mefenza and Vergnaud [2018], and we proposed a more efficient solution Kalkar et al. [2020] that does not require pairing calculation.

In all algorithms, inputs are public variable $A_1, \cdots, A_n$, public variable $B_1, \cdots, B_n$ and outputs are $e(A_1, B_1), \cdots, e(A_n, B_n)$.

In order to calculate $e(A_i, B_i)$, Mefenza and Vergnaud [2018] proposes the following

i) C chooses random $\lambda_i \in 0, 1, \ldots, p-1$ and a random point $R \in \mathbb{G}_2$.

ii) C precomputes $R_i = \lambda_i R$ for $1 \leq i \leq n$.

iii) C selects n random elements $b_1, \ldots, b_n \in \{0, 1, \ldots, 2^t - 1\}$.

iv) C computes $B'_i = b_i B_i + R_i$ for $i \in \{1, \ldots, n\}$.

v) C sends $A_1, \ldots, A_n, B_1, \ldots, B_n, B'_1, \ldots, B'_n$ to S.

vi) S computes and sends $\alpha_i = e(A_i, B_i)$, $\alpha'_i = e(A_i, B'_i)$ to C.

vii) C verifies $\alpha_i, \alpha'_i \in \mathbb{G}_T$ for each $1 \leq i \leq n$.

viii) C computes $A = \sum_{i=1}^{n} \lambda_i A_i$.

ix) C computes $\alpha = e(A, R)$.

x) C computes $\alpha' = \prod_{i=1}^{n} \alpha'_i$ and $\alpha'' = \prod_{i=1}^{n} \alpha_i^{b_i}$.

xi) C verifies $\alpha' = \alpha'' \times \alpha$.

xii) C outputs $e(A_i, B_i)$ as α_i if all verifications are succesfull. Otherwise, outputs $\perp$ and halts.

Secret Variable, Secret Variable. In this type batch pairing computation, both input values are secret and variable. To the best of our knowledge, there are no cryptographic

schemes that require computation of SVSV type pairings. However, this is the most generic way of batch pairing delegation and any construction for this type should also work for other types since both input and output values should be secured. To the best of our knowledge, no delegation protocol for this type is proposed until Kalkar et al. [2020].

Secret Variable Secret Constant. Pairings of this type have secret inputs, one is variable and one is constant. This type of pairings appear in few cryptographic schemes, some are Boneh and Boyen [2004]; Chen and Kudla [2003]. Tsang et al. [2007] presents the first outsourcing protocol for this type and we give a more efficient algorithm.

In order to calculate $e(A_i, B_i)$, Tsang et al. [2007] proposes the following protocol:

i) C chooses random $r_B, r_P \in \mathbb{Z}_p^*$ and precomputes $\tilde{B} = r_B B, \tilde{P} = r_P P$, and $\chi = e(\tilde{P}, \tilde{B}) = e(P, B)^{r_P r_B}$.

ii) C chooses random $r_i, a_i \in \mathbb{Z}_q^*$ and computes

$$\tilde{A}_i = r_i A_i \quad \text{for } 1 \leq i \leq n, \tag{6.1}$$

$$\tilde{A} = \tilde{P} + \sum_{i=1}^{n} a_i \tilde{A}_i. \tag{6.2}$$

iii) C sends $\tilde{A}_1, \ldots, \tilde{A}_n, \tilde{A}$ to S and gets $\alpha = e(\tilde{A}, \tilde{B})$ and $\alpha_i = e(\tilde{A}_i, \tilde{B})$.

iv) C verifies that $\alpha, \alpha_1, \ldots, \alpha_n \in \mathbb{G}_T$.

v) C computes $\alpha' = \chi \prod_{i=1}^{n} \alpha_i^{a_i}$.

vi) C checks $\alpha = \alpha'$ and outputs $e(A_i, B)$ as $\alpha_i^{\frac{1}{r_B r_i}}$. Otherwise, outputs $\perp$ and halts.

Secret Variable, Public Variable.

SVPV type pairings have secret variables on one side of the input, and public variables on the other side. This type of pairings appear in searchable encryption Boneh and Waters [2007]; Shi et al. [2007], trace-and-revoke broadcast system Boneh and Waters [2006], identity based encryption Gentry [2006]. To the best of our knowledge, no delegation protocol for this type is proposed until Kalkar et al. [2020].

Secret Variable, Public Constant.

This type has secret variable inputs on the one side and a public constant input on the other. Just like SVPC, this type of pairings are rare in cryptographic constructions. One example is ciphertext-policy attribute based encryption, Bethencourt et al. [2007]. Tsang et al. [2007] proposed to use modified version of their SVSC algorithm. Later, Shao and Wei [2018] proposed another algorithm which is more efficient. However, their algorithm does not control if the pairing values returned by the delegatee are elements of the target group $\mathbb{G}_T$, so their scheme is susceptible to subgroup attacks Barreto et al. [2015]. This threat can be avoided using necessary group membership tests. In addition to that, their scheme can be extremely more efficient if the scalar a_i's are chosen from exponents of smaller bit length. We give another algorithm which uses Kalkar et al. [2020] PVPV and is more efficient.

Public Variable, Secret Constant. This type of pairings have one public variable input and one secret constant input. Some examples are identity based encryption Boneh and Franklin [2001]; Gentry [2006]; Sakai et al. [2000], certificateless encryption Chow et al. [2006]; Dent [2008], and attribute-based encryption Bethencourt et al. [2007]; Goyal et al. [2006]. This type of batch pairing calculation commonly appears in decryption algorithms, and the secret constant variable corresponds to the secret key. Most of these encryption mechanisms includes a Key Generation Center (KGC), and secret keys are distributed by KGC. Hence, it is reasonable to assume that $\gamma = e(P, -B)$ is given by KGC to the delegator, where P is a generator of $\mathbb{G}_1$ and B is the secret constant part of the inputs.

Tsang et al. [2007] presents the first outsourcing protocol for this type, later Mefenza and Vergnaud [2018] proposes a faster version. Mefenza and Vergnaud [2018]'s algorithm for PVSC type pairings, namely Algorihm 5, requires computing n pairings $\chi_i = e(X_i, Q_0)^{-1}$. Using our techniques, one can mitigate the need for pairing computation. Following the same notation by Mefenza and Vergnaud [2018], one can modify their Algorithm 5 to remove pairing computations performed by outsourcer. During the precomputation phase, they choose random $X_i \in \mathbb{G}_1$ and compute $\chi_i = e(X_i, Q_0)^{-1}$, where $Q_0 = rQ$ with random r. Instead, one can get $X_i = x_i A$ with random x_i's and compute χ_i as $\chi_i = \gamma^{x_i r}$, where $\gamma = e(P, -B)$ (in their notation). In addition to that, they use their PVPC algorithm in PVSC algorithm. Kalkar et al. [2020] uses Algorithm PVPV, instead. These changes make the algorithm

pairing free and improve efficiency.

Public Variable, Public Constant. PVPC type pairings have public variable input on one side and public constant input on the other side. Some cryptographic schemes involving this type of pairing calculations are Boldyreva [2002]; Boneh and Franklin [2001]; Boneh and Boyen [2008]; Boyen and Waters [2007]; Chow [2005]; Yao et al. [2004]. Tsang et al. [2007] proposed that their SVSC protocol can be used with slight modifications. Then, Mefenza and Vergnaud [2018] improved the proposed protocol efficiency wise, but it still requires pairing computation. Kalkar et al. [2020] proposes the first solution which do not require calculating pairings.

In order to calculate $e(A_i, B_i)$, Mefenza and Vergnaud [2018] proposes the following mechanism:

i) C selects a random point P_0 and precomputes $\chi = e(P_0, B)$.

ii) C sends $A_1, \cdots, A_n$ and B to S.

iii) S computes $\alpha_i = e(A_i, B)$ for $i \in \{1, \cdots, n\}$.

iv) C selects n

6.1.2. Multiple Servers

Luo et al. [2016] proposed a scheme for delegation of generic batch pairings. They use a multiplicative notation for the groups $\mathbb{G}_1$ and $\mathbb{G}_2$. To be consistent with the rest of the paper, we summarize their scheme for a single delegation $e(A, B)$, where $A \in \mathbb{G}_1$ and $B \in \mathbb{G}_2$ in the usual additive notation. The Rand algorithm outputs the following values

$$uP, 2uP, vQ, 2vQ, -2vQ, xP, 2xP, yQ, 2yQ, -2yQ,$$

$$e(P, Q)^{2uv}, e(P, Q)^{2xy},$$

where P is the generator of $\mathbb{G}_1$ (of prime order q) and Q is the generator of $\mathbb{G}_2$ (of prime order q) and $u, v, x, y \in_R \mathbb{Z}_q^*$.
Let T denote the delegator and U_1 and U_2 be the two untrusted servers. The scheme in by Luo et al. [2016] is as follows:

i) T queries U_1 in random order as follows:

$$U_1(A + uP, B + vQ) \leftarrow \alpha_1 = e(A + uP, B + vQ), \tag{6.3}$$

$$U_1(A + 2xP, -B - 2yQ) \leftarrow \alpha_2 = e(A + 2xP, -B - 2yQ). \tag{6.4}$$

ii) Then similarly, T queries U_2 in random order as follows:

$$U_2(A + xP, B + yQ) \leftarrow \beta_1 = e(A + xP, B + yQ), \tag{6.5}$$

$$U_2(A + 2uP, -B - 2vQ) \leftarrow \beta_2 = e(A + 2uP, -B - 2vQ). \tag{6.6}$$

iii) T verifies

$$\alpha_1^2 \cdot \beta_2 \cdot e(P, Q)^{2uv} \stackrel{?}{=} \beta_1^2 \cdot \alpha_2 \cdot e(P, Q)^{2xy}. \tag{6.7}$$

iv) If the verification step fails T outputs $\perp$.

v) Else, T outputs $e(A, B) = \alpha_1^2 \cdot \beta_2 \cdot e(P, Q)^{2uv}$.

Now we present an attack on the verifiability of Luo et al. [2016] by Uzunkol et al. [2017]. Assume U_1 is a malicious server. It could successfully guess the positions of α_1 and α_2 with probability $1/2$. Instead of sending α_1 and α_2, U_1 could send the bogus values $\theta_1 = \alpha_1 \cdot C$ and $\theta_2 = \alpha_2 \cdot C^2$ and would pass the verification step with probability at least $1/2$, where $C \in \mathbb{G}_3$ is any arbitrary bogus value. Then, the scheme outputs $C^2 e(A, B)$ instead of $e(A, B)$ with probability at least $1/2$. Obviously, a malicious server U_2 could mount a similar simple attack.
This fairly simple attack shows that Luo et al. [2016]' claim of having a fully verifiable scheme is unfortunately false.

6.2. Proposed Schemes

Kalkar et al. [2020] brings together the techniques that help to make a batch pairing outsourcing protocol secure and more efficient. Then by applying these techniques, we propose solutions for SVSV and SVPV type batch pairing outsource protocols which had no solutions before. In addition to this, we point out that Shao and Wei [2018]'s SVPC proposal is susceptible to small subgroup attacks and propose a secure and more efficient algorithm. For the PVPV type, we introduce a scheme which

is 1.5 times more efficient than Mefenza and Vergnaud [2018]'s. We also propose solutions for the remaining two types, namely PVPC and PVSC, which require no pairing computations on the end-user side with slightly less computational cost than the ones proposed by Mefenza and Vergnaud [2018]. All of the proposed protocols frees the delegator from including pairing arithmetic libraries.

Public Variable, Public Variable. Mefenza and Vergnaud [2018]'s second algorithm for PVPV type pairings, namely Algorihm 4, requires computing a pairing $e(P, R)$. This can be avoided. In the following, we give a pairing free and more efficient batch outsourcing algorithm for PVPV type. This algorithm is the base algorithm for Kalkar et al. [2020] PVPC, PVSC, and SVSC.

i) C chooses random $a_i \in 0, \ldots, 2^{t-1}$.

ii) C computes $A_i'' = a_i A_i'$ for $1 \leq i \leq n$.

iii) C sends $A_1, \ldots, A_n, A_1'', \ldots, A_n'', B_1, \ldots, B_n$ to S.

iv) S computes and sends $\alpha_i = e(A_i, B_i)$, $\alpha_i' = e(A_i'', B_i)$ to C.

v) C verifies $\alpha_i, \alpha_i' \in \mathbb{G}_T$ for each $1 \leq i \leq n$

vi) C verifies $\alpha_i' = \alpha_i^{a_i}$.

vii) C outputs $e(A_i, B_i)$ as α_i if all verifications are successful.

One can see that Kalkar et al. [2020] PVPV works correctly if the server S sends the correct values of $\alpha_1, \ldots, \alpha_n$, $\alpha_1', \ldots, \alpha_n'$.

Theorem 6.1: Kalkar et al. [2020] PVPV is $\beta-$verifiable with $1 - 2^{-t}$.

Proof . Pairing values are outputed as α_i's which are checked at step 6 with the equation $\alpha_i' = \alpha_i^{a_i}$. Probability of accepting a wrong pairing value at this step is 2^{-t} by the small exponents test. ■

Computational cost. Step 2 costs $10.7tn\, M_p$ (n multiplications on $\mathbb{G}_1$ with scalars of bit-length t), step 5 costs $108n\, M_p$($2n$ group membership tests), and step 5 costs $36tn\, M_p$ (n exponentiations on $\mathbb{G}_T$ with exponents of bit-length t). So, total cost is $46.7tn + 108n\, M_p$.

Theorem 6.2: For 128-bit security($t = 128$), Kalkar et al. [2020] PVPV is α-efficient with $\alpha = 3.76$.

Secret Variable, Secret Variable. To the best of our knowledge, no delegation protocol for this type is proposed until Kalkar et al. [2020]. This protocol does not require any pairing calculations by the end user C under the assumption of $e(P, Q)$ is known by C, where P and Q are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. In some algorithms, $\gamma = e(P, Q)$ is given among public parameters. Even if it is not given as a public parameter, it can be added. Inputs are secret $A_1, \cdots, A_n$, secret $B_1, \cdots, B_n$ and outputs are $e(A_1, B_1), \cdots, e(A_n, B_n)$.

i) C chooses random $x, y, y_1, y_2, \ldots, y_n \in \mathbb{Z}_q^*$ and precomputes $X = xP, Y = yP$, $Y_1 = y_1Y, \ldots, Y_n = y_nY, \beta = \gamma^{xy}$.

ii) C chooses random $a_i, b_i \in \mathbb{Z}_q^*$ for $1 \leq i \leq n$.

iii) C calculates $A'_i = a_iA_i, B'_i = b_iB_i$ for $1 \leq i \leq n$.

iv) C chooses $c_i \in \{0, 1, \ldots, 2^t - 1\}$ for $1 \leq i \leq n$.

v) C computes $B''_i = c_iB'_i + Y_i$ for $1 \leq i \leq n$.

vi) C computes $A = \sum_{i=1}^{n} y_iA'_i - X$.

vii) C sends $A, Y, A'_1, \ldots, A'_n, B'_1, \ldots, B'_n, B''_1, \ldots, B''_n$ to S.

viii) S computes and sends $\alpha = e(A, Y), \alpha_i = e(A'_i, B'_i), \alpha'_i = e(A'_i, B''_i)$ to C.

ix) C verifies $\alpha, \alpha_i, \alpha'_i \in \mathbb{G}_T$. C computes $\alpha' = \prod_{i=1}^{n} a'_i, \quad \alpha'' = \prod_{i=1}^{n} \alpha_i^{c_i}$.

x) C verifies $\alpha' = \alpha\alpha''\beta$.

xi) C outputs $e(A_i, B_i)$ as α_i^{1/a_ib_i}, if all verifications are successful.

It is easy to see that above protocol works correctly if delegatee sends correct values of $\alpha, \alpha_1, \ldots, \alpha_n, \alpha'_1, \ldots, \alpha'_n$. In addition to that, delegatee can only cheat C with a probability $1/2^t$.

Theorem 6.3: Kalkar et al. [2020] SVSV is secure and $\beta-$verifiable with $\beta = 1 - 2^{-t}$.

Proof . Final pairing results are calculated as $e(A_i, B_i) = \alpha_i^{1/a_ib_i}$. In this calculation a_i, b_i values are on the client side. Only values that come from the server are α_i's. Correctness of these values are checked at step 12 with equation $\alpha' = \alpha\alpha''\beta$. Probability of excepting a wrong α_i at this step is $1/2^t$, by the small exponents test. ■

Computational cost. Step 1 costs $4933n + 9866 + 16596\, M_p$ ($(n + 2)$ multiplications on $\mathbb{G}_1$, 1 exponentiation on $\mathbb{G}_T$) which is the precomputation cost. Step 3 costs

$4933n + 11848n\,M_p$ (n multiplications on $\mathbb{G}_1$, n multiplications on $\mathbb{G}_2$), step 5 costs $10.7tn + 29n\,M_p$ (n multiplications on $\mathbb{G}_2$ by scalars of length t, n additions on $\mathbb{G}_2$), step 6 costs $4933n + 11n\,M_p$ (n multiplications and n additions on $\mathbb{G}_1$), step 9 costs $(2n+1)54\,M_p$ ($2n+1$ group membership tests), step 10 costs $(2n-2)54 + 36tn\,M_p$ ($2n-2$ multiplications on $\mathbb{G}_T$ and n exponentiations on $\mathbb{G}_T$ by exponents of length t), step 11 costs 108 (2 multiplications on $\mathbb{G}_T$), and finally step 12 costs $16596n\,M_p$(n exponetiations on $\mathbb{G}_T$). So, total cost is $46.7tn + 38566n + 54\,M_p$.

Theorem 6.4: For 128-bit security($t = 128$), Kalkar et al. [2020] SVSV is α-efficient with $\alpha = 0.51$.

Secret Variable, Secret Constant. We proposed a more efficient algorithm for this type. Kalkar et al. [2020] PVSC can be used as its is for SVSC type. So, we get the following theorems readily.

Theorem 6.5: The batch pairing delegation for SVSC type is secure and $\beta-$verifiable with $\beta = 1 - 2^{-t}$.

Theorem 6.6: For 128-bit security($t = 128$), the batch pairing delegation algorithm for SVSC type is α-efficient with $\alpha = 2.08$.

Secret Variable, Secret Variable. To the best of our knowledge, no delegation protocol for this type is proposed until Kalkar et al. [2020].

For this type of batch pairings, one can use algorithm suggested for SVSV type by letting $b_i = 1$.

Theorem 6.7: The batch pairing delegation algorithm for SVPV type is secure and $\beta-$verifiable with $\beta = 1 - 2^{-t}$.

Proof . The proof is similar to that of Theorem 6.3. ■

Computational cost. Cost of this algorithm is $46.7tn + 33633n + 54\,M_p$ which n multiplications on $\mathbb{G}_1$ less than the cost of Kalkar et al. [2020] SVSV.

Theorem 6.8: For 128-bit security($t = 128$), the batch pairing delegation algorithm for SVSV is α-efficient with $\alpha = 0.58$.

Secret Variable, Public Constant.

Assume that $\gamma = e(P, B)$ is known by the outsourcer, where B is the public constant part of the input. When an outsourcer C wants to calculate (A_i, B) for $i \leq 1 \leq n$, for some n, C follows:

Kalkar et al. [2020] SVPC

i) C chooses random $x_1, x_2, \ldots, x_n \in \mathbb{Z}_q^*$ and precomputes $X_1 = x_1P, \ldots, X_n = x_nP$ and $\omega_1 = \gamma^{x_1}, \omega_2 = \gamma^{x_2}, \ldots, \omega_n = \gamma^{x_n}$.

ii) C calculates $A_i' = A_i - X_i$.

iii) C uses Algorithm Algorithm PVPV to get $\alpha_1 = e(A_1', B), \ldots, \alpha_n = e(A_n', B)$.

iv) C outputs $\alpha_i.\omega_i$ as $e(A_i, B)$.

It is easy to see that above protocol works correctly if delegatee sends correct values of $\alpha_1, \ldots, \alpha_n$. In addition to that, delegatee can only cheat C with a probability $1/2^t$.

Theorem 6.9: Kalkar et al. [2020] SVPC is secure and $\beta-$verifiable with $\beta = 1 - 2^{-t}$.

Proof . The proof is similar to that of Theorem 6.1. ■

Computational cost. Step 1 costs $4933n + 16596n\, M_p$ (n multiplications on $\mathbb{G}_1$ and n exponentiations on $\mathbb{G}_T$), so cost of precomputations is $21529n\, M_p$. Step 2 costs $11n\, M_p$ (n additions on $\mathbb{G}_1$), step 3 costs $46.7tn + 108n\, M_p$ (Algorithm PVPV), and finally step 4 costs $54n\, M_p$. So, total cost is $46.7tn + 173n\, M_p$.

Theorem 6.10: For 128-bit security($t = 128$), the batch pairing delegation algorithm for PVPC type is α-efficient with $\alpha = 3.72$.

Public Variable, Secret Constant.

Inputs are public $A_1, \cdots, A_n$, secret constant B and outputs are $e(A_1, B), \cdots, e(A_n, Q)$

i) C chooses random $b, x_1, x_2, \ldots, x_n \in \mathbb{Z}_q^*$ and precomputes $B' = bB$, $X_i = x_iP, \chi_i = \gamma^{x_i b}$ for $1 \leq i \leq n$.

ii) C computes $A_i' = b^{-1}A_i - X_i$ for $1 \leq i \leq n$.

iii) C uses Algorithm PVPV to compute $\alpha_i = e(A_i', B')$ for $1 \leq i \leq n$.

iv) C computes $e(A_i, Q)$ as $\alpha_i\chi_i$.

Theorem 6.11: Kalkar et al. [2020] PVSC is secure and $\beta-$verifiable with $\beta = 1-2^{-t}$.

Proof . The proof is similar to that of Theorem 6.1. ■

Computational cost. Step 1 costs $11848+4933n+16596n\ M_p$(1 multiplication on $\mathbb{G}_2$, n multiplications on $\mathbb{G}_1$ and n exponentiations on $\mathbb{G}_T$). So, cost of precomputation step is $21529n + 11848\ M_p$. Step 2 costs $4933n + 11n\ M_p$(n multiplications and n additions on $\mathbb{G}_1$), step 3 costs $46.7tn + 9n\ M_p$ as calculated in Algorithm PVPV, and finally step 4 costs $54n\ M_p$(n multiplications on $\mathbb{G}_T$). Total cost of Algorithm PVSC is $46.7tn + 5007n\ M_p$.

Theorem 6.12: For 128-bit security($t = 128$), the batch pairing delegation algorithm for PVSC type is α-efficient with $\alpha = 2.08$.

Public Variable, Public Constant.

Kalkar et al. [2020] PVPV can be used for PVPC type pairings by letting $B_1, \ldots, B_n = B$.

Theorem 6.13: The batch pairing delegation algorithm for PVPC type is $\beta-$verifiable with $\beta = 1 - 2^{-t}$.

Proof . The proof is similar to that of Theorem 6.1 ■

Theorem 6.14: For 128-bit security($t = 128$), the batch pairing delegation algorithm for PVPC type is α-efficient with $\alpha = 3.76$.

6.3. Comparison

We have used MIRACL library for efficiency comparison of the mentioned protocols. We ran our tests 10000 times and calculated average timing. We used optimal Ate-pairing on the curves BN256, KSS384, and BLS512 since cheap group membership tests are possible. Please note that even though these curves are not subgroup secure, cost of operations are the same with their subgroup secure versions, except for pairing. Pairing costs slightly more on the subgroup secure versions of these curves Barreto et al. [2015]. Since cost of group operations are the same, our efficiency results hold for the subgroup secure versions too.

Table 6.1: Efficiency of batch pairing algorithms.

		BN256	KSS384	BLS512
PVPV	Mefenza and Vergnaud [2018] (§4.2)	$< 1{,}66^{\spadesuit}$	$< 2{,}08^{\spadesuit}$	$< 0{,}96^{\spadesuit}$
	Kalkar et al. [2020]	2.31	3.09	1.32
PVPC	Tsang et al. [2007]	2.04	2.85	1.25
	Mefenza and Vergnaud [2018] (§4.1)	2.30	3.08	1.32
	Kalkar et al. [2020]	2.31	3.09	1.32
PVSC	Tsang et al. [2007]	1.11	1.50	0.64
	Mefenza and Vergnaud [2018] (§4.3)	1.81	2.70	1.23
	Kalkar et al. [2020]	1.97	2.80	1.26
SVSC	Tsang et al. [2007]	1.03	1.43	0.63
	Kalkar et al. [2020]	1.97	2.80	1.26
SVPC	Tsang et al. [2007]	1.03	1.43	0.63
	Shao and Wei [2018]	2.04	2.84	1.25
	Kalkar et al. [2020]	2.28	3.06	1.31
SVSV	Kalkar et al. [2020]	0.80	1.03	0.47
SVPV	Kalkar et al. [2020]	0.85	1.06	0.48

$^{\spadesuit}$ Mefenza and Vergnaud's algorithm needs 1 constant pairing computation. We did not take it into account while doing calculations since its efficiency changes depending on the number of pairings outsourced.

While making calculations, we used 100-bit scalars for the curve BN256 since its security is shown to be 2^{100} Barbulescu and Duquesne [2019], and 128-bit scalars for the curves KSS384 and BLS512 whenever multiplication/exponentiation by a small scalar is mentioned.

Note that, for PVPC and PVSC types Mefenza *et. al.* Mefenza and Vergnaud [2018] use endomorphisms and achieve 128-bit security with $t = 126$. Same thing can be applied to our algorithms, too. For simplicity, we calculate efficiency of the algorithms without taking use of endomorphisms into account.

Efficiency of the protocols are given in Table 6.1, where the most efficient protocol for each type is shown with blue. Efficiency results are calculated by cost of n pairing computations divided by cost of outsourcing n pairings. So, a greater value means a more efficient scheme.

7. CONCLUSION

This thesis focuses on pairing based cryptography and its applications. After investigating pairings, pairing friendly curves, and pairing based cryptographic primitives, we proposed two applications involving pairings; an electronic checkbook scheme and a blockchain-based electronic exam. Moreover, we investigated outsourcing of pairing computation, gave a literature survey and proposed the most efficient algorithms for batch pairing outsourcing.

Electronic checkbook schemes were proposed throughout the years, however there were no formal security definitions regarding e-checkbooks. This thesis fills this gap and proposes game-based security definitions for e-checkbook unforgeability, e-check unforgeability and non-manipulability, and e-check anonymity properties. In addition to that, it proposes a transferable mechanism with privacy in mind. The proposed scheme prevents someone to forge a valid checkbook or check, and manipulate an e-check while being still efficient than most of the previously proposed schemes. Regarding electronic checkbook schemes, next steps would be creating a blockchain based checkbook scheme.

The second contribution of this thesis is proposing a blockchain based electronic exam that leverages verifiable credentials. The proposed scheme achieves security and full privacy of the candidates/examiners and can easily be audited in case of any dispute since the data always remains, unchanged due to the nature of blockchain. We believe that, registration phase can be realized without introducing other schemes, blind signatures in this case, using solely self-sovereign identity concepts while satisfying anonymity of the candidates and also guaranteeing unique enrollment of a candidate. This remains as a future work.

Even though pairing based cryptography promises many applications, there are some cases that a limited device can not leverage the benefits of pairing based cryptography. This device may have either have low storage or low power capacity. If it has low storage, then deploying pairing libraries may be infeasible. If it has low power, the power may not be enough for pairing computation. The last contribution of this thesis is giving a literature survey on secure delegation of pairing computation, stating attacks on some of the given protocols, and comparing their efficiency. In the

case of batch pairing delegation, this thesis also proposes mechanisms for all of the batch pairing types depending on the inputs to the pairing function being secret/public, constant, verifiable. For some types, there were no propositions up until this thesis, and for the others, this thesis proposes the most efficient mechanisms and removes the need of installing pairing libraries. Regarding pairing outsource, future work would be investigating how these techniques can be applied to zero knowledge proofs or researching if there is a way to not outsource the pairing computation itself but the computation of Miller loop and final exponentiation.

REFERENCES

Ahmed F. R. A., Ahmed T. E., Saeed R. A., Alhumyani H., Abdel-Khalek S., Abu-Zinadah H. (2021). "Analysis and challenges of robust e-exams performance under covid-19". Results in Physics, 23:103987.

Androulaki E., Barger A., Bortnikov V., Cachin C., Christidis K., De Caro A., Enyeart D., Ferris C., Laventman G., Manevich Y. (2018). "Hyperledger fabric: a distributed operating system for permissioned blockchains". In Proceedings of the 13th EuroSys conference, pages 1–15.

Arabacı O., Kiraz M. S., Sertkaya I., Uzunkol, O. (2015). "More efficient secure outsourcing methods for bilinear maps". Cryptology ePrint Archive, Report 2015/960.

Baek J., Zheng, Y. (2004). "Identity-based threshold decryption". Public Key Cryptography – PKC 2004, pages 262–276, Springer Berlin Heidelberg.

Barbulescu R., Duquesne, S. (2019). "Updating key size estimations for pairings". Journal of Cryptology, 32(4):1298–1336.

Barreto P. S. L. M., Costello C., Misoczki R., Naehrig M., Pereira G. C. C. F., Zanon G. (2015). "Subgroup security in pairing-based cryptography". Progress in Cryptology – LATINCRYPT 2015, pages 245–265, Cham. Springer International Publishing.

Barreto P. S. L. M., Galbraith S. D., hÉigeartaigh C. Ó., Scott M. (2007). "Efficient pairing computation on supersingular abelian varieties". Designs, Codes and Cryptography, 42(3):239–271.

Barreto P. S. L. M., Libert B., McCullagh N., Quisquater J.-J. (2005). "Efficient and provably-secure identity-based signatures and signcryption from bilinear maps". Advances in Cryptology - ASIACRYPT 2005, pages 515–532. Springer Berlin Heidelberg.

Bella G., Giustolisi R., Lenzini G., Ryan P. Y. (2017). "Trustworthy exams without trusted parties". Computers & Security, 67:291–307.

Bellare M., Garay J. A., Rabin T. (1998). "Fast batch verification for modular exponentiation and digital signatures". Advances in Cryptology — EUROCRYPT'98, pages 236–250. Springer Berlin Heidelberg.

Bethencourt J., Sahai A., Waters B. (2007). "Ciphertext-policy attribute-based encryption". In 2007 IEEE Symposium on Security and Privacy (SP '07), pages 321–334.

Beuchat J.L., González-Díaz J.E., Mitsunari S., Okamoto E., Rodríguez-Henríquez F., Teruya T. (2010). "High-speed software implementation of the optimal ate pairing over barreto–naehrig curves". Pairing-Based Cryptography - Pairing 2010: 4th

International Conference. Proceedings, pages 21–39. Springer Berlin Heidelberg.

Blanchet B. (2014). "Automatic Verification of Security Protocols in the Symbolic Model: The Verifier ProVerif".

Boldyreva A. (2002). "Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme". Public Key Cryptography — PKC 2003, pages 31–46. Springer Berlin Heidelberg.

Boneh D., Boyen X. (2004). "Efficient selective-id secure identity-based encryption without random oracles". Advances in Cryptology - EUROCRYPT 2004, pages 223–238. Springer Berlin Heidelberg.

Boneh D., Boyen X. (2008). "Short signatures without random oracles and the sdh assumption in bilinear groups". Journal of Cryptology, 21(2):149–177.

Boneh D., Drijvers M., Neven G. (2018). "Compact multi-signatures for smaller blockchains". Advances in Cryptology – ASIACRYPT 2018, pages 435–464.

Boneh D., Franklin M. (2001). "Identity-based encryption from the weil pairing". Advances in Cryptology – CRYPTO 2001, volume 2139 of Lecture Notes in Computer Science, pages 213–229. Springer Berlin Heidelberg.

Boneh D., Gentry C., Lynn B., Shacham H. (2003). "Aggregate and verifiably encrypted signatures from bilinear maps". Advances in Cryptology – EUROCRYPT 2003, pages 416–432. Springer Berlin Heidelberg.

Boneh D., Lynn B., Shacham H. (2001). "Short signatures from the weil pairing". Advances in Cryptology – ASIACRYPT 2001, pages 514–532. Springer Berlin Heidelberg.

Boneh D., Waters B. (2006). "A fully collusion resistant broadcast, trace, and revoke system". In Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS '06, pages 211–220, New York, NY, USA.

Boneh D., Waters, B. (2007). "Conjunctive, subset, and range queries on encrypted data". Theory of Cryptography, pages 535–554. Springer Berlin Heidelberg.

Boyen X. (2003). "Multipurpose identity-based signcryption. Advances in Cryptology" - CRYPTO 2003, pages 383–399. Springer Berlin Heidelberg.

Boyen X., Waters, B. (2007). "Full-domain subgroup hiding and constant-size group signatures". Public Key Cryptography – PKC 2007, pages 1–15. Springer Berlin Heidelberg.

Boyk V., Peinado M., Venkatesan R. (1998). "Speeding up discrete log and factoring based schemes via precomputations". In Advances in Cryptology — EUROCRYPT'98: International Conference on the Theory and Application of

Cryptographic Techniques Espoo, Finland, May 31 – June 4, 1998 Proceedings, pages 221–235. Springer Berlin Heidelberg.

Brands S. (1993). "An Efficient Off-line Electronic Cash System Based On The Representation Problem". Technical report, Centrum Wiskunde & Informatica (CWI).

Brickell E. F., Gordon D. M., McCurley K. S., Wilson D. B. (1993). "Fast exponentiation with precomputation". In Advances in Cryptology — EUROCRYPT' 92: Workshop on the Theory and Application of Cryptographic Techniques Balatonfüred, Hungary, May 24–28, 1992 Proceedings, pages 200–207. Springer Berlin Heidelberg.

Camenisch J., Drijvers M., Lehmann A. (2016). "Anonymous Attestation Using the Strong Diffie Hellman Assumption Revisited". In TRUST, volume 9824 of LNCS, pages 1–20. Springer.

Canard S., Devigne J., Sanders O. (2014). "Delegating a pairing can be both secure and efficient". Applied Cryptography and Network Security: 12th International Conference, ACNS 2014, Lausanne, Switzerland, June 10-13, 2014. Proceedings, pages 549–565, Cham. Springer International Publishing.

Castella-Roca J., Herrera-Joancomarti J., Dorca-Josa A. (2006). "A secure e-exam management system". In First International Conference on Availability, Reliability and Security (ARES'06), pages 8–pp. IEEE.

Catalano T., Gatti L. (2017). "Representing teachers as criminals in the news: A multimodal critical discourse analysis of the atlanta schools' cheating scandal". Social Semiotics, 27(1):59–80.

Cavallo B., Di Crescenzo G., Kahrobaei D., Shpilrain V. (2015). "Efficient and secure delegation of group exponentiation to a single server". In Radio Frequency Identification: 11th International Workshop, RFIDsec 2015, New York, NY, USA, June 23-24, 2015, Revised Selected Papers, pages 156–173, Cham. Springer International Publishing.

" Chang C.-C., Chang S.-C., Lee J.-S. (2009). "An on-line electronic check system with mutual authentication". Computers & Electrical Engineering, 35(5):757 – 763.

Chang C.-C., Chang S.-C., Wu Y.-C. (2016). "Novel electronic check mechanism using elliptic curve cryptosystem". Journal of Computers, 27(3):111–122.

Chaum D., den Boer B., van Heyst E., Mjølsnes S., Steenbeek A. (1990a). "Efficient offline electronic checks". Advances in Cryptology — EUROCRYPT '89, pages 294–301. Springer Berlin Heidelberg.

Chaum D., Fiat A., Naor M. (1990b). "Untraceable electronic cash". Advances in Cryptology — CRYPTO' 88, pages 319–327. Springer New York.

Chen C.-L., Wu C.-H., Lin W.-C. (2010). "Improving an on-line electronic check system with mutual authentication". In Proceedings of International Conference on Advanced Information Technologies (AIT 2010).

Chen L., Kudla C. (2003). "Identity based authenticated key agreement protocols from pairings". In 16th IEEE Computer Security Foundations Workshop, 2003. Proceedings., pages 219–233.

Chen T.-H., Yeh S.-C., Liao K.-C., Lee W.-B. (2009). "A practical and efficient electronic checkbook". Journal of Organizational Computing and Electronic Commerce, 19(4):285–293.

Chen W.-K. (2005). "Efficient on-line electronic checks". Applied Mathematics and Computation, 162(3):1259 – 1263.

Chen X., Susilo W., Li J., Wong D. S., Ma J., Tang S., Tang, Q. (2015). "Efficient algorithms for secure outsourcing of bilinear pairings". Theoretical Computer Science, 562(Supplement C):112 – 121.

Chevallier-Mames B., Coron J.-S., McCullagh N., Naccache D., Scott M. (2005). "Secure delegation of elliptic-curve pairing". Cryptology ePrint Archive, Report 2005/150.

Chow S. S. M. (2005). "Verifiable pairing and its applications". In Proceedings of the 5th International Conference on Information Security Applications, WISA'04, pages 170–187, Berlin, Heidelberg. Springer-Verlag.

Chow S. S. M., Boyd C., Nieto J. M. G. (2006). "Security-mediated certificateless cryptography". Public Key Cryptography – PKC 2006, pages 508–524. Springer Berlin Heidelberg.

Deborah L J., Rawal B. S., Wang Y. (2019). "Secure online examination system for e-learning". In 2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE), pages 1–4. IEEE.

Dent A. W. (2008). "A survey of certificateless encryption schemes and security models". International Journal of Information Security, 7(5):349–377.

Diffie W., Hellman M. (1976). "New directions in cryptography". IEEE transactions on Information Theory, 22(6):644–654.

Dong M., Ren Y., Zhang X. (2017). "Fully verifiable algorithm for secure outsourcing of bilinear pairing in cloud computing". KSII Transactions on Internet and Information Systems, pages 3648–3663.

Dreier J., Giustolisi R., Kassem A., Lafourcade P., Lenzini G., Ryan P. Y. (2014). "Formal analysis of electronic exams". In 2014 11th International Conference on Security and Cryptography (SECRYPT), pages 1–12. IEEE.

El Mrabet N., Joye M. (2017). "Guide to Pairing-based Cryptography". Chapman and Hall/CRC.

ElGamal T. (1985). "A public key cryptosystem and a signature scheme based on discrete logarithms". IEEE transactions on information theory, 31(4):469–472.

Galbraith S. D., Paterson K. G., Smart N. P. (2008). "Pairings for cryptographers". Discrete Applied Mathematics, 156(16):3113–3121.

Gentry C. (2006). "Practical identity-based encryption without random oracles". Advances in Cryptology - EUROCRYPT 2006, pages 445–464. Springer Berlin Heidelberg.

Girault M., Lefranc D. (2005). "Server-aided verification: Theory and practice". Advances in Cryptology - ASIACRYPT 2005, pages 605–623. Springer Berlin Heidelberg.

Giustolisi R., Iovino V., Lenzini G. (2017). "Privacy-preserving verifiability-a case for an electronic exam protocol". In SECRYPT, pages 139–150.

Giustolisi R., Lenzini G., Bella G. (2013). "What security for electronic exams?" In 2013 International Conference on Risks and Security of Internet and Systems(CRiSIS), pages 1–5. IEEE.

Giustolisi R., Lenzini G., Ryan P. Y. (2014). "Remark!: A secure protocol for remote exams". In Cambridge International Workshop on Security Protocols, pages 38–48. Springer.

Goldwasser S., Micali S., Rivest R. L. (1988). "A digital signature scheme secure against adaptive chosen-message attacks". SIAM J. Comput., 17(2):281–308.

Goyal V., Pandey O., Sahai A., Waters B. (2006). "Attribute-based encryption for fine-grained access control of encrypted data". In Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS '06, pages 89–98, New York, NY, USA. ACM.

Hess F., Smart N., Vercauteren F. (2006). "The eta pairing revisited". Information Theory, IEEE Transactions on, 52(10):4595–4602.

Hinarejos M. F., Ferrer-Gomila J., Draper-Gil G., Huguet-Rotger L. (2012). "Anonymity and transferability for an electronic bank check scheme". In 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications, pages 427–435.

Hohenberger S., Lysyanskaya A. (2005). "How to securely outsource cryptographic computations". In Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings, volume 3378 of Lecture Notes in Computer Science, pages 264–282. Springer.

Huszti A., Petho A. (2010). "A secure electronic exam system". Publicationes Mathematicae Debrecen, 77(3-4):299–312.

Joux A. (2000). "A one round protocol for tripartite diffie–hellman". Algorithmic Number Theory, pages 385–393. Springer Berlin Heidelberg.

Kalkar O., Kiraz M. S., Sertkaya I., Uzunkol O. (2018). "A more efficient 1-checkable secure outsourcing algorithm for bilinear maps". Proceedings of The 11th WISTP International Conference on Information Security Theory and Practice (WISTP'2017).

Kalkar O., Sertkaya I. (2022) "Permissioned blockchain based remote electronic examination". Turkish Journal of Electrical Engineering and Computer Sciences, Vol. 30, No. 2, Article 3.

Kalkar O., Sertkaya I., Kavut S. T. (2020). "On the batch outsource of pairing computations". The Computer Journal.

Kang B. G., Lee M. S., Park J. H. (2005). "Efficient delegation of pairing computation". Cryptology ePrint Archive, Report 2005/259.

Kausar S., Huahu X., Ullah A., Wenhao Z., Shabir M. Y. (2020). "Fog-assisted secure data exchange for examination and testing in e-learning system". Mobile Networks and Applications, pages 1–17.

Kim S., Oh H. (2002). "A new electronic check system with reusable refunds". International Journal of Information Security, 1(3):175–188.

Kiyomura Y., Inoue A., Kawahara Y., Yasuda M., Takagi T., Kobayashi T. (2017). "Secure and efficient pairing at 256-bit security level". In Applied Cryptography and Network Security: 15th International Conference, ACNS 2017, Kanazawa, Japan, July 10-12, 2017, Proceedings, pages 59–79, Applied Cryptography and Network Security: 15th International Conference, ACNS 2017. Springer International Publishing.

Koblitz N. (1987). "Elliptic curve cryptosystems". Mathematics of computation, 48(177):203–209.

Koblitz N., Menezes A. (2005). "Pairing-Based Cryptography at High Security Levels". In Cryptography and Coding, volume 3796 of Lecture Notes in Computer Science, pages 13–36. Springer Berlin Heidelberg.

Luo X., Yang X., Niu X. (2018). "An efficient and secure outsourcing algorithm for bilinear pairing computation". In Advances in Internetworking, Data & Web Technologies: The 5th International Conference on Emerging Internetworking, Data & Web Technologies (EIDWT-2017), pages 328–339, Cham. Springer International Publishing.

Luo Y., Fu S., Huang K., Wang D., Xu M. (2016). "Securely outsourcing of bilinear pairings with untrusted servers for cloud storage". In 2016 IEEE

Trustcom/BigDataSE/ISPA, pages 623–629.

Malone-Lee J. (2002). "Identity-based signcryption". Cryptology ePrint Archive, Report 2002/098.

Mathapati M., Kumaran T. S., Kumar A. K., Kumar, S. V. (2017). "Secure online examination by using graphical own image password scheme". In 2017 IEEE International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM), pages 160–164. IEEE.

McCullagh N., Barreto P. S. L. M. (2004). "Efficient and forward-secure identity-based signcryption". Cryptology ePrint Archive, Report 2004/117.

Mefenza T., Vergnaud D. (2018). "Verifiable outsourcing of pairing computations"

. Menezes A., Okamoto T., Vanstone S. A. (1993). "Reducing elliptic curve logarithms to logarithms in a finite field". IEEE Trans. Information Theory, 39(5):1639–1646.

Miller V. S. (1985). "Use of elliptic curves in cryptography". In Conference on the theory and application of cryptographic techniques, pages 417–426. Springer.

Mitchell I., Hara S., Sheriff M. (2019). "dapper: Decentralised application for examination review". In 2019 IEEE 12th International Conference on Global Security, Safety and Sustainability (ICGS3), pages 1–14. IEEE.

Muzaffar A. W., Tahir M., Anwar M. W., Chaudry Q., Mir S. R., Rasheed Y. (2021). "A systematic review of online exams solutions in e-learning: Techniques, tools, and global adoption". IEEE Access, 9:32689–32712.

Pasupathinathan V., Pieprzyk J., Wang H. (2005). "Privacy enhanced electronic cheque system". In Seventh IEEE International Conference on E-Commerce Technology (CEC'05), pages 431–434.

Plateaux A., Lacharme P., Coquet V., Vernois S., Murty K., Rosenberger C. (2013). "An e-payment architecture ensuring a high level of privacy protection". Security and Privacy in Communication Networks, pages 305–322, Cham. Springer International Publishing.

Rackoff C., Simon D. R. (1992). "Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack". Advances in Cryptology — CRYPTO '91, pages 433–444. Springer Berlin Heidelberg.

Ren Y., Ding N., Wang T., Lu H., Gu D. (2016). "New algorithms for verifiable outsourcing of bilinear pairings". Science China Information Sciences, 59(9):99103.

Ren Y., Dong M., Niu Z., Du X. (2017). "Non-interactive verifiable outsourcing algorithm for bilinear pairing with improved checkability". Security and

Communication Networks, pages 1–9.

Sakai R., Kasahara M. (2000). "Cryptosystems based on pairing" (in japanese). In Symposium on Cryptography and Information Security SCIS'00, pages 26–28.

Schnorr C.-P. (1989). "Efficient identification and signatures for smart cards". In Conference on the Theory and Application of Cryptology, pages 239–252. Springer.

Scott M., Costigan N., Abdulwahab W. (2006). "Implementing cryptographic pairings on smartcards". In Cryptographic Hardware and Embedded Systems - CHES 2006, volume 4249 of Lecture Notes in Computer Science, pages 134–147. Springer Berlin Heidelberg.

Sertkaya I., Kalkar O. (2019). "An efficient electronic checkbook scheme with mutual authentication". Suleyman Demirel University Journal of Natural and Applied Sciences, pages 590 – 596.

Sertkaya I., Kalkar O. (2020). "Security Analysis and Attacks on Some Electronic Checkbook Schemes". under review.

Sertkaya I., Kalkar O. (2021). "A privacy enhanced transferable electronic checkbook scheme". Wireless Personal Communications, pages 1–27.

Shao J., Wei G. (2018). "Secure outsourced computation in connected vehicular cloud computing". IEEE Network, 32(3):36–41.

Shi E., Bethencourt J., Chan T.-H. H., Song D., Perrig A. (2007). "Multi-dimensional range query over encrypted data". In Proceedings of the 2007 IEEE Symposium on Security and Privacy, SP '07, pages 350–364, Washington, DC, USA. IEEE Computer Society.

Silverman J. (2009). "The Arithmetic of Elliptic Curves". Graduate Texts in Mathematics. Springer New York.

Sukadarmika G., Hartati R. S., Sastra N. P. (2018). "Introducing tamex model for availability of e-exam in wireless environment". In 2018 International Conference on Information and Communications Technology (ICOIACT), pages 163–167. IEEE.

Tian H., Zhang F., Ren, K. (2015). "Secure bilinear pairing outsourcing made more efficient and flexible". In Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '15, pages 417–426, New York, NY, USA. ACM.

Traoré I., Nakkabi Y., Saad S., Sayed B., Ardigo J. D., de Faria Quinan P. M. (2017). "Ensuring online exam integrity through continuous biometric authentication". In Information Security Practices, pages 73–81. Springer.

Tsang P. P., Chow S. S. M., Smith S. W. (2007). "Batch pairing delegation".

Advances in Information and Computer Security, pages 74–90. Springer Berlin Heidelberg.

Uzunkol O., Kalkar O., Sertkaya I. (2017). "Fully verifiable secure delegation of pairing computation: Cryptanalysis and an efficient construction". Cryptology ePrint Archive, Report 2017/1173.

Wang Y., Manulis M., Au M. H., Susilo W. (2013). "Relations among privacy notions for signcryption and key invisible "sign-then-encrypt"". Cryptology ePrint Archive, Report 2013/230.

Wang Y., Wu Q., Wong D. S., Qin B., Chow S. S. M., Liu Z., Tan X. (2014). "Securely outsourcing exponentiations with single untrusted program for cloud storage". In Computer Security - ESORICS 2014: 19th European Symposium on Research in Computer Security, Wroclaw, Poland, September 7-11, 2014. Proceedings, Part I, pages 326–343, Cham. Springer International Publishing.

Yao D., Fazio N., Dodis Y., Lysyanskaya A. (2004). "Id-based encryption for complex hierarchies with applications to forward security and broadcast encryption". In Proceedings of the 11th ACM Conference on Computer and Communications Security, CCS '04, pages 354–363, New York, NY, USA. ACM.

Yu H.-C., Hsi K.-H., Kuo P.-J. (2002). "Electronic payment systems: an analysis and comparison of types". Technology in Society, 24(3):331 – 347.

BIOGRAPHY

Öznur Kalkar achieved her Bachelor of Science and Master of Science degree in Mathematics at Koc University in 2012 and 2014, respectively. She is pursuing her PhD in Math at Gebze Technical University. If you are reading this, she completed her PhD studies. She worked at TUBITAK BILGEM as an applied cryptographer during 2015-2022 with a demonstrated history of cryptographic protocol design. During 2018-2022, she worked at TUBITAK BILGEM Blockchain Research Lab and was responsible for designing cryptographic protocols on blockchain and cryptocurrencies. Since 2022 March, she is working as an applied cryptographer at Silent Protocol. She is skilled in elliptic curve and pairing based cryptography, in particular zero-knowledge proofs, threshold cryptography, and anonymous credentials.

APPENDICES

Appendix : Publications

Kalkar Ö., Sertkaya İ., (2021), "A privacy enhanced transferable electronic checkbook scheme", Wireless Personal Communications, 123, 2895–2921.

Sertkaya İ., Kalkar Ö., (2022), "Permissioned blockchain based remote electronic examination", Turkish Journal of Electrical Engineering & Computer Sciences, Vol. 30: No. 2, Article 3

Kalkar Ö., Sertkaya İ., Tutdere S., (2022), "On the batch outsource of pairing computations", The Computer Journal