PAIRING BASED NON-REPUDIATION PROTOCOLS IN CRYPTOGRAPHY

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF APPLIED MATHEMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

ÖMER SEVER

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
CRYPTOGRAPHY

JUNE 2017

Approval of the thesis:

PAIRING BASED NON-REPUDIATION PROTOCOLS IN CRYPTOGRAPHY

submitted by **ÖMER SEVER** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Cryptography Department, Middle East Technical University** by,

Prof. Dr. Bülent KARASÖZEN _____
Director, Graduate School of **Applied Mathematics**

Prof. Dr. Ferruh ÖZBUDAK _____
Head of Department, **Cryptography**

Prof. Dr. Ersan AKYILDIZ _____
Supervisor, **Mathematics Department, METU**

Examining Committee Members:

Prof. Dr. Ferruh ÖZBUDAK
Cryptography Department, METU _____

Prof. Dr. Ersan AKYILDIZ
Mathematics Department, METU _____

Assoc. Prof. Dr. Murat CENK
Cryptography Department, METU _____

Assoc. Prof. Dr. Sedat AKLEYLEK
Computer Engineering Department, 19 Mayıs University _____

Assist. Prof. Dr. Oğuz YAYLA
Mathematics Department, Hacettepe University _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: ÖMER SEVER

Signature :

ABSTRACT

PAIRING BASED NON-REPUDIATION PROTOCOLS IN CRYPTOGRAPHY

SEVER, Ömer

Ph.D., Department of Cryptography

Supervisor : Prof. Dr. Ersan AKYILDIZ

June 2017, 85 pages

Bilinear pairing on an elliptic curve is a mapping of a pair of elements on an elliptic curve into an element of a finite field. It is called symmetric when two elements of the domain are in the same group, it is called asymmetric otherwise. Generally symmetric pairings classified as Type-I and asymmetric pairings as Type-III. Type-II is a special case of Type-III which we don't consider in this thesis.

Although the first use of bilinear pairings in cryptography has the intention to attack elliptic curve cryptosystems, in recent years they have been widely used to construct new encryption and signature schemes. As a main building block for non-repudiation protocols, signatures with different properties are implemented by using pairings on elliptic curves. Verifiably encrypted signature scheme due to Chen and Gu is a typical example for such a pairing based implementations.

In the first part of this thesis, we propose an adaptation of certificateless public key cryptography to hybrid verifiably encrypted signature scheme due to Chen and Gu. This is called CL-HVESS. Then we expand CL-HVESS to Type-III pairings to mitigate the risks of recent attacks on Type-I pairings. In addition to this, we also present a replay attack to Chen and Gu protocol.

In the second part we propose a non-repudiation protocol which has a new structure based on pairing based cryptography. The hybrid structure consists of two rounds; first round runs with an online Trusted Third Party (TTP) then second and next rounds run with offline TTP. Our contribution here is the usage of signed Joux Tri-partite key exchange scheme in the first round as a security enhancing method.

In the third part we propose a new scheme that combines signcryption and verifiably encrypted signatures which we call VESigncrypt. We use it in a fair secret contract signing protocol. VESigncrypt has single recipient, multi recipient and publicly verifiable versions. To the best of our knowledge, this scheme is the first of its kind in the literature.

In the last part of the thesis we first present a survey for isogeny based cryptography. Then, we propose a new verifiably encrypted probabilistic signature scheme based on isogenies. Finally we present new signature and verifiably encrypted signature schemes based on isogeny pairing groups.

Keywords: Bilinear Pairings, Non-Repudiation Protocols, Verifiably Encrypted Signatures, Signcryption, Isogeny Based Cryptography

ÖZ

KRİPTOGRAFİDE EŞLEME TABANLI İNKAR EDEMEZLİK PROTOKOLLERİ

SEVER, Ömer

Doktora, Kriptografi Bölümü

Tez Yöneticisi : Prof. Dr. Ersan AKYILDIZ

Haziran 2017 , 85 sayfa

Eliptik eğriler üzerinde tanımlı çift doğrusal eşleme, eliptik eğri üzerindeki bir çift noktanın, bir sonlu alan elemanına eşleştirilmesidir. Eşlemenin tanım bölgesindeki her iki eleman da aynı gruptansa, eşleme simetrik olarak adlandırılır, farklı gruptansa asimetrik olarak adlandırılır. Genellikle simetrik eşlemeler Tip-1, asimetrik eşlemeler Tip-3 olarak sınıflandırılır. Tip-2 bu tezde ele alınmayan, Tip-3'ün özel bir durumudur.

Kriptografide çift doğrusal eşlemelerin ilk kullanımı eliptik eğrisi kripto sistemlerine saldırı maksatlı olsa da son yıllarda, yeni şifreleme ve imzalama teknikleri oluşturmak için yaygın şekilde kullanılmıştır. Farklı özelliklere sahip imzalar, inkar edememe protokollerinin ana unsuru olarak, eliptik eğriler üzerinde eşlemeler kullanılarak gerçeklenmektedir. Chen ve Gu'ya ait doğrulanabilir şifreli imzalama tekniği bu çeşit eşleme tabanlı uygulamalara tipik bir örnektir.

Bu tezin ilk bölümünde sertifikasız açıkanahtar kriptografinin Chen ve Gu'nun hibrid doğrulanabilir şifreli imzalama tekniğine adaptasyonu teklif edilmektedir. Bu adaptasyon CL-HVESS olarak adlandırılmıştır. Sonrasında CL-HVESS Tip-III eşlemelerle kullanıma verilerek Tip-I eşlemelere son dönemde yapılan saldırılardan doğan riskler bertaraf edilmektedir. Birinci bölümde son olarak Chen ve Gu'nun protokolüne tekrarlama saldırısı sunulmaktadır.

İkinci bölümde eşleme tabanlı kriptografiye dayanan yeni yapısal inkar edemezlik protokolü sunulmaktadır. Hibrid yapı; ilki çevrim-içi, ikincisi çevrim-dışı olmak üzere iki safhadan oluşmaktadır. Bu

kısımdaki katkı, Joux'nun üçlü anahtar paylaşım meknizmasının güvenlik artırıcı metod olarak kullanımıdır.

Üçüncü bölümde şifrelimzalama ve doğrulanabilir şifreli imzalamayı birleştiren ve adına VESigncrypt dediğimiz yeni bir teknik önerilmektedir. Önerilen teknik adil gizli kontrat imzalama protokolünde kullanılmaktadır. VESigncrypt'in tek alıcılı, çok alıcılı ve herkes tarafından doğrulanabilir versiyonları bulunmaktadır. Bildiğimiz kadarıyla, bu teknik literatürde türünün ilk örneğidir.

Son bölümde önce eşgen tabanlı kriptografinin özeti sunulmaktadır. Sonrasında olasılıklı ve eşgen tabanlı yeni bir şifrelimzalama tekniği önerilmektedir. Bu çalışmada son katkımız eşgen eşleme gruplara dayalı yeni imzalama ve şifrelimzalama teknikleridir.

Anahtar Kelimeler: Çift-Doğrusal Eşlemeler, İnkar Edemezlik Protokolleri, Doğrulanabilir Şifreli İmzalar, Şifrelimzalama, Eşgen Tabanlı Kriptografi

To my wife and daughter

Hülya Sever, Yeliz Sever

ACKNOWLEDGMENTS

I would like to thank my supervisor Professor Ersan Akyıldız for the continuous support of my Ph.D study, for his patience, motivation, and immense knowledge. It was a great honor to work with him and our cooperation influenced my academical and world view highly.

TABLE OF CONTENTS

ABSTRACT v

ÖZ vii

ACKNOWLEDGMENTS x

TABLE OF CONTENTS xi

LIST OF TABLES xviii

LIST OF FIGURES xix

LIST OF ABBREVIATIONS xx

CHAPTERS

1 INTRODUCTION 1

2 BASICS FOR NON-REPUDIATION AND PAIRINGS 3

2.1 Mathematical Background 3

2.1.1 Bilinear Pairings 3

2.1.2 Pairing Types 3

2.1.3 Pairing Friendly Curves 5

2.1.4 Security of Pairing 6

2.1.4.1 Problems for elliptic curves 6

2.1.4.2 Problems for pairings 7

2.2 Pairing Based Schemes 7
2.2.1 Joux's One Round Protocol For Tri-partite Key Exchange 7
2.2.2 Verifiably Encrypted Signature 8
2.2.3 Signcryption 8
2.2.4 Short Signatures 9
2.3 Non-Repudiation 10
2.3.1 Properties of Non-repudiation Protocols 10
2.3.1.1 Trusted Third Party Involvement 10
2.3.1.2 Fairness 10
2.3.1.3 Transparency / Verifiability 11
2.3.1.4 Timeliness 11
2.3.1.5 State Storage 11
2.3.1.6 Communication Channels 12
2.3.1.7 Efficiency 12
2.3.1.8 Confidentiality 12
2.3.2 Certified E-Mail 12
2.3.2.1 CEM Protocols with Inline TTP 13
2.3.2.1.1 Bahreman and Tygar's CEM Protocol 13
2.3.2.1.2 Cimato et al. CEM Protocol 13
2.3.2.2 CEM Protocols with Online TTP 13
2.3.2.2.1 Oppliger and Stadlin's CEM Protocol [31] . . 14
2.3.2.2.2 Deng et al. CEM Protocol [33] 14

2.3.2.3 CEM Protocols with Offline TTP 15
2.3.2.3.1 Asokan, Schunter and Waidner's CEM Protocol [34] 15
2.3.2.3.2 Bao, Deng and Mao's CEM Protocol [35] . . 15
2.3.2.3.3 VRES Protocol[66] 15
2.3.3 Contract Signing 17
2.3.3.1 Ateniese Protocol 17
2.3.3.2 Garay-Jakobsson-McKenzie Protocol 18
2.3.4 Fair Exchange 19
2.3.4.1 Kupcu Protocol 19
2.3.4.2 Alaraj Protocol 20
2.4 Implementation and Emerging Areas 22
2.4.1 Emerging Areas 22
2.4.2 Implementation 23
3 IMPROVED CONTRACT SIGNING PROTOCOL BASED ON CERTIFICATELESS HYBRID VERIFIABLY ENCRYPTED SIGNATURE SCHEME 27
3.1 Introduction 27
3.2 An Adaptation of Certificateless Public Key Cryptography to HVESS 27
3.3 Expansion of CL-HVESS to Type-III Pairings 29
3.4 Attack and Improvement to Fair Contract Signing Protocol 30
3.4.1 Attack to Contract Signing Protocol 30
3.4.2 Improvement to Contract Signing Protocol 31
3.4.3 Analysis of Protocol 32

3.5 Conclusion 32

4 HYBRID NON-REPUDIATION PROTOCOL WITH PAIRING BASED CRYPTOGRAPHY 33

4.1 Introduction 33

4.2 Protocol Definition 33

4.2.1 Notation 34

4.2.2 Protocol Description 34

4.2.2.1 Online Round 35

Cancellation Sub-protocol 36

Dispute Resolution 36

4.2.2.2 Off-line Round 37

Cancellation Sub-protocol 37

Dispute Resolution 38

4.3 Protocol Analysis 38

4.3.1 Fairness and Non-Repudiation 38

4.3.2 Timeliness 39

4.3.3 TTP State 39

4.3.4 Efficiency and Comparison 39

4.3.5 Key escrow and Revocation 39

4.3.6 Confidentiality 40

4.4 Certificateless ID-Based Signature and Encryption Scheme 40

4.4.1 Certificateless ID-Based Encryption 40

4.4.1.1 Encryption . . . 40
4.4.1.2 Decryption . . . 41
4.4.1.3 Proof of Decryption . . . 41
4.4.2 Certificateless ID-Based Signature . . . 41
4.4.2.1 Signature . . . 41
4.4.2.2 Verification . . . 41
4.4.2.3 Proof of Verification . . . 42
4.5 Conclusion . . . 42
5 VERIFIABLY ENCRYPTED SIGNCRYPTION . . . 43
5.1 Introduction . . . 43
5.2 General Description . . . 43
5.2.1 Signcryption . . . 43
5.2.2 Verifiably Encrypted Signatures . . . 43
5.3 Verifiably Encrypted Signcryption Scheme . . . 44
5.4 Multi-Recipient Verifiably Encrypted Signcryption Scheme . . . 45
5.5 Fair Two-Party Secret Contract Signing Protocol . . . 47
5.5.1 First Case . . . 47
5.5.2 Second Case . . . 48
5.6 Security and Performance Analysis . . . 49
5.6.1 Confidentiality of VESigncrypt . . . 49
5.6.2 Unforgeability of VESigncrypt . . . 50
5.6.3 Opacity of VESigncrypt . . . 50

5.6.4 Performance Analysis 50

5.7 Public Verifiable Verifiably Encrypted Signcryption Scheme 52

5.8 Conclusion 53

6 POST-QUANTUM ASPECTS OF PAIRINGS 55

6.1 Post-Quantum Cryptography 55

6.2 Isogeny Based Cryptography 55

6.2.1 Problems for Isogeny 55

6.2.2 Isogeny Based Schemes 56

6.2.2.1 Key Exchange Scheme 56

6.2.2.2 Man-In-The-Middle Attack 57

6.2.2.3 Public Key Encryption Scheme 58

6.2.2.4 Strong Designated Verifier Signature Scheme 59

6.2.2.5 Undeniable Signature Scheme 60

6.2.2.6 Undeniable Blind Signature Scheme 61

6.2.2.7 Isogeny Based Signature Schemes 63

6.2.2.8 Comparison of Isogeny Based Signature Schemes . . . 67

6.2.2.9 New VES Construction Based on Isogeny Signature Scheme 67

6.3 Isogeny Pairing Groups 69

6.3.1 Problems for Isogeny Pairing Groups 70

6.3.2 Isogenous Pairing Groups 70

6.3.2.1 Identity Based Encryption Scheme 71

6.3.2.2 New Identity Based Signature Scheme 72

6.3.2.3 New Identity Based Verifiably Encrypted Signature Scheme 73

REFERENCES 77

CURRICULUM VITAE 85

LIST OF TABLES

TABLES

Table 2.1 Comparison of the properties of CEM protocols 16

Table 5.1 Comparison of our scheme with [51] for single recipient 51

Table 5.2 Comparison of our scheme with [51] for multi-recipient 51

Table 5.3 Running times of our scheme with different supersingular curves 51

Table 6.1 Comparison of parameter sizes 67

LIST OF FIGURES

FIGURES

Figure 2.1 TTP Involvement 11

Figure 2.2 Bahreman and Tygar's protocol 13

Figure 2.3 Cimato et al. protocol 14

Figure 2.4 Oppliger and Stadlin's protocol 14

LIST OF ABBREVIATIONS

ABE	Attribute Based Encryption
ABS	Attribute Based Signature
CEM	Certfied E-Mail
CL-HVESS	Certificateless Hybrid Verifiably Encrypted Signature Scheme
CL-PKC	Certificateless Public Key Cryptography
DH	Diffie-Hellman
DSA	Digital Signature Algorithm
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
EOO	Evidence of Origin
EOR	Evidence of Receipt
GMP	GNU Multiple Precision Arithmetic Library
IBC	Identity Based Cryptography
IoT	Internet of Things
IPG	Isogeny Pairing Groups
KGC	Key Generation Center
NRD	Non-Repudiation of Delivery
NRO	Non-Repudiation of Origin
NRR	Non-Repudiation of Receipt
NRS	Non-Repudiation of Submission
PBC	Pairing Based Cryptography
PKC	Public Key Cryptography
PKG	Private Key Generator
TTP	Trusted Third Party
VES	Verifiably Encrypted Signcryption
WSN	Wireless Sensor Networks

CHAPTER 1

INTRODUCTION

Non-repudiation protocols are used for exchange of information with evidence of non-repudiation. Application of Non-repudiation protocols are spreaded over Certfied E-mail (CEM), Electronic Contract Signing, e-commerce and electronic payment. Although there are many different types of Non-repudiaition protocols such as Certified E-mail, Contract signing, fair exchange, differing in their goals; they are related with each other and share the properties Non-repudiation and fairness in common. To show these differences with an example; when non-repudiation protocol is based on message delivery like in Certified E-mail, the receiver has to provide Non-Repudiation of Receipt (NRR) in order to get the message and obtain the Non-Repudiation of Origin (NRO) for that message. But when non-repudiation protocol is based on exchange of evidence of non-repudiation not the message itself like in contract signing, obtaining a message content is not important but exchanging signed message/contract fairly is the main goal of the application.

Non-repudiation is defined as a security service by which the entities involved in a communication can not deny having participated, spcifically, the sender can not deny having sent a message and the receiver can not deny having received a message by NIST in [1].
Non-repudiation is primarily depending on asymmetric cryptography specifically to signatures which are accepted as evidences. Regarding how used in a protocol, evidence of origin supplies NRO and evidence of receipt supplies NRR.

Non-Repudiation Protocols can satisfy various properties in different ways like:

- Fairness: Strong, weak, light
- Non-Repudiation: NRO, NRR, NRS, NRD
- State storage: Statefull, stateless
- Timeliness: Synchronous, Asynchronous
- TTP Inclusion: In-line, On-line, Off-line, Probabilistic

These properties and non-repudiation protocols have been studied in [2], [3], [25] [44] and [5].

As a kind of non-repudiation protocol, contract signing share similar properties with other protocols. The goal of contract signing protocols is exchange of evidence of non-repudiation not the message itself. Differing from certified e-mail or fair exchange in the sense that obtaining message content is

not important but exchanging signed message/contract fairly is the main goal of the contract signing protocol.

Contract signing protocols are being widely used over digital environment and treated as an application of non-repudiation protocols. As a kind of non-repudiation protocols, the most important property of contract signing protocols is fairness. Verifiably encrypted signatures are used mainly for fair exchange and contract signing protocols to sustain fairness in cryptograhic manner. Although cofidentiality of the message is not as important as fairness for ordinary contracts, in the case of secret contracts confidentiality will be as important as fairness. Signcryption as a cryptoraphic method combines signing and encryption usually in sing then encrypt order. In this work we propose a new scheme (to the best of our knowledge, this scheme is the first of its kind in the literature) that combines signcryption and verifiably encrypted signatures which we call VESigncrypt.

In the non-repudiation protocols public key cryptography (PKC) plays a crucial role. Signatures and encryption can implemented by conventional public key techniques like DSA, ECDSA, DH, ECDH or by pairing based techniques. Pairing based cryptography (PBC) brings some new features (like short signatures, ID-Based cryptography) or ease to design new schemes (like signcryption, blind signatures). PKC is generally based on certificates binding identities with public keys which are approved by Certificate Authorities. Differing from classical PKC, in ID-Based Cryptography public keys are dependant on user identities and/or identifiers. This difference brings advantages and disadvantages together as discussed in [15]. The advantages of ID-Based Cryptography are mainly achieving different encryption and signature schemes like ID-Based encryption [16], blind [17], short [18], ring [19] and verifiably encrypted [23], [45] signatures which are summarized in [4]. The disadvantage of ID-Based cryptography is if the public key is dependant only on identity of a user, key generator knows the private keys of users.

CHAPTER 2

BASICS FOR NON-REPUDIATION AND PAIRINGS

2.1 Mathematical Background

2.1.1 Bilinear Pairings

Pairings in elliptic curve cryptography are functions which map a pair of elliptic curve points to an element of the multiplicative group of a finite field. Below is the simple definition of a bilinear pairing , more information on pairings like Weil or Tate pairings, divisiors and curve selection can be found in [6] as a summary and in [46] in more details.

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be additive abelian group of order q and $\mathbb{G}_3$ be multiplicative group of order q, a pairing is a function

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3 \tag{2.1}$$

which satisfies the following properties:

a) e is bilinear: For all $P, S \in \mathbb{G}_1$ and $Q, T \in \mathbb{G}_2$ we have $e(P + S, Q) = e(P, Q)e(S, Q)$ and $e(P, Q + T) = e(P, Q)e(P, T)$

b) e is non-degenerate: For all $P \in \mathbb{G}_1$, with $P \neq 0$ there is some $Q \in \mathbb{G}_2$ such that $e(P, Q) \neq 1$ and for all $Q \in \mathbb{G}_2$, with $Q \neq 0$ there is some $P \in \mathbb{G}_1$ such that $e(P, Q) \neq 1$

Consequtive properties of bilinearity are:

- $e(P, 0) = e(0, Q) = 1$
- $e(-P, Q) = e(P, Q)^{-1} = e(P, -Q)$
- $e([a]P, Q) = e(P, Q)^a = e(P, [a]Q)$ for all $a \in \mathbb{Z}$

Remark: For the cryptographic use, we also require that $e(P, Q)$ can be computed efficiently.

2.1.2 Pairing Types

When we denote pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$ with three groups of prime order n and elliptic curve $E(\mathbb{F}_q)$ is defined over field of size q; consequently $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_3$ are generally subgroups of $E(\mathbb{F}_q)$, $E(\mathbb{F}_{q^k})$ and $\mathbb{F}^*_{q^k}$, respectively. In this work we will use the type definitions given in [47] as;

Type-1 $\mathbb{G}_1 = \mathbb{G}_2$

Type-2 $\mathbb{G}_1 \neq \mathbb{G}_2$ there is an efficiently computable homomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$ but there is no such homomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2$. (The situation when there is such a homomorphism is considered as a case of Type-1.)

Type-3 $\mathbb{G}_1 \neq \mathbb{G}_2$ and there is not any efficiently computable homomorphism between $\mathbb{G}_2$ and $\mathbb{G}_1$

While, Type-1 pairing is defined over supersingular elliptic curves $E(\mathbb{F}_q)$, Type-2 and Type-3 pairings are defined over ordinary curves. In Type-2 pairings the trace map from $\mathbb{G}_2$ to $\mathbb{G}_1$ is the required homomorphism. Generally kernel of the trace map is taken as the $\mathbb{G}_2$ in Type-3 and thus trace map is trivial and generally there is not an efficiently computable homomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$.

In this work we used mainly Type I [47] supersingular curves for pairing instantiation in which $\mathbb{G}_1 = \mathbb{G}_2$, since the representation of schemes are easier. In this type $\mathbb{G}_1$ is a subgroup of $E(\mathbb{F}_q)$. There is a distortion map ψ which maps $\mathbb{G}_1$ into $E(\mathbb{F}_{q^k})$ and the modified pairing $\hat{e}(P,Q) : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_3$ for $P, Q \in \mathbb{G}_1$ is defined by:
$\hat{e}(P,Q) = e(P, \psi(Q))$ as shown in section X in [46].

Some of the well known pairing methods are; Weil, Tate, Ate, Eta pairing. The first pairing type introduced by Weil in 1940's for a proof and the algorithm to implement Weil pairing was introduced by Miller in 1986. Both the definition (Chapter 2. Defn.II.12) and algorithm (Chapter 2. Alg.II.1) as described in [12] are given below:

Definition 2.1.1. (Weil Pairing)

Let E be an elliptic curve over $\mathbb{F}_q$, for $r \in \mathbb{N}$ with $gcd(r, q-1) = 1$, let $m \in \mathbb{N}$ be the smallest integer such that $E[r] \subset E(\mathbb{F}_{q^m})$ and let μ_r be the group of *r*th root of unity in $\mathbb{F}_{q^m}$ then Weil pairing is defined as;
$w_r : E[r] \times E[r] \to \mu_r \in \mathbb{F}_{q^m} : (P,Q) \to w_r(P,Q) = \frac{f(D_Q)}{g(D_P)}$ where $div(f) = rD_P, div(g) = rD_Q$ and $D_P \equiv (P) - (O), D_Q \equiv (Q) - (O)$ with an empty support. Here O stands for the identity element of E and D_P, D_Q are divisors on E.

And following the proposition [12] II.14, when $P \neq Q$
$w_r(P,Q) = (-1)^r \frac{f_{r,P}(Q)}{f_{r,Q}(P))}$

Algorithm 1: Miller Algorithm to compute $f_{n,P}(D)$

Input: $n \in \mathbb{N}$, $P \in E[r]$ and divisor D with $Supp(D) \cap (P, O) = \emptyset$
Output: $f_{n,P}(D)$

Represent $n = \sum_{j=0}^{L} n_j 2^j$, with $n_j \in (0,1)$ and $n_L = 1$

$T \leftarrow P; f \leftarrow 1$

for $jfromL-1to0$

$f \leftarrow c^2.l_{T,T}(D)/v_{[2]T}(D)$

$T \leftarrow [2]T$

if $n_j = 1$ **then**

$f \leftarrow f.l_{T,P}(D)/v_{T\oplus P}(D)$

$T \leftarrow T \oplus P$

fi

endfor

return f

2.1.3 Pairing Friendly Curves

It is well known that the number of points of the elliptic curve E over $\mathbb{F}_q$ is given by $\#E(\mathbb{F}_q) = q+1-t$ and by Hasse's theorem $|t| \leq 2\sqrt{q}$. If the characteristics of E,p devides t then $E(\mathbb{F}_q)$ is said to be supersingular, otherwise it is called ordinary. The following theorem ([59]) gives the classification of supersingular curves over any finite field.

Theorem 2.1.1. *Let E be a supersingular elliptic curve over $\mathbb{F}_q$ of order $q+1-t$ where $q = p^m$. Then supersingular curves are divided into six classes regarding embedding degree $k \leq 6$:*

1. *k=1; $t^2 = 4q$ and m is even*
2. *k=2; $t = 0$ and $E(\mathbb{F}_q) \cong \mathbb{Z}_{q+1}$*
3. *k=2; $t = 0$ and $E(\mathbb{F}_q) \cong \mathbb{Z}_{(q+1)/2} \oplus \mathbb{Z}_2 and q = 3(mod 4)$*
4. *k=3; $t^2 = q$ and m is even*
5. *k=4; $t^2 = 2q$ and $p = 2$ and m is odd*
6. *k=6; $t^2 = 3q$ and $p = 3$ and m is odd*

The classification is dependent on the embedding degree k of a curve $E(\mathbb{F}_q)$ where it can be defined as the embedding degree of the subgroup of $E(\mathbb{F}_{q^k})$ of order r, where r is the largest prime divisor of $\#E(\mathbb{F}_{q^k})$. And formal definition of embedding degree is as follows;

Definition 2.1.2. (Embedding Degree)

Let $\mathbb{G} =< g >$ be a cyclic subgroup of $E(\mathbb{F}_q)$ of order r. Then the embedding degree of $\mathbb{G}$ is the smallest positive integer k such that $E[r] \subset E(\mathbb{F}_{q^k})$. With the following theorem it means that if $gcd(r, q-1) = 1$ the embedding degree is the smallest k such that $r|q^k - 1$

Theorem 2.1.2. *Let $\mathbb{G} =< g >$ be a cyclic subgroup of $E(\mathbb{F}_q)$ of order r with $gcd(r, q-1) = 1$. Then $E[r] \subset E(\mathbb{F}_{q^k})$ if and only if $r|q^k - 1$.*

The characteristics of the isomorphism classes for supersingular curves over $\mathbb{F}_q$ of characteristic 2 and 3 (respectively) have been given in [60] and [61] (respectively). Supersingular curves over prime fields $p \geq 5$ exists and has the property $\#E(\mathbb{F}_p) = p + 1$, namely $t = 0$ and embedding degree $k = 2$ ([62][Theorem 13.12]).

Special techniques are needed to produce pairing friendly ordinary elliptic curves such as complex multiplication. This method is used to construct curves with endomorphism ring isomorphic to a given order in a quadratic imaginary field $Q(\sqrt{-D})$ with determined number of points, where D is the discriminant of the characteristic polynomial. Some of the examples of constructed pairing friendly ordinary elliptic curves are;

- Miyaji, Nakabayashi and Takano (MNT) [63] were the first to give an explicit construction of prime field ordinary elliptic curves with the embedding degrees k = 3, 4, 6 suitable for pairings.
- Freeman [64] gives a sparse faimily of ordinary elliptic curves over prime fields with embedding degree k = 10.
- Cocks and Pinch [20] gives a general construction of curves with different embedding degrees. But in this method $\rho = log(q)/log(r) \approx 2$ which leads to inefficient implementation. But two extensions of this method Scott and Barreto [21] and Brezing and Weng [22] gave efficient constructions with first fixing embedding degree k.

In general Freeman et al.[65] represented curve parameters t, r, q as polynomials t(x), r(x), q(x) and classified the pairing friendly elliptic curves. Definition of pairing friendly elliptic curve is given by:

Definition 2.1.3. (Pairing Friendly)

Let $E(\mathbb{F}_q)$ be an elliptic curves with characteristic p with embedding degree k. E is called pairing friendly if the following conditions hold;

1. For a prime r such that $r|\#E(\mathbb{F}_q), r \geq \sqrt{q}$
2. $k < \log_2(r)/8$

The definition is restating the need for small embedding degree and a subgroup of large prime order.

2.1.4 Security of Pairing

2.1.4.1 Problems for elliptic curves

For elliptic curves following problems are defined in additive group $\mathbb{G}_1$ of order q.

1. **Discrete Logarithm Problem (DLP):** For $P, Q \in \mathbb{G}_1$, find $x \in \mathbb{Z}_q^*$ such that $Q = [x]P$ whenever such x exists.

2. **Decisional Diffie-Hellman Problem (DDHP):** For $P \in \mathbb{G}_1$ and $x, y, z \in \mathbb{Z}_q^*$ given $P, [x]P$, $[y]P, [z]P$ decide whether $z \equiv xy \ mod(q)$

3. **Computational Diffie-Hellman Problem (CDHP):** For $P \in \mathbb{G}_1$ and $x, y \in \mathbb{Z}_q^*$ given $P, [x]P, [y]P$ compute $[xy]P$

2.1.4.2 Problems for pairings

Following problems are defined for pairings $\hat{e}(P, Q) : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_3$

1. **Bilinear Diffie-Hellman Problem (BDHP):** For $P \in \mathbb{G}_1$ and $x, y, z \in \mathbb{Z}_q^*$ given $P, [x]P$, $[y]P, [z]P$ compute $\hat{e}(P, P)^{xyz}$

2. **Decisional Bilinear Diffie-Hellman Problem (DBDHP):** For $P \in \mathbb{G}_1$ and $x, y, z, r \in \mathbb{Z}_q^*$ given $P, [x]P, [y]P, [z]P, r$ decide whether $r \equiv \hat{e}(P, P)^{xyz} mod(q)$

3. **Bilinear Inverse-Square Diffie-Hellman Problem (BISDHP):** For $P \in \mathbb{G}_1$ and $x, y \in \mathbb{Z}_q^*$ given $P, [x]P, [y]P$ compute $\hat{e}(P, P)^{x^{-2}y}$

We note that it was proved in [8] that BISDHP is polynomial time equivalent to BDHP.

2.2 Pairing Based Schemes

2.2.1 Joux's One Round Protocol For Tri-partite Key Exchange

Joux protocol [42] is known as the first implementation of pairings in cryptography. Although it is vulnerable to man-in-the-middle attack, like Diffie-Hellman key exchange protocol, it is a breakthrough for pairing based cryptography just like Diffie-Hellman protocol for public key cryptography.

Scheme 1: Joux's One Round Tri-partite Key Exchange

Setup: Type-1 pairing $\hat{e}(P, Q) : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_3$ where $(\mathbb{G}_1, +), (\mathbb{G}_3, .)$ both of order q and $\mathbb{G}_1 =< P >$. Three parties X, Y, Z with secret keys $x, y, z \in \mathbb{Z}_q^*$ respectively executes the protocol below.
Protocol:

1. X sends $[x]P$ to Y and Z. Y sends $[y]P$ to X and Z. Z sends $[z]P$ to X and Y.

2. X computes $\hat{e}([y]P, [z]P)^x)$, Y computes $\hat{e}([x]P, [z]P)^y$ and Z computes $\hat{e}([x]P, [y]P)^z$

3. Agreed key is $\hat{e}(P, P)^{xyz})$

2.2.2 Verifiably Encrypted Signature

Boneh et al [70] were first to introduce verifiably encrypted signature. In a protocol sender S can send an verifiably encrypted signature to a receiver R. The receiver R can check that verifiably encrypted signature validity but can not get the actual signature without help of an Adjudicator. When the receiver requests from the adjudicator to adjudicate, he can recover the actual signature from verifiably encrypted signature. Below is the verifiably encrypted sigrature in [70] converted to Type-1 and additive format.

Scheme 2: Verifiably Encrypted Sigrature

Setup: Type-1 pairing $\hat{e}(P,Q) : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_3$ where $(\mathbb{G}_1, +), (\mathbb{G}_3, .)$ both of order q and $\mathbb{G}_1 =< P >$. Signer has private/public key pair $(x, [x]P)$, Adjudicator has private/public key pair $(x', [x']P)$. And a hasf function $H : \{0,1\}^* \rightarrow \mathbb{G}_1$
Protocol:
Signature: Signer computes normal signature with his private key, $\sigma = [x]H(m)$.
Verification: Receiver can verify normal signature by public key of signer, $\hat{e}(P,\sigma) = \hat{e}([x]P, H(m))$
Verifiably Encrypted Sigrature: Select a random r. Compute;

- $\sigma = [x]H(m)$
- $\mu = [r]P$
- $\sigma' = [r][x']P$
- $\omega = \sigma + \sigma'$
- Verifiably encrypted signature is (ω, μ).

Verifiably Encrypted Sigrature Verification: Check if $\hat{e}(P,\omega) = \hat{e}([x]P, H(m)).\hat{e}([x']P, \mu)$
Adjudication: First check the validity of (ω, μ) with private key x'. Afterwards ordinary signature is $\sigma = \omega - [x']\mu = \sigma + \sigma' - [x'][r]P = \sigma + \sigma' - \sigma' = \sigma$.

2.2.3 Signcryption

Malone-Lee has described Identity-Based Signcryption by combining Boneh and Franklin's Identity-Based Encryption with a version of Hess's Identity-Based Signature.

Scheme 3: Identity-Based Signcryption

Setup: Type-1 pairing $\hat{e}(P,Q) : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_3$ where $(\mathbb{G}_1, +), (\mathbb{G}_3, .)$ both of order q and $\mathbb{G}_1 =< P >$. Trusted authority generates the secret master key $t \in \mathbb{Z}_q^*$ and public key $(Q_{TA} = [t]P)$ Sender has public/private (Q and S) key pair $(Q_S = H_1(ID_S), S_S = [t]Q_S)$, Receiver has public/private key pair $(Q_R = H_1(ID_R), S_R = [t]Q_R)$. $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$, $H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$, $H_3 : \mathbb{G}_3 \rightarrow \{0,1\}^*$

are hash functions.
Protocol:
Signcrypt: Select a random x. Compute;

- $[x]P \rightarrow U$
- $[H_2](U||m) \rightarrow r$
- $[x]Q_{TA} \rightarrow Z$
- $[r]S_S + Z \rightarrow V$
- $\hat{e}(Z, Q_R) \rightarrow y$
- $[H_3](y) \rightarrow k$
- $k \oplus m \rightarrow c$
- $(c, V, U) \rightarrow \sigma$

UnSigncrypt: Parse received σ as (c, V, U)

- $\hat{e}(S_R, U) \rightarrow y$
- $[H_3](y) \rightarrow k$
- $k \oplus c \rightarrow m$
- $[H_2](U||m) \rightarrow r$

Check if $\hat{e}(V, P) = \hat{e}(Q_S, Q_T A)^r . \hat{e}(U, Q_T A)$

2.2.4 Short Signatures

Pairing based cryptography has introduced the short signatures which has generally half size of ECDSA or DSA signatures. The first short signature scheme [18] described below has signature in bit length about $\rho log(r) = log(q)$ where $\rho = log(q)/log(r)$.

Scheme 4: BLS Short Signature

Setup: Type-1 pairing $\hat{e}(P, Q) : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_3$ where $(\mathbb{G}_1, +), (\mathbb{G}_3, .)$ both of order q and $\mathbb{G}_1 =< P >$. with hash function $H_1 : \{0, 1\}^* \rightarrow \mathbb{G}_1$
Protocol:
Key Generation: Select a random x as secret key and compute [x]P as the public key;
Sign: $\sigma = [x]H(m)$ is signature
Verify: With ([x]P, m, σ); check if $\hat{e}(P, \sigma) = \hat{e}([x]P, H(m)$.

Another short signature was proposed by Akleylek et al. [8] which depends on BISDHP. Here we give the description of the scheme, the security proof and comparison with other schemes is presented in [8].

Scheme 5: AKSY Short Signature

Setup: Let $(\mathbb{G}_1, +)$ and $(\mathbb{G}_2, \cdot)$ be cyclic groups of prime order n, $P \in \mathbb{G}_1$, $\mathbb{G}_1 =< P >$ and $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ be a bilinear map. Let $H : Z_2^{\infty} \rightarrow Z_2^{\lambda}$, where $160 \leq \lambda \leq log(n)$ be a cryptographic hash function.
Protocol:
Key Extraction: Signer selects $x \in \mathbb{Z}_n$ randomly and calculates $P_{pub1} = x^2 P$ and $P_{pub2} = 2xP$. Signer extracts P, P_{pub1} and P_{pub2} as the public keys and keeps x as the secret key.
Sign: Signer calculates the signature, $s = (H(m) + x)^{-2} P$ by a private key x for a message m.
Verify: Verifier can verify the signature for a pair (m, s) with public keys P, P_{pub1} and P_{pub2}, if

$$e(H(m)^2 P + P_{pub1} + P_{pub2} H(m), s) = e(P, P) \text{ holds.}$$

2.3 Non-Repudiation

2.3.1 Properties of Non-repudiation Protocols

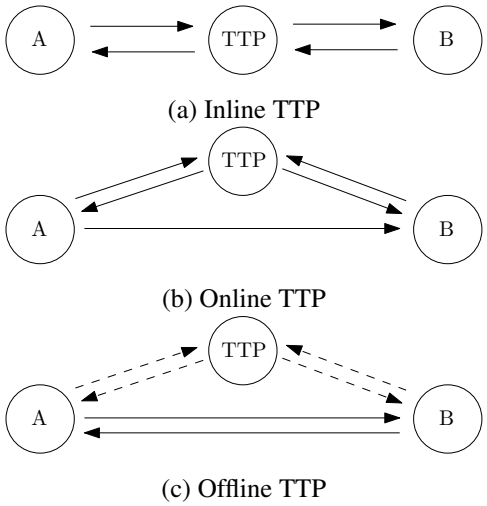
Non-repudiation and fairness are absolutely necessary properties of non-repudiation protocols. Some properties are generally required and common between different non-repudiation protocols but some are optionally required like confidentiality. In this section we define the basic properties of non-repudiation protocols and categorize with respect how these properties satisfied or not.

2.3.1.1 Trusted Third Party Involvement

A Trusted Third Party (TTP) is said to be *Inline* if all the protocol messages pass through the TTP, i.e. the communicating parties do not exchange messages bypassing the TTP. In protocols with *Online* TTP, the TTP takes part in the protocol but not necessarily in all of the steps. Protocols with *Offline* TTP do not make use of TTP at the protocol execution, however, the TTP is used if a dispute arises or protocol execution is interrupted due to a network error, etc. Message flows in all types of TTP's are depicted in Figure 2.1. In the figure, dashed lines for *Offline* TTP indicate that these steps are not part of the usual protocol execution, they are executed only in case of an exception. Offline protocols are also called *optimistic protocols* because TTP is not needed if sender and receiver behave honest and the protocol ends successfully.

2.3.1.2 Fairness

The most important property a non-repudiation protocol must satisfy is *fairness*. A protocol is said to satisfy *fairness* if at the end of the protocol execution, either each party gets the expected items



(a) Inline TTP

(b) Online TTP

(c) Offline TTP

Figure 2.1: TTP Involvement

(message, EOO, EOR), or none of them gets a valuable information. This definition of fairness is also called *strong fairness*. A protocol is said to satisfy *weak fairness* if at the end of protocol execution, either each part gets the expected items, or if one party does not get the items it expects, he/she can prove this to an adjudicator.

2.3.1.3 Transparency / Verifiability

TTP's can also be classified according to its presence in the generated evidences. A *transparent* TTP is inconceivable, whether it is involved in the protocol, otherwise the TTP is called *verifiable*, meaning that the TTP has actively involved in the evidence generation. These two properties cannot be satisfied simultaneously. Therefore, a TTP is either *transparent* or *verifiable*.

2.3.1.4 Timeliness

Timeliness propertry is closely related to *fairness*, and also a mandatory requirement for a protocol. A protocol satisfies *timeliness* if a party can choose to quit the protocol at any step without losing fairness.

2.3.1.5 State Storage

TTP's can further be classified according to how long they need to store protocol data in order to respond to requests. A TTP is called *stateless*, if either it does not have to store any data, or the data it stores can be deleted in a finite and known amount of time in order to process incoming requests. A TTP is called *stateful* if either it has to store protocol data forever, or this data can be deleted in a finite and unknown amount of time, in order to process incoming requests.

2.3.1.6 Communication Channels

The type of communication channel used between the parties can be classified into three. In an *unreliable channel*, a message sent is not guaranteed to reach its destination. In a *resilient channel*, a message is assumed to reach its destination eventually, but in a finite and unknown amount of time. In an *operational channel*, messages reach their destination in a finite and known amount of time.

2.3.1.7 Efficiency

Efficiency of a protocol is evaluated with respect to the running time of other protocols under the same computation and network resources. There are several factors affecting the efficiency of a protocol, the number of protocol steps, choice of the cryptographic algorithms and the size of the protocol messages.

2.3.1.8 Confidentiality

A non-repudiation protocol satisfies *confidentiality* if no one except sender and reciever needs to access the message content during protocol execution. This is an optional property, not all non-repudiation protocols support confidentiality.

2.3.2 Certified E-Mail

Certified E-Mail (CEM) protocols are usually first classified up to the inclusion of TTP. If there is not any TTP in the design than we call them optimistic protocols otherwise we classify up to how the TTP is involved into the protocol. Since the connection speed and computing power was poor when the first time the CEM protocols were introduced type of involvement was more important. In this respect offline is preferable to online and online is preferable to inline. Also the hardness of design of these types is listed similarly as offline, online, inline from hard to easy. Below we give examples of different CEM protocols with rescpect to TTP involvement and compare them according to the properties defined previous Section .

We now give notations and describe the cryptographic primitives used in this section.

- A and B refer to sender and receiver, respectively.
- m denotes the message.
- $Sign_X(m)$ is the pair (m, s), where s is the signature generated by secret key of X over the message m.
- $H(m)$ denotes the hash of the message, m, produced by a cryptographic hash function.
- $PK_X(m)$ denotes the asymmetric encryption of message m under the public key of X. By using public key encryption algorithm, one ensures that only X can read the message since only X has the private key.
- $Enc_k(m)$ is the symmetric encryption of message m under the secret key k.

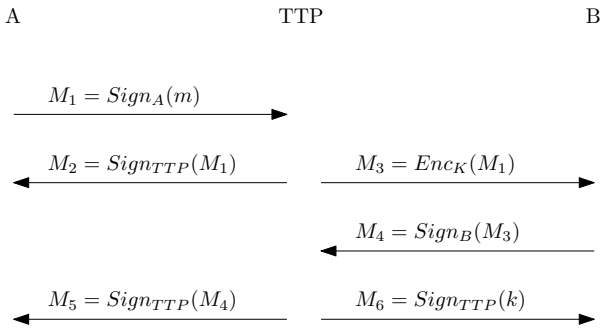


Figure 2.2: Bahreman and Tygar's protocol

2.3.2.1 CEM Protocols with Inline TTP

2.3.2.1.1 Bahreman and Tygar's CEM Protocol Bahreman and Tygar proposed the first CEM protocol with an Inline TTP [27]. This protocol is characterized by the active use of TTP, called the postmaster. The postmaster acts as an independent agent arbitrating the exchange of a receipt from the recipient, for the message and evidence of origin from the sender. Details of the protocol are demonstrated in Figure 2.2, and explained below.

1. A signs his/her e-mail message and sends the pair $M_1 = (m, s)$ to the postmaster.
2. The postmaster verifies A's signature. Then, the postmaster generates a pseudorandom number k. By using k, encryption is performed with a symmetric key encryption algorithm and the ciphertext is sent to B.
3. Immediately after B receives the ciphertext, B signs M_3 and sends it to the postmaster to declare the acception of the message.
4. In order to check the validity of M_4, the postmaster compares the original ciphertext with the one produced by applying verification operation to M_4. If the verification fails, the postmaster aborts the protocol and stops. Otherwise, postmaster signs M_4 and k, and then sends these to A and B, respectively.
5. After verifying the postmaster's signature, B uses the received key, k, to decrypt the ciphertext. If k is not received after a specified time-out period or if the received signature is invalid, B is going to repeat requesting the key from the postmaster by resubmitting receipt. A also verifies the postmaster's signature. If it is invalid or A never receives a response, she can request the postmaster to retransmit.

2.3.2.1.2 Cimato et al. CEM Protocol Cimato et al. has adopted a protocol with an Inline TTP [28], which is based on [27]. Protocol steps are given in detail in Figure 2.3.

2.3.2.2 CEM Protocols with Online TTP

The protocol proposed by Zhou and Gollmann is defined in [29] and the protocol proposed by Zhang and Shi is defined in [30]. Since these two protocols are studied in [2], we describe two other protocols

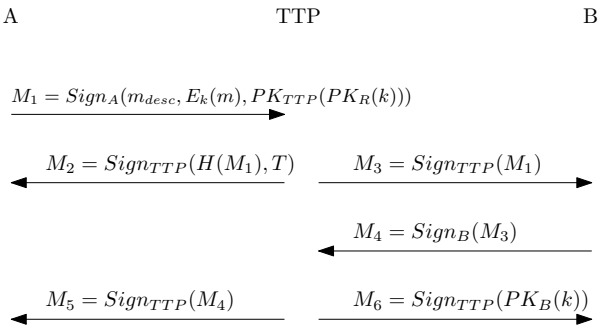


Figure 2.3: Cimato et al. protocol

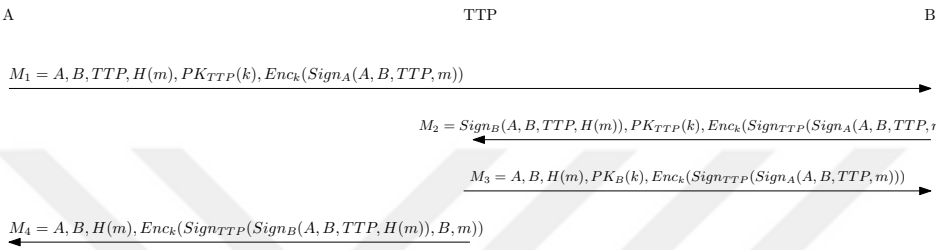


Figure 2.4: Oppliger and Stadlin's protocol

with Online TTP.

2.3.2.2.1 Oppliger and Stadlin's CEM Protocol [31] Oppliger and Stadlin proposed two protocols namely Basic Certified Mail Protocol and Stateless Certified Mail Protocol (SCMP). Here, we especially describe the SCMP in Figure 2.4 which has been analyzed by Shao, Wang and Zhou in [32]. They mounted a replay attack on the protocol, and suggested an improved version.

Originator starts the protocol by sending two messages, one to receipent one to TTP. It is up to the recipient to request the message key from TTP. It, in turn, delivers the message key to the recipient and the receipt to the sender in one atomic action. The conceptual phases of the protocol is explained below.

1. A composes a message that includes identities (A, B and TTP), the encrypted message m, encrypted k with public key of TTP and an encryption of signed identities and m with k. Simultaneously, A sends the message at Step 1 to TTP.

2. After receiving the message, B computes hash value of m and signs it with the identities. Then, B encrypts k with the public key of TTP. B sends the computed values together with the encrypted signature of identities and m with k.

3. TTP decrypts the received message. After signing this message, TTP encrypts it with k. Then, TTP encrypts k with the public key of B. TTP sends these to B. Similarly, TTP signs the received message with identity B and m. Then, TTP encrypts it with k. TTP sends these to A.

2.3.2.2.2 Deng et al. CEM Protocol [33] Deng et al. proposed two Certified Mail Protocols (CMP) which are optimal in the number of protocol steps and both meet the requirements of non-repudiation of

origin, non-repudiation of delivery, and fairness. They differ only for confidentiality, in which, latter satisfies (but not to the TTP) at the cost of performing some additional encryption and decryption operations.

2.3.2.3 CEM Protocols with Offline TTP

2.3.2.3.1 Asokan, Schunter and Waidner's CEM Protocol [34] Asokan, Schunter and Waidner proposed a generic fair exchange protocol for *general purpose*, which does not include a TTP. This protocol was the first *generic purpose* protocol since different types of items (certified mail, contract signing or payment with receipt) can be exchanged.

2.3.2.3.2 Bao, Deng and Mao's CEM Protocol [35] The previously presented offline protocol and Zhou and Gollmann's protocol [29] achieve true fairness conditionally when the TTP involves the protocol. Bao, Deng and Mao proposed three protocols which achieve true fairness. The first protocol is a kind of contract signing which shares signatures on the same file. The second one is for fair exchange in business applications which shares signatures on different files. The last one is for confidential certified email. Details of the protocol can be found in [35].

2.3.2.3.3 VRES Protocol[66] In [66] a fair exchange protocol for e-goods deliveries using verifiable and recoverable signature encryptions (VRES) depending on RSA is presented.

Main:

Step 1 $Alice \xrightarrow{<E_{k_a}(D_a), CertD_a, E_{sk_a}(h(D_a))>} Bob$

Step 2 $Bob \xrightarrow{<(x_b, xx_b, y_b), S_b, C_{bt}>} Alice$

Step 3 $Alice \xrightarrow{<k_a>} Bob$

Step 4 $Bob \xrightarrow{<r_b>} Alice$

Notation:

1. $E_{k_a}(D_a)$ symmetric encryption of file D with symmetric key k_a chosen by Alice.
2. $CertD_a$ Certificate for D_a generated by CA which is composed of $CertD_a =< Desc_a, h(E_{k_a}(D_a)), h(D_a), h(k_a) >$.
3. $E_{sk_a}(h(D_a))$ Signature of Alice on file D.
4. (x_b, xx_b, y_b) VRES of Bob for file D where;

- $x_b = (r_b x (h(D_a))^{d_b}) mod n_b$ is encryption of $rec_b = (h(D_a))^{d_b}$ which is the signature of Bob on D where r_b is a random prime generated by Bob.
- $xx_b = (r_b x E_{skbt}(h(y_b)) mod n_{bt}$ is a control number which confirms the correct use of r_b.
- $y_b = (r_b)^{e_b} mod(n_b x n_{bt})$ is encryption of r_b by Bob's public key which is recoverable by TTP.

5. $S_b = E_{sk_b}(h(C_{bt}, y_b, h(k_a), Alice))$ which is Bob's recovary authorization token.
6. C_{bt} = Certificate of Bob issued by TTP where public key is $pk_{bt} = (e_{bt} = e_b, n_{bt})$ and private key is $sk_{bt} = (d_{bt}, n_{bt})$.

Dispute Resolution:

Step 1 $Alice \xrightarrow{<y_b, S_b, C_{bt}, k_a>} TTP$

Step 2 $TTP \xrightarrow{r_b} Alice$

Step 3 $TTP \xrightarrow{<k_a>} Bob$

VRES is based on the thorem of cross-decryption; for two RSA based cryptosystems with same public exponents ($e_1 = e_2 = e$), relatively prime modulis ($n_1 and n_2$) and two messages $m_1 and m_2$ following holds;
$(m_1^e mod(n_1 x n_2) mod n_1) = m_2^e mod(n_1)$ iff $m_1 = m_2$
$(m_1^e mod(n_1 x n_2) mod n_2) = m_2^e mod(n_2)$ iff $m_1 = m_2$
By using this theorem, either of the private keys of Bob (d_1) or TTP (d_2) is used to decrypt (m^e $mod(n_1 x n_2)$) and so VRES is realized.

Table 2.1 summarizes the fulfilment of the properties defined in Section 2 for CEM protocols explained in this section.

Table2.1: Comparison of the properties of CEM protocols

Property	[27]	[38]	[28]	[36]	[31]	[33]	[34]	[35]	[66]
TTP	Inline	Inline	Inline	Inline	Online	Online	Offline	Offline	Offline
Fairness	✓	✓	✓	✓	✓	✓	✓	✓	✓
Confidentiality		✓	✓	✓	✓				✓
Timeliness	✓		✓		✓		✓		✓
Transparent							✓	✓	✓
Verifiable	✓	✓	✓	✓	✓	✓			
Stateful	✓	✓	✓	✓	✓	✓			
Stateless							✓	✓	✓

2.3.3 Contract Signing

Electronic contract signing eases the problem of paper contract signing procedure of participants who need to be at the same place and time. Nevertheless e-contract signing brings the problem of exchanging the signatures of participants in a fair way. Especially one participant may send his/her signature but may not get the respective signature of the other participant. Thus we need the electronic contract signing protocols which guarantee non-repudiation and fairness.

Another property for contract signing protocols is abuse-freeness which is defined as; no participant should be able to prove that it can uniliterally determine the outcome of the protocol. After introducing VES, the abuse-freeness for a VES is described in [118].

2.3.3.1 Ateniese Protocol

Ateniese [37] has proposed a fair exchange of digital signatures with verifiable encryption. Although he proposed an off-line protocol there is an initialization phase which is done once for certifying the public key of the protocol initiator by TTP. This is an initial version of verifiably encrypted signatures but since initialization phase is done for once not specific for each contract, it is not equivalent to verifiably encrypted signature.

Initialization:

Step 1 $Alice \xrightarrow{(e,n),CERT_A} TTP$

TTP first verifies $CERT_A$ and generates a random x.

Step 2 $TTP \xrightarrow{CERT_{TTP,A},=S_TTP(g,y=g^x,Alice,(e,n))} Alice$

General Protocol:

Step 1 $Alice \xrightarrow{P_{TTP}(S_{Alice}(m)),V_{Evidence}} Bob$

Where $S_{Alice}(m)$ is the signature of Alice of message m, P_{TTP} is the encryption with TTP's public key, V is the evidence showing that she has correctly encrypted her signature on m, means that she has made a TTP verifiable encryption of signature .

Step 2 Bob first verifies $V_{Evidence}$ if valid than

$Bob \xrightarrow{S_{Bob}(m)} Alice$

Step 3 Alice first verifies Bob's signature if valid than

$$Alice \xrightarrow{S_{Alice}(m)} Bob$$

Adj If Alice signature is invalid or if Bob does not receive anything he sends

$$Bob \xrightarrow{P_{TTP}(S_{Alice}(m)),S_{Bob}(m)} TTP$$, if TTP verifies all, then

$$TTP \xrightarrow{P_{TTP}(S_{Alice}(m))} Bob$$ and

$$TTP \xrightarrow{S_{Bob}(m)} Alice$$

Protocol with RSA:
Protocol's first step which is a verifiable encryption of an RSA signature and its verification is given as follows;
Alice sends to Bob $K_1 = H(m)^d y^r, K_2 = g^r, EQDLOG(m, y^{er}, g^r, y^e, g), CERT_{TTP,A}$, than Bob verifies $CERT_{TTP,A}$ and computes $W = K_1^e H(m)^{-1} mod n$ verify $EQDLOG(m, W, K_2, y^e, g)$ and check whether $c = c'$ where;
$(c, s) = EQDLOG(m, y^{er}, g^r, y^e, g)$,
$c = H(m||y^{er}||g^r||y^e||g||(y^e)^t||g^t)$,
$s = t - cr$
$c' = H(m||W||K_2||y^e||g||(y^e)^s W^c||g^s K_2^c)$

2.3.3.2 Garay-Jakobsson-McKenzie Protocol

Garay et al [69] have proposed an abuse-free optimistic contract signing protocol with using private contract signatures (PCS). PCS is also a similar technique to verifiable encrypted signatures with properties; a) Unforgeability: $PCS_A(m, B, T)$ can be created by A b) Designated Verifier: There is an algorithm $FakeSign_B(m, A, T)$ which can also be verifiable same as $PCS_A(m, B, T)$. c) Convertibility: $PCS_A(m, B, T)$ can be converted into a standard signature only by A or T. PCS needs non-interactive zero knowledge schemes like Schnorr signatures with following proof of statement;

" X is a TTP-encryption of 1 AND I can sign m as Alice
OR
X is a TTP-encryption of 2 AND I can sign m as Bob "

where X is the encryption of a message (1 or 2) with TTP's public key. The protocol consists of Main, Abort and Resolve parts:

Main:

Step 1 $$Alice \xrightarrow{PCS_A(m,B,T)} Bob$$
Bob checks the PCS, if not valid quits. Else goes step two.

Step 2 $Bob \xrightarrow{PCS_B(m,A,T)} Alice$
Alice checks the PCS, if not valid aborts. Else goes step three.

Step 3 $Alice \xrightarrow{S-Sig_A(m)} Bob$
Bob checks the signature, if not valid resolves. Else goes step four.

Step 4 $Bob \xrightarrow{S-Sig_B(m)} Alice$
Alice checks the signature, if not valid resolves. Else protocol finishes.

Here $S - Sig_A(m)$ denotes the universally verifiable signature converted from $PCS_A(m, B, T)$ and $S - Sig_B(m)$ denotes the universally verifiable signature converted from $PCS_B(m, A, T)$.

Abort:
Alice sends $< m, A, B, abort >_A$ to TTP. TTP checks if the signature is correct and neither Alice nor Bob have resolved TTP sends $<< m, A, B, abort >_A >_{TTP}$ back to Alice. If either Alice or Bob have resolved, he sends the stored value of resolve sub-protocol.

Resolve:
To start resolve sub-protocol for Bob, he first converts $PCS_B(m, A, T)$ to $S - Sig_B(m)$ and sends both to TTP. TTP acts as;

Step 1 If Alice has aborted the protocol previously, he sends $<< m, A, B, abort >_A >_{TTP}$ to Bob.

Step 2 If Alice has resolved the protocol previously, he sends $S - Sig_A(m)$ to Bob.

Step 3 Otherwise he sends conversion of $PCS_A(m, B, T)$ to Bob.

2.3.4 Fair Exchange

In 06KupcuLysanskaya they worked on Distributed Arbiters Fair Exchange (DAFE) protocols in which multiple autonomous (who communicate with exchangers not with themselves) multiple arbiters (TTP in fact). And they showed DAFE protocols without timeout mechanisms (which also requires syncronization) cannot provide fairness realistically.

2.3.4.1 Kupcu Protocol

In [67] a fair exchange protocol for bartering digital files, or receiving a payment in return to a file. In their protocol at first Alice sends a verifiable escrow (e-cash) to Bob and then they exchange encrypted files. Here escrow means the message which is encrypted by the public key of the TTP, and verifiable escrow means that the recepient can verify the ciphertext contains the exppected content by satisfying some relations. Than, Alice sends escrow of her key with her signature on the escrow to Bob. Afterward Bob sends the key for his file to Alice. Finally Alice sends the key for her file to Bob. The steps are summerized below;

Main:

Step 1 $Alice \xrightarrow{<e-coin, VE_{TTP}(Endorsement, Pk_A)>} Bob$

Alice sends a fresh e-coin with a verifiable escrow $VE_{TTP}(Endorsement, Pk_A)$ of the endorsement labeled with the public key. This step is done for once for different barters or exchanges.

Step 2 $Alice \xrightarrow{c_A = Enc_{K_A}(f_A)} Bob$

Alice sends encryption of her file f_A with a fresh syymetric key K_A.

Step 3 $Bob \xrightarrow{c_B = Enc_{K_B}(f_B)} Alice$

Bob sends encryption of his file f_B with a fresh syymetric key K_B.

Step 4 $Alice \xrightarrow{<Escrow = E_{TTP}(K_A; h_{f_A}, h_{f_B}, h_{c_A}, h_{c_B}, time), sign_{SK_A}(Escrow)>} Bob$

The escrow contains the key of Alice and labeled with four hash values and time.If any of the hashes, time or s,gnature is invalid, Bob aborts the protocol.

Step 5 $Bob \xrightarrow{K_B} Alice$

Bob sends the key to Alice, then Alice checks the key if it opens c_B correctly, else she runs the resolve sub-protocol.

Step 6 $Alice \xrightarrow{K_A} Bob$

When Bob receives the key from Alice, he checks the key if it opens c_A correctly, else he runs the resolve sub-protocol.

Resolve:

Bob sends the TTP, escrow and the signature received from Alice in step 4 and verifiable escrow received from Alice in step 1. The TTP checks the signature and labels, if all are valid he asks Bob for his key K_B to verify basically if it decrypts a ciphertext with hash h_{c_B} to a plaintext with hash h_{f_B}. If the key is coorect, TTP decrypts the escrow sends the key K_A to Bob. Bob controls if the key decrypts file f_A coorectly, else he proves this situation to TTP by using a special mechanism shown in Appendix B of [67] and requests from TTP the endorsement in the verifiable escrow. The resolve case for Alice is simple, she asks for the key K_B, if TTP has the key he sends it to Alice, else Alice consider the trade is aborted.

2.3.4.2 Alaraj Protocol

In [68] a fair exchange protocol for exchange of files with similar technique of [66] verifiable and recoverable encryptions is proposed.

Main:

Step 1 $Alice \xrightarrow{<Enc.k_a(F_a), C_{at}, Enc.pk_b(X_a, Z_a), Y_a, S_a, Enc.pk_b(k_b)>} Bob$

Where $C_{at} =< Alice, pk_{at}, W_{at}, Sig_t >$ is a RSA based certificate and $W_{at} = (h(sk_t + pk_{at})^{-1}.d_{at})$

Step 2 $Bob \xrightarrow{<Enc.k_b(F_b)>} Alice$

Step 3 $Alice \xrightarrow{<r_a>} Bob$

Notation:

1. $< pk_x, sk_x >$ public and private key pair for user X.
2. k_x symmetric key for encrypting and decrypting a file.
3. $Enc.pk_x(M)$ message M is asymmetricaly encrypted with public key pk_x of user X.
4. $Enc.sk_x(M)$ decryption of message M with private key pk_x of user X.
5. $Enc.k_x(M)$ message M is symmetrically encrypted by key k_x chosen by user X.
6. F_x file of user X.
7. $Sig_x(M)$ Signature on message M by user X.
8. $h(M)$ One-way hash function applied to message M.
9. heF_x keyed hash of encrypted F_x by k_x.

The key point in the protocol is the use of verifiable and recoverable encryption of Alice's symmetric key k_a which is used to encrypt F_a. This allows Bob to verify if it is correct and TTP will be able to recover it if Alice misbehaves. To compute verifiable and recoverable encryption Alice chooses a large prime r_a relatively prime to modules n_a of her public key $pk_a =< e_a, n_a >$ and continues as follows; Computes $X_a = r_a * k_a, Y_a = r_a^{e_a} mod(n_a * n_{at})$ where $< e_a, n_a >$ is the public key of user Alice and $< e_{at} = e_a, n_{at} >$ is the public key for Alice produced by TTP. Also computes $Z_a = k_a^{e_a} mod(n_a * n_{at})$. So, X_a, Y_a, Z_a together composes verifiable and recoverable encryption of Alice's key k_a. Notice here that Y_a can be decrypted by either sk_a or sk_{at} known by TTP.
Verifiable property is to be ensured by $S_a = sk_a(h(C_{at}, Y_a, Y_b, Alice))$ where $Y_b = h(Enc.k_b(F_b))$ provided by Bob to Alice before protocol run. However this assumption makes the protocol runs in fact in 4 steps instead of 3 steps. Besides that there is one more assumption that $Enc.pk_a(k_a)$ is known by Bob before the protocol and that makes an extra step also.

Dispute:

Step 1 $Bob \xrightarrow{<Enc.k_b(F_b),C_{at},Y_a,S_a>} TTP$

Step 2 $TTP \xrightarrow{<Enc.k_b(F_b)>} Alice$

Step 3 $TTP \xrightarrow{<r_a>} Bob$

2.4 Implementation and Emerging Areas

2.4.1 Emerging Areas

There are various areas like cloud computing, healthcare systems, wireless sensor networks etc. where pairings are used to meet specific requirements. In cloud computing and data storage, mainly encryption techniques like searchable, homomorphic or attribute based encryption have been studied. To assure both privacy and authenticity in a cloud environment Attribute Based Encryption (ABE) is preferable. For example medical and health data should be kept private to public access but also be available to authorized personnel. In the case of the authorization of hospital personnel with respect to patients illness only personnel with related properties (like doctors in dermatology and nurses in emergency service) can reach the related data, attribute based encryption provides the required flexiblity and security. In [100] [102] [101] ABE is used with this provision. ABE with different variations is also used in cloud computing, in [103] hierarchical attribute-set-based encryption (HASBE) has been proposed which is an hierarchical extension of attribute-set-based encryption which provides multiple-levels of attribute authorities.

Besides this broadcast encryption has application areas like radio systems (Over The Air Re-keying), military broadcast systems (submarine communication), Pay-TV. Broadcast encryption is a scheme that enables a broadcaster to encrypt a message for a large set of users (all subscribers of a paid TV for example) but only a qualified subset (subscribers who have paid their bill for example) of that large set can decrypt the message. The first broadcast encryption scheme with pairings is proposed in [91] with supersingular curves and has been improved in [92] with using ordinary Barreto-Naehrig curves in asymetric pairing type.

For non-repudiation, anonymity, authentication with attributes or identity or membership different signature types have been proposed to be used in cloud, WSN, e-voting, e-commerce etc. We give some state of the art information about these primitives;

Ring Signatures: Signature type in which a member of a group can sign a message without information leaking about the signer. So it reveals the predicate that the message was signed by a list of possible users. When signing public keys of all group members are used, but only single private key of the signer is used. The members of the group is not prearranged, chosen by signer ad-hoc. The first ring signature was proposed in [93] depends on RSA, but there are many ring signatures based on pairings [19], [94].

Group Signatures: Similar to ring signature but in group signature, there is a group manager who can reveal which member has signed the message in the name of group. There can be special procedure to join the group and special public/private keys for group members, so the group is prearranged. The notion of group signature goes back to 1991, Chaum and van Heyst, there are many implementations with pairings like [95], [96].

Mesh Signatures: Mesh signatures generalize the ring signature notion by allowing the combination of signatures by one or multiple signers from a larger group. And instead of generating and publishing each public key in a ring scheme, one certificate authorithy is needed by this means the signer can hide

in a crowder group who does not have a published verification key.

Attribute Base Signatures: Along with ABE Attribute Based Signature (ABS) schemes are also developed to support cloud and Internet of Things (IoT). Although ring and mesh signatures are both introduced with the purpose of leaking information anonymously, ABS are proposed to satisfy this purpose more effectively. It guarantees that a single user with proper attributes has signed the message and colluding parties can not be able to pool their attributes unlike from mesh signatures.

Blind Signatures: The message signed is blinded to the signer, but the verification can be done publicly by original message. This type of signatures are mainly used in e-voting and e-cash systems. First introduced by David Chaum in Crypto 82 but implemented by pairings [17].

Designated Verifier Signatures: In this type of signature scheme signature can only be verified by a single or a set of designated verifier chosen by signer. First proposed in EuroCrypt96 by Markus Jakobsson et al. Similar to designated verifier signature undeniable signatures are introduced which includes signer and verifier interaction. These are used in private and authenticated mail and communication systems.

2.4.2 Implementation

Pairing Based Cryptography (PBC) Libraries: There are many PBC libraries in the literature, among other implementations Ben Lynn's pairing library is the most cited, used and re-distributed one. Lynn's [105] PBC library is coded in C and based on GMP which is freely available and it is licensed under GPL. It inspired and used in other implementations with different functionalities listed below;

- **CP-ABE** Ciphertext Policy ABE [106] is the most used ABE implementation which is also based on [105].
- **QED** QED enables queries over ciphertext, especially comparison queries like subset and range queries.
- **Crypt:PBC** This is a Perl module interface for PBC library.
- **PBC_bce** Library is is an implementation of the Boneh-Gentry-Waters broadcast encryption scheme.

And some example schemes are included in the library, such as;

1. Boneh-Lynn-Shacham short signatures
2. Hess identity-based signatures
3. Joux tripartite Diffie-Hellman
4. Paterson identity-based signatures

5. Yuan-Li identity-based authenticated key agreement
6. Zhang-Kim identity-based blind/ring signatures
7. Zhang-Safavi-Naini-Susilo signatures
8. Boneh-Boyen short signatures
9. Boneh-Boyen-Shacham short group signatures
10. Cha-Cheon identity-based signatures

Some libraries like Lynn's, are purely designed for PBC, but there are also other general cryptographic libraries which cover PBC like MIRACL, Magma etc. To mention some featured ones; MIRACL (Multiprecision Integer and Rational Arithmetic Cryptographic Library) is a high performance cryptographic library which was free at the beginning but propriatary today and used commonly commercially. It's amin focus is on Elliptic Curve Cryptography but also supports pairings. Its benchmarks of pairing computation for supersingular curves is between 1.2 and 42 ms and for ordinary curves between 1.1 and 34 ms ranging from 80 to 256 bit level security.

MAGMA (Magma Computational Algebra System) is developed for algebraic calculations. It also supports pairing computations.

Apart from general purpose pairing libraries there are also specific purpose libraries like TinyPBC for limited sources like sensor networks, TEPLA and RELIC with limited implementation. TEPLA (University of Tsukuba Elliptic Curve and Pairing Library) is another C and GMP based library for elliptic curve and pairings. But it only supports calculation on Barreto-Naehrig (BN) curve and Optimal Ate pairing on BN curve. It has a benchmark of 6.4 ms on BN254 curve.

Link to all Pairing Based Cryptography libraries can be found in the web site [117]

Java: As a platform independant language Java provides using applications on Java Runtime Environment installed machines without recompiling. There are several Java libraries for PBC Java1, Java2, JavaMobil, jPBC. Among them jPBC argues that it is the only library which is full and free implementation of PBC on Java. jPBC does indeed provides a port to PBC library of Lynn. They give in [107] benchmarks for Android mobile platform also.

Mobile: In [92] an implementation result of broadcast encryption on smartphones with a proprietary industrial library but did not give results for pairing computation.

In [111] Malina et al have implemented again group signature scheme of [96] with MNT curves of Type-D (175 bit prime base field) with security level of 128 bit. They get result on three android platforms as approximately 3500, 3000 and 2400 ms for pairing operation. The first two observations are on older runtime (Dalvik) of Android whereas last one is on new runtime (ART) of Android 4.4 which brings about 20%. They again used the jPBC for implementation but differing from [107], they performed asymmetric pairing operations on MNT curves. The results for Type-A 512 bit curves are similar to results of [107] about 500 ms without precomputation. For Type-D curves of base field sizes of 159, 201 and 204 bit length, best results with new runtime 3000, 4000 and 5800 ms, respectively.

In [113] they have implemented group signature scheme of [96] with Barreto-Naehrig curves with prime base field of length 256 and embedding degree k=12. They observe an ATE pairing time of 63 ms with some optimization methods such as use of Jacobian coordinates and joint point-and-line computation and overall group signature. They also presented different optimization methods for implementing group signature scheme such as Shamir trick and observe 157 ms for whole scheme. By using other optimization techniques as exchanging bilinear pairing with exponentiation and delegating pairings they observe about 95 ms of signature implementation.

For Android devices Liu et al [114] developed a new library for PBC. Their advantage is unlike jPBC in which whole PBC library have been wrapped into Java, they have wrapped only algorithms for cryptosystems. This difference makes developing codes in native C not in Java and alleviates extra costs caused by input/output between C and Java. They watch the steps as; to port GMP into Android used prebuilt GMP 5 for Android , modified and ported PBC. At the end to implement BLS scheme for example one does not need to write BLS in Java instead makes minor changes in BLS C code. To talk about benchmarks; a pairing runs about 27 ms, BLS signing in about 48 ms and BLS verificationin 33 ms.

In [116] they argue that mPBC library is used to implement the schemes of MobInfoSec project for mobile devices . Unfortunately there is not any more information about implementation, detail, code or benchmark. But they advise to use cloud based techniques to implement pairing based cryptography as they did with DropBox but again there is not any information about how they did it.

In [115] another Java based PBC library mPBC 's developed. When we look at their benchmarks; the pre-processed Tate pairing operation computes in 426 ms in DVM (Samsung GT-N7000) and 4.5 ms in JVM (Sager NP5160). Which also shows the incoherency on running times of Java-based PBC libraries between Java Virtual Machine (JVM) and Dalvik Virtual Machine (DVM). They assert that mPBC library outperforms the existing Java-based PBC libraries in DVM, and as the fastest PBC library to date in the JVM.

CHAPTER 3

IMPROVED CONTRACT SIGNING PROTOCOL BASED ON CERTIFICATELESS HYBRID VERIFIABLY ENCRYPTED SIGNATURE SCHEME

3.1 Introduction

Certificateless Public Key Cryptography (CL-PKC) has been arisen to mitigate the difficulties of PKI and Identity Based Cryptography (IBC). Briefly; in conventional PKI certificates has a role to bind public keys with idendities by signatures of certificate authorities. But managing these certificates involves some proceses as; issuing, storing, transfering, verifying and revocating. CL-PKC has an advantage especially in transfer and verification processes since there is not actually a certificate to transfer and verify, but only public keys are needed and shared. In IBC the Private Key Generator (PKG), Key Generation Center (KGC) or Trusted Third Party (TTP), whatever you call, has the capability of key escrowing, since he/she generates the private key of users depending on identities. In CL-PKC however, PKG/KGC only generates partial private key and user also generates other partial private key which eliminates the key escrow capability of PKG/KGC.

In this chapter, we first propose an adaptation of certificateless public key cryptography to hybrid verifiably encrypted signature scheme [45] which we call CL-HVESS. Then we expand CL-HVESS to Type-III pairings to mitigate the risks of recent attacks on Type-I pairings. Finally in Section 3.4 we present a replay attack to Chen and Gu protocol [45] and propose an improvement which is resistant to replay attacks. We also extend this improvement to our CL-HVESS protocol.

As a kind of non-repudiation protocol, contract signing share similar properties with other protocols. The goal of contract signing protocols is exchange of evidence of non-repudiation not the message itself. Differing from certified e-mail or fair exchange is that obtaining message content is not important but exchanging signed message/contract fairly is the main goal of the contract signing protocol.

3.2 An Adaptation of Certificateless Public Key Cryptography to HVESS

ID-Based signature and encryption schemes use publicly known variables such as identity or e-mail of a user to derive public key without any key distribution for public keys. For signing and decrypting, user contacts to a Private Key Generator (PKG, CA etc.) to derive the private key which is dependent on the identity and master key of the PKG.

This scheme has some disadvantages as stated in [4]

- The PKG can calculate users private keys which is a problem for both non-repudiation and confidentiality in security protocols
- User has to authenticate himself to PKG
- PKG needs a secure channel to send users private key
- User has to publish PKG's public parameters

Chen and Gu have developed and used HVESS [45] which is subject to problems stated above. To eliminate some of the above mentioned disadvantages, we adapted Riyami and Paterson's [43] certificateless public cryptography scheme to HVESS and call the adapted scheme shortly as CL-HVESS. Most of the parts of our scheme is similar to the original one [45] naturally.

Setup : Let $\mathbb{G}_1$ be additive group of prime order q and $\mathbb{G}_3$ be multiplicative group of prime order q. Choose an arbitrary generator $P \in \mathbb{G}_1$, a random secret PKG master key $s \in \mathbb{Z}_q^*$ and a random secret adjudicator master key $s_T \in \mathbb{Z}_q^*$. Set $P_{pub} = [s]P$ and $P_{adj} = [s_T]P$. $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$ and $H_2 : \{0,1\}^*$ X $\mathbb{G}_1 \rightarrow \mathbb{Z}_q^*$ are hash functions. Publish $(\mathbb{G}_1, \mathbb{G}_3, q, \hat{e}, P, P_{pub}, P_{adj}, H_1, H_2)$.

Extract : Public and private key pair for user ID is computed as follows:

- TTP or PKG computes $P_{pub} = [s]P$ and $[s]H_1(ID)$ as the partial private key then send to user ID.
- User ID computes $P_{pub\_ID} = [X_{ID}][s]P$ and $R_{pub\_ID} = [X_{ID}]P$ as public keys then computes $d_{ID} = [X_{ID}][s]H_1(ID)$ as private key.

Sign : Given a private key d_{ID} and a message m, pick a random $r \in \mathbb{Z}_q^*$, compute $U = [r]P, h = H_2(m, U), V = [r]H_1(ID) + [h]d_{ID}$ and output a signature (U, V).

Verify : Given a signature (U, V), of an identity ID and public keys P_{pub_ID}, R_{pub_ID} first check certificateless public keys as $\hat{e}(R_{pub\_ID}, P_pub) \stackrel{?}{=} \hat{e}(P_{pub\_ID}, P)$ then compute $h = H_2(m, U)$, and accept the signature and return 1 if and only if $\hat{e}(P, V) = \hat{e}(U + [h]P_pub\_ID, H_1(ID)$. The proof of verification for a valid signature (U, V) is as follows;
$\hat{e}(P, V) = \hat{e}(P, [r]H_1(ID) + [h]d_{ID})$
$= \hat{e}(P, [r]H_1(ID) + [h][X_{ID}][s]H_1(ID))$
$= \hat{e}(P, ([r] + [h][X_{ID}][s])H_1(ID))$
$= \hat{e}(([r] + [h][X_{ID}][s])P, H_1(ID))$
$= \hat{e}([r]P + [h][X_{ID}][s]P, H_1(ID))$
$= \hat{e}(U + [h]P_{pub\_ID}, H_1(ID))$

SignVE : Given a private key d_{ID} and a message m, pick randomly $r_1, r_2 \in \mathbb{Z}_q^*$, compute $U_1 = [r_1]P, U_2 = [r_2]P, h = H_2(m, U_1), V = [r_1]H_1(ID) + [h]d_{ID} + [r_2]P_{adj}$, and output a verifiably encrypted signature (U_1, U_2, V)

VerifyVE : Given a verifiably encrypted signature (U_1, U_2, V) of a user ID for a message m, compute $h = H_2(m, U_1)$, accept the signature if and only if $\hat{e}(P, V) = \hat{e}(U_1 + [h]P_pub\_ID, H_1(ID)$. $\hat{e}(U_2, P_{adj})$. The proof of verification for a valid verfiably encrypted signature (U_1, U_2, V) is as follows;

$\hat{e}(P,V) = \hat{e}(P,[r_1]H_1(ID) + [h]d_{ID} + [r_2]P_{adj})$
$= \hat{e}(P,[r_1]H_1(ID) + [h][X_{ID}][s]H_1(ID)).\hat{e}(P,[r_2][s_T]P)$
$= \hat{e}(P,([r_1] + [h][X_{ID}][s])H_1(ID)).\hat{e}(U_2,P_{adj})$
$= \hat{e}(([r_1] + [h][X_{ID}][s])P,H_1(ID)).\hat{e}(U_2,P_{adj})$
$= \hat{e}([r_1]P + [h][X_{ID}][s]P,H_1(ID)).\hat{e}(U_2,P_{adj})$
$= \hat{e}(U_1 + [h]P_pub\_ID,H_1(ID)).\hat{e}(U_2,P_{adj})$

Adjudication : Adjudicator can calculate $V_1 = V - [s_T]U_2$ from (U_1,U_2,V) by using his/her private key after validating. Validation requires first verification of verifiably encrypted signature (U_1,U_2,V) and then verification of adjudicated verifiably encrypted signature (U_1,V_1) as an original signature. First part is same procedure as $VerifyVE(U_1,U_2,V)$, for the validation of second part: $V_1 = V - [s_T]U_2 = V - [s_T][r_2]P = V - [r_2]P_{adj} = [r_1]H_1(ID) + [h]d_{ID} + [r_2]P_{adj} - [r_2]P_{adj}$ $= [r_1]H_1(ID) + [h]d_{ID}$ so $\hat{e}(P,V_1) = \hat{e}(P,[r_1]H_1(ID) + [h]d_{ID}) = \hat{e}(U_1 + [h]P_pub\_ID,H_1(ID)$

3.3 Expansion of CL-HVESS to Type-III Pairings

In the previous section we have adapted Certificateless PKC to HVESS on Type-I pairings in which $\mathbb{G}_1 = \mathbb{G}_2$. Since Type-I pairings are susceptible to recent quasi-polynomial attacks [49], [50], here we expanded CL-HVESS to Type-III pairings. Type-II pairings are not suitable for CL-HVESS because there is not efficiently computable hash function to $\mathbb{G}_2$.

Setup : Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be additive abelian group of order q and $\mathbb{G}_3$ be multiplicative group of order q. Choose arbitrary generators $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$ a random secret PKG master key $s \in \mathbb{Z}_q^*$ and a random secret adjudicator master key $s_T \in \mathbb{Z}_q^*$. Set $P_{pub} = [s]P$, $Q_{pub} = [s]Q$, $P_{adj} = [s_T]P$ and $Q_{adj} = [s_T]Q$ choose cryptographic hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$, $H_2 : \{0,1\}^* \rightarrow \mathbb{G}_2$ and $H_3 : \{0,1\}^*$ X $\mathbb{G}_1 \rightarrow \mathbb{Z}_q^*$. Publish the system parameters $(\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_3,q,e,P,Q,P_{pub},Q_{pub},P_{adj},Q_{adj},$ $H_1,H_2,H_3)$

Extract : Public and private key pair for user ID is computed as follows:

- TTP or PKG computes $P_{pub} = [s]P, Q_{pub} = [s]Q$ and $[s]H_1(ID), [s]H_2(ID)$ as the partial private keys then send to user ID.
- User ID computes $P_{pub\_ID} = [X_{ID}][s]P, Q_{pub\_ID} = [X_{ID}][s]Q$ and $R\_P_{pub\_ID} = [X_{ID}]P$, $R\_Q_{pub\_ID} = [X_{ID}]Q$ as public keys then computes $d\_P_{ID} = [X_{ID}][s]H_1(ID), d\_Q_{ID} = [X_{ID}][s]H_2(ID)$ as private keys.

Sign : Given a private key d_Q_{ID} and a message m, pick a random $r \in \mathbb{Z}_q^*$, compute $U = [r]P, h = H_3(m,U), V = [r]H_2(ID) + [h]d\_Q_{ID}$ and output a signature (U,V).

Verify : Given a signature (U,V), of an identity ID and public keys $P_{pub\_ID}, R\_P_{pub\_ID}$ first check certificateless public keys as $\hat{e}(R\_P_{pub\_ID},Q_pub) \stackrel{?}{=} \hat{e}(P_{pub\_ID},Q)$ then compute $h = H_3(m,U)$, and accept the signature and return 1 if and only if $\hat{e}(P,V) = \hat{e}(U+[h]P_pub\_ID,H_2(ID)$. The proof of verification for a valid signature (U,V) is as follows;
$\hat{e}(P,V) = \hat{e}(P,[r]H_2(ID) + [h]d\_Q_{ID})$
$= \hat{e}(P,[r]H_2(ID) + [h][X_{ID}][s]H_2(ID))$

$= \hat{e}(P, ([r] + [h][X_{ID}][s])H_2(ID))$
$= \hat{e}(([r] + [h][X_{ID}][s])P, H_2(ID))$
$= \hat{e}([r]P + [h][X_{ID}][s]P, H_2(ID))$
$= \hat{e}(U + [h]P_{pub\_ID}, H_2(ID))$

SignVE : Given a private key d_Q_{ID} and a message m, pick randomly $r_1, r_2 \in \mathbb{Z}_q^*$, compute $U_1 = [r_1]P, U_2 = [r_2]Q, h = H_3(m, U_1), V = [r_1]H_2(ID) + [h]d\_Q_{ID} + [r_2]Q_{adj}$, and output a verifiably encrypted signature (U_1, U_2, V)

VerifyVE : A user acknowledges (U_1, U_2, V) as a verifiably encrypted signature if and only if $\hat{e}(P, V) = \hat{e}(U_1 + [h]P_pub\_ID, H_2(ID).\ \hat{e}(P_{adj}, U_2)$. The proof of verification for a valid verfiably encrypted signature (U_1, U_2, V) is as follows;
$\hat{e}(P, V) = \hat{e}(P, [r_1]H_2(ID) + [h]d\_Q_{ID} + [r_2]Q_{adj})$
$= \hat{e}(P, [r_1]H_2(ID) + [h][X_{ID}][s]H_2(ID)).\hat{e}(P, [r_2][s_T]Q)$
$= \hat{e}(P, ([r_1] + [h][X_{ID}][s])H_2(ID)).\hat{e}([s_T]P, [r_2]Q)$
$= \hat{e}(([r_1] + [h][X_{ID}][s])P, H_2(ID)).\hat{e}(P_{adj}, U_2)$
$= \hat{e}([r_1]P + [h][X_{ID}][s]P, H_2(ID)).\hat{e}(P_{adj}, U_2)$
$= \hat{e}(U_1 + [h]P_pub\_ID, H_2(ID)).\hat{e}(P_{adj}, U_2)$

Adjudication : Adjudicator can calculate $V_1 = V - [s_T]U_2$ from (U_1, U_2, V) by using his/her private key after validating. Validation requires first verification of verifiably encrypted signature (U_1, U_2, V) and then verification of adjudicated verifiably encrypted signature (U_1, V_1) as an original signature. First part is same procedure as $VerifyVE(U_1, U_2, V)$, for the validation of second part: $V_1 = V - [s_T]U_2 = V - [s_T][r_2]Q = V - [r_2]Q_{adj} = [r_1]H_2(ID) + [h]d\_Q_{ID} + [r_2]Q_{adj} - [r_2]Q_{adj} = [r_1]H_2(ID) + [h]d\_Q_{ID}$ so $\hat{e}(P, V_1) = \hat{e}(P, [r_1]H_2(ID) + [h]d\_Q_{ID}) = \hat{e}(U_1 + [h]P_pub\_ID, H_2(ID)$

3.4 Attack and Improvement to Fair Contract Signing Protocol

3.4.1 Attack to Contract Signing Protocol

Here we show a replay attack to Chen and Gu protocol [45], in which the responder site could get the adjudicated contract but the initiator A, can not get the contract signed by the intended responder B, instead get the contract signed by a colluder C. The attack of the scenerio is figured in Fig.1 and then explained further below.

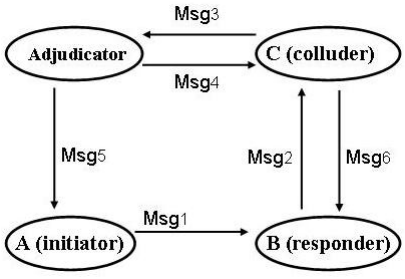


Fig 1. Replay attack on protocol

Msg 1 $A \rightarrow B : ID_A, C, SignVE\{d_A, ID_A, C, P_{Adj}\}$

Msg 2 B colludes with C and sends verifiably encrypted signed contract to C
$B \rightarrow C : ID_A, C, SignVE\{d_A, ID_A, C, P_{Adj}\}$

Msg 3 C signs the contract by his private key and request from the adjudicator to resolve the dispute.
$C \rightarrow Adj : (ID_A, C, SignVE\{d_A, ID_A, C, P_{Adj}\})$
and $Sign(d_C, ID_C, C)$

Msg 4 After the adjudicator verifies the signed and verifiably encrypted signed contract, delivers the adjudicated contract to C.
$Adj \rightarrow C : Adjudication(SignVE\{d_A, ID_A, C, P_{Adj}\})$

Msg 5 After the adjudicator verifies the signed and verifiably encrypted signed contract, delivers the signed contract to A.
$Adj \rightarrow A : Sign(d_C, ID_C, C)$

Msg 6 Colluder C returns the adjudicated contract to B.
$C \rightarrow B : Adjudication(SignVE\{d_A, ID_A, C, P_{Adj}\})$

This contract signing protocol is based on HVESS as cryptographic signature scheme, which was proven as secure in [45]. Besides the cryptographic security, information sent in the protocol / signature scheme and control checks are also very important for a security protocol. In this attack we exploited a security flaw of missing information, namely identifier of responder, in the signature scheme. It may be claimed as the contract is suited for A and B but this would not be a formal security check for a protocol.

3.4.2 Improvement to Contract Signing Protocol

The improvement to the protocol is very easy as to include the identifier of responder to the signed message and check this before any response. For adding the CL-HVESS, we include the public keys of the sender to the message. The improved protocol is shown below; note that Signed or verifiably signed messages also include the original messages.

Msg 1 $A \rightarrow B : SignVE\{d_A, ID_A, ID_B, C, P_{pub\_A},$
$R\_P_{pub\_A}, P_{Adj}, Q_{Adj}\}$

Msg 2 $B \rightarrow A : Sign\{d_B, ID_B, ID_A, C, P_{pub\_B},$
$R\_P_{pub\_B}, P_{Adj}, Q_{Adj}\}$

Msg 3 $A \rightarrow B : Sign\{d_A, ID_A, ID_B, P_{pub\_A}, R\_P_{pub\_A}, Msg2\}$

3.4.3 Analysis of Protocol

Although there is not a formal security proof for CL-HVESS, we can make an informal comparison between original protocol and our work. When you use traditional ID-Based encryption and signature methods, as done in the original scheme, TTP can generate and escrow private keys of all users. But in certificateless scheme of [43] users can generate their own private keys. Also revocating a disclosed or lost private key in pure ID-Based crypto systems is difficult because you have to change the corresponding public key and so the ID of that user depends on. Using schemes of [43] TTP can not escrow keys but can revocate keys easily which is important for contract signing protocols depending on pairings. Addition to security analysis we can say the improved protocol is resistant to replay attacks. When we compare our adapted protocols with original version in view of efficiency, there is not so much difference between them. Both Type I and Type III versions of CL-HVESS have same calculations except setup phase which is done for only once. Below is the comparison of efficiency:

- **Sign** same as original; 3 scalar multiplication.
- **Verify** extra two pairings to check certificateless public keys; in total 4 pairings, 1 scalar multiplication.
- **SignVE** same as original; 5 scalar multiplication.
- **VerifyVE** extra two pairings to check certificateless public keys; in total 5 pairings, 1 scalar multiplication.
- **Adjudication** same as original; 1 scalar multiplication.

3.5 Conclusion

We proposed adaptation of certificateless public key cryptography to hybrid verifiably encrypted signature scheme [45] which we call CL-HVESS. Adaptation of certificateless PKC prevents some problems of pure ID based schemes especially generation of user private keys by PKG. Then we expanded CL-HVESS to Type-III pairings to mitigate the risks of recent attacks on Type-I pairings. We also presented a replay attack to Chen and Gu protocol [45], in which the responder site could get the adjudicated contract but the initiator A, can not get the contract signed by the intended responder B, instead get the contract signed by a colluder C. Then we propose an improvement to the protocol which is resistant to replay attacks and also included the CL-HVESS to the improved protocol. Formal security proof of CL-HVESS remains as a future work.

CHAPTER 4

HYBRID NON-REPUDIATION PROTOCOL WITH PAIRING BASED CRYPTOGRAPHY

4.1 Introduction

Non-repudiation protocols are used for exchange of information with evidence of non-repudiation. Application of Non-repudiation protocols are spreaded over Certfied E-mail, Electronic Contract Signing, e-commerce and electronic payment.
Although there are many different types of Non-repudiation protocols such as Certified E-mail, Contract signing, fair exchange, differing in their goals; they are related with each other and share the properties Non-repudiation and fairness in common. To show these differences with an example; when non-repudiation protocol is based on message delivery like in Certified E-mail, receiver has to provide NRR in order to get the message and obtain the NRO for that message. But when non-repudiation protocol is based on exchange of evidence of non-repudiation not the message itself like in contract signing, obtaining message content is not important but exchanging signed message/contract fairly is the main goal of the application.
In Section 4.2 we first give description of our two-round protocol which is based on Joux tri-partite key exchange. Then we analyze it in Section 4.3. In Section 4.4 we restate the certificateless PKC of [43] which we used in previous section.

4.2 Protocol Definition

We present an ID-based hybrid non-repudiaiton protocol using the Joux tri-partite key exchange scheme. Our protocol is hybrid because in the first round of exchange TTP is on-line but in the next rounds with same entities TTP works off-line. TTP in the protocol also acts as PKG. If we had used traditional ID-Based encryption and signature methods, TTP can generate and escrow private keys of all users. But in certificateless scheme of [43] users can generate their own private keys. Also revocating a disclosed or lost private key in pure ID-Based crypto systems is difficult because you have to change the corresponding public key and so the ID of that user depends on. Using schemes of [43] TTP can not escrow keys but can revocate keys easily which is important for our non-repudiation protocol depending on pairings.

4.2.1 Notation

Description of notation is as follows:

- A: Sender
- B: Receiver
- TTP: Trusted Third Party
- M_i: Message labeled $i; 1 \leq i \leq 6$
- $Sig_X\{M\}$: Message M signed by agent X's private key by ID-Based Signature Scheme
- $(M)_k$: Message M symmetrically encrypted by key k
- $\{M\}_X$: Message M encrypted for agent X's public key by ID-Based Encryption Scheme
- S_id: Session identifier
- EOO: Evidence Of Origin
- EOR: Evidence Of Receipt
- EOS: Evidence Of Submission of key
- EOD: Evidence Of Delivery
- $h(M)$: Hash of message M
- M_id: Message identifier is equal to $h(h(M), S_id)$
- kek_sid: Key encryption key which is equal to $h(\hat{e}([x]P, [y]P)^z, s_id)$

4.2.2 Protocol Description

The protocol starts with an initialization and registration at the beginning.
Initialization: TTP generates *setup* phase shown in Section 4.4 and publishes system parameters $\mathbb{G}_1, G_3\ \hat{e}, P, P_{pub}, P_{pub\_ID}, H_1, H_2$. TTP generates $s \in \mathbb{Z}_q^*$ where $P_{pub} = [s]P$ and keeps secret, TTP also generates its own public key P_{pub_TTP} and corresponding private key.
Registration: A user with identitiy ID registers to the TTP. First TTP sends the partial key to user ID, then user ID computes public key $P_{pub\_ID} = [X_{ID}]P_{pub}$ where $[X_{ID}] \in \mathbb{Z}_q^*$ and sends to TTP over authentic channel. User ID computes his private key as shown in Section 4.4.
Execution: The sender A with public key P_{pub_A}, private key d_A computes $[x]P$ where $x \in \mathbb{Z}_q^*$ chosen randomly for Joux tri-partite scheme. The receiver B with public key P_{pub_B}, private key d_B computes $[y]P$ where $y \in \mathbb{Z}_q^*$ random element. TTP with public key P_{pub_TTP}, private key d_{TTP} computes $[z]P$ where $z \in \mathbb{Z}_q^*$ chosen randomly.
For the first time of exchange between the participants A, B and TTP round 1 procedure is executed, for next exchanges with the same participants round 2 procedure is executed.

4.2.2.1 Online Round

Round 1 is the online mod of the hybrid protocol.
Main protocol of Round 1, shown in Figure 1 below is as follows:

Step 1 $A \overset{EOO}{\rightarrow} B:$ $M_1 = Sig_A\{A, B, TTP, S_id, h(M), [x]P, (A, B, TTP, \{M\}_B)_k\}$

$A \overset{EOO}{\rightarrow} TTP:$ $M_2 = Sig_A\{M_1, \{k\}_{TTP}\}$

Step 2 $B \overset{EOR}{\rightarrow} A:$ $M_3 = Sig_B\{A, B, TTP, S_id, h(M), [y]P, (A, B, TTP, \{M\}_B)_k\}$

$B \overset{EOR}{\rightarrow} TTP:$ $M_4 = M_3$

Step 3 $TTP \overset{EOD}{\rightarrow} B:$ $M_5 = Sig_{TTP}\{A, B, TTP, S_id, h(M), [z]P, (k)_{kek\_sid}\}$

$TTP \overset{EOS}{\rightarrow} A:$ $M_6 = M_5 + M_4$

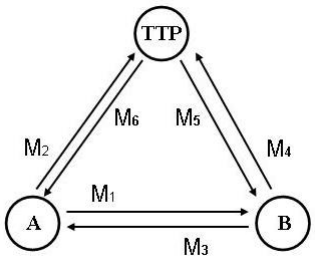


Figure 1. First Round Message Flow

Here the critical point in the protocol is the usage of signed Joux tri-partite key exchange, after the Step 3 of the Round 1, A, B and TTP has $[x]P, [y]P, [z]P$ in common. This means that they can compute $\hat{e}([y]P, [z]P)^x) = \hat{e}([x]P, [z]P)^y = \hat{e}([x]P, [y]P)^z$.
The steps defined above follow previous one after some checks, as;
In Step 2 receiver B checks the identities, signature of sender A in M_1.
In Step 3 TTP checks:

- First, the identities, session identifier and signature of sender A in message M_2.
- Secondly, checks if the key k, which was sent in Step 1 by A is working properly. TTP decrypts the encrypted part $(A, B, TTP, \{M\}_B)_k$ in message M_1 by the key k and checks the ID's are correct.

- Thirdly, checks the identities, session identifier and signature of receiver B in message M_4 which is equal to M_3.
- Finally, cross-checks the encrypted part in M_3 is same as the encrypted part in M_1.

If any of these checks fail then TTP cancels the protocol. Otherwise TTP continues to Step 3, calculates the kek_sid the key encryption key which is equal to $h(\hat{e}([x]P, [y]P)^z, s_id)$, encryptes the key k with kek_sid and sends the messages M_5 and M_6.

Cancellation Sub-protocol After Step 1 sender A can cancel the protocol by sending TTP a cancellation message. The TTP confirms the *Cancellation* request if the signature is valid and the request is coming from the sender of the message. The cancellation sub-protocol works as follows;

Step 1 $A \overset{Cancel}{\rightarrow} B, TTP: \qquad M_1' = Sig_A\{Cancel, M_2\}$

Step 2 $TTP \overset{Confirm}{\rightarrow} A, B: \qquad M_2' = Sig_{TTP}\{Cancel - Confirm, S_id, M_1', \}$

If A sends *Cancellation* request to only TTP and B sends M_3 and M_4 meanwhile, TTP gets both *Cancellation* request and M_4. TTP aborts the protocol in this case also. But any *Cancellation* request from sender after Step 3 is not accepted. *Cancellation* confirmation is not valid without M_2'. By this way A cannot repudiate M_1 and M_2.

After Step 1 before Step 2 receiver B can also cancel the protocol by sending TTP a *Cancellation* request. The TTP confirms the *Cancellation* request if the signature is valid and the request is coming from the receiver of the message. The *Cancellation* sub-protocol works as follows;

Step 1 $B \overset{Cancel}{\rightarrow} A, TTP: \qquad M_1' = Sig_B\{Cancel, M_1\}$

Step 2 $TTP \overset{Confirm}{\rightarrow} A, B: \qquad M_2' = Sig_{TTP}\{Cancel - Confirm, S_id, M_1', \}$

Any *Cancellation* request from receiver after Step 2 is not accepted.

Dispute Resolution After Step 2 if the receiver B did not get the key from TTP, recipient B can run *Resolve* sub-protocol. This is a case if the message M_3 has reached to sender a but message M_4 has not reached to TTP, because of network error or sender A blocks it as an active attack. The *Resolve* sub-protocol works as follows;

Step 1 $B \overset{Resolve}{\rightarrow} A, TTP: \qquad M_1' = Sig_B\{Resolve, M_1, M_4\}$

Step 2 $TTP \overset{Confirm}{\rightarrow} B: \quad M_2' = M_5$
$TTP \overset{Confirm}{\rightarrow} A: \quad M_3' = M_6$

Before confirmation for resolve request TTP checks the same points as done in main protocol at Step 3.

4.2.2.2 Off-line Round

Round 2 is the off-line mod of the hybrid protocol.
After **Round 1** with online TTP users can pass to Off-line TTP. A, B and TTP has $[x]P, [y]P, [z]P$. A, B and TTP have previously computed $\hat{e}([y]P, [z]P)^x)$, $\hat{e}([x]P, [z]P)^y$ and $\hat{e}([x]P, [y]P)^z$ respectively. Now they can use this saved pairing for computing new kek_sid with new sid. Main protocol of Round 2, shown in picture below is as follows:

Step 1 $A \overset{EOO}{\rightarrow} B: M_1 = Sig_A\{A, B, TTP, S_id, M_id,$
$(M_{Subj}, h(M), h(M, S_id))_{kek\_sid}\},$
$\{A, B, TTP, S_id, M_{kek\_sid}\}_{TTP}\}$

Step 2 $B \overset{EOR}{\rightarrow} A: \quad M_2 = Sig_A\{M_1, M_{Subj}, M_id, h(M), h(M, S_id)\}$

Step 3 $A \rightarrow B: M_3 = Sig_A\{A, B, TTP, S_id, M_id, (M)_{kek\_sid}\},$

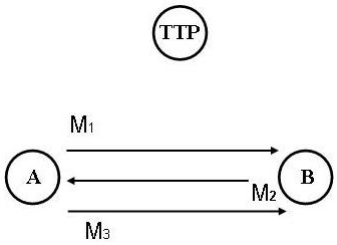


Figure 2. Second Round Message Flow

The steps defined above follow previous one after some checks, as;
In Step 2 receiver B checks the identities,signature of sender A and kek_sid is working properly by decrypting the message identifier encrypted in M_1.
In Step 3 sender A checks the identities, session identifier, signature of sender B and message subject M_{Subj} has been properly decrypted by B. If any of these checks fail then TTP cancels the protocol.

Cancellation Sub-protocol After Step 1 sender A can cancel the protocol by sending TTP a cancellation message. The TTP checks first if the signature is valid and the request is coming from the

sender of the message. The TTP confirms the *Cancellation* request if the status of the session is not Resolved. The cancellation sub-protocol works as follows;

Step 1 $A \overset{Cancel}{\rightarrow} TTP, B: \qquad M_1' = Sig_A\{Cancel, M_1\}$

Step 2 If $(Status(S_id) == Resolved)$

2.a Then $TTP \overset{Reject}{\rightarrow} A: \qquad M_2' = Sig_{TTP}\{Cancel - Reject, S_id, M_2\}$
$TTP \overset{Reject}{\rightarrow} B: \qquad M_3' = Sig_{TTP}\{Cancel - Reject, (M)_{kek\_sid}\}$

2.b Else $TTP \overset{Confirm}{\rightarrow} A, B: \qquad M_2' = Sig_{TTP}\{Cancel - Confirm, S_id, M_1'\}$
$(Status(S_id) = Cancelled)$

Dispute Resolution After Step 2 if receiver B does not get message M_3 or the hash of the message does not match with the hash in the first message, receiver B runs *Resolve* sub-protocol.
The *Resolve* sub-protocol works as follows;

Step 1 $B \overset{Resolve}{\rightarrow} TTP, A: \qquad M_1' = Sig_B\{Resolve, M_1, M_2\}$

Step 2 If $(Status(S_id) == Cancelled)$

2.a Then $TTP \overset{Reject}{\rightarrow} B, A: \qquad M_2' = Sig_{TTP}\{Resolve - Reject, S_id, Sig_{TTP}\{Cancel - Confirm, S_id, M_1'\}\}$

2.b Else $TTP \overset{Confirm}{\rightarrow} A, B: \qquad M_2' = Sig_{TTP}\{Resolve - Confirm, S_id, M_id, (M)_{kek\_sid}, M_1, M2\}, (Status(S_id) = Resolved)$

4.3 Protocol Analysis

4.3.1 Fairness and Non-Repudiation

Proposed non-repudiation protocol satisfies fairness in both rounds. By inclusion of online TTP in first round, TTP checks the previous messages, identities, signatures and finally send complementary evidences for both sender and receiver. This achieves strong fairness at the end of the protocol as either each party gets the expexted items (NRO,NRR,Message) or none of them gets a valuable information. If the sender denies, having sent a message M, the receiver can show $NRO = M_1 + M_6$ and adjudicator rejects denial unless the protocol is cancelled by TTP. In case of cancellation, the sender should show a confirmed cancellation. If the receiver denies, having received a message M, the sender can show $NRR = M_3 + M_5$ and adjudicator rejects denial unless the protocol is cancelled by TTP. In case of cancellation, the receiver should show a confirmed cancellation. Since *Cancellation* requests after Step 2 is not accepted, cancellation confirmation and messages M_5 and M_6 can not be present at same time.
For the second round, strong fairness is achieved by help of dispute resolution sub-protocols. Dishonest

users can try to get non-repudiation evidences hindering other party to get respective evidence. As a case for dishonest sender; after Step 2, A gets successfully EOR, but can misbehave as sending a cancellation request before a resolve request. In this case since the exchange will be cancelled by TTP and confirmation of cancellation is sent to both parties. Receiever can show to adjudicator that the exchange with S_id is cancelled and EOR in his message M_2 is not valid anymore. As a case for dishonest receiver; after Step 1, B gets successfully EOO, but can misbehave as sending a resolve request to only TTP. In this case the TTP will resolve the issue only if user B sends valid EOR, and this EOR in M_2 will be forwarded to sender A also.

4.3.2 Timeliness

Asynchronous timeliness is achieved in the proposed protocol by means of cancellation sub-protocols without any time constraint.

4.3.3 TTP State

TTP works in a statefull manner as has to keep states of protocol with respect to session identifiers. TTP also keeps securely keys for respective pariticipating parties.

4.3.4 Efficiency and Comparison

The communication and computation bottleneck of the protocol is TTP for the first round. Since TTP in our protocol acts also as PKG, this situation naturally increased the burden of TTP. But this is not a necessity, PKG and TTP can be different. In that case users should get both PKG parameters and TTP pairing parameters which requires two registration. For the next rounds pairing computations on both sides seems as the reason of computational burden when compared to traditional PKI signatures and encryption.
The proposed protocol has inevitably common characteristics with previously proposed non-repudiation protocols stated in [26], [58] and [2]. It satisfies the required properties as NRO, NRR, strong fairness and asynchronous timeliness but lacks in efficiency because of pairing computation, online TTP and statefull structure.
The advantage of using a hybrid protocol over other types (pure in-line, on-line or offline) is a kind of optimization between the security and performance. First online round embedded with Joux Tripartite key exchange scheme enhances the security and next rounds give better performance as being off-line. Our new design does not contribute new capabilities over previous protocols at the moment but it shows that non-repudiation protocols can be built on pairing based cryptography.

4.3.5 Key escrow and Revocation

Generally key escrow is accepted as a positive capability for authorized third party to gain access to keys needed to decrypt encrypted data. But from a view of non-repudiation key escrow property of full Identity-based cryptosystems is regarded as a negative capability. That is why we used identifier based encryption and signature schemes (Certificateless PKC). TTP can not hold an escrow capability for the private keys of users A and B as stated in [43]. Key revocation can not be handled properly by PKG in a full Identity-based cryptosystem. But by using identifier based encryption and signature

schemes (Certificateless PKC) this problem is also eliminated.
Another key escrow capability inherent from certificateless PKC, which is not mentioned by Riyami and Paterson in [43] for session keys, is as follows: If the user ID sends $Y_{ID} = [r][X_{ID}]P$ to TTP, then TTP can escrow session encryption key k as $k = V \oplus H_2(\hat{e}(Y_{ID}, U))$. This method is similar to Verheul's escrow scheme described in [48]. Here the user ID does not reveal its private key X_{ID} to the escrow agent.
But this inherent escrow property is not practically applicable to our protocol's encryption cases for user B and TTP. The case for user B in first round; B can not send $Y_B = [r][X_B]P$ to escrow authority because $[r]P$ is also encrypted with the Message for TTP. The case for TTP is not appropriate for key escrow since it acts also as the PKG. The usage of this property is only possible when TTP and PKG are different authorities. In that case TTP can send $Y_{TTP} = [r][X_{TTP}]P$ to an escrow authority (possibly PKG) for decryption.

4.3.6 Confidentiality

Confidentiality of the message is ensured in both rounds against evasdroppers. In the first round message is kept secret even to TTP, but in the second round message can be decrypted by TTP if the cancellation or dispute resolution sub-protocols executed. This property is inserted to improve the efficiency and generally TTP will not be joining the communication. If required, this property can be changed as it is done in the first round.

4.4 Certificateless ID-Based Signature and Encryption Scheme

To ensure non-repudiation in our protocol we used Riyami and Paterson's certificateless ID-Based encryption and signature schemes described in [43] to eliminate some of the disadvantages stated in Chapter 3. Here we present their seminal work in our notation.
The *setup* phase is same for both encryption and signature scheme:
Setup : Let $\mathbb{G}_1$ be additive group of prime order q and $\mathbb{G}_3$ be multiplicative group of prime order q. Choose an arbitrary generator $P \in \mathbb{G}_1$ and a random secret master key $s \in \mathbb{Z}_q^*$. Set $P_{pub} = [s]P$ choose cryptographic hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$ and $H_2 : \mathbb{G}_3 \rightarrow \{0,1\}^*$. Publish the system parameters $(\mathbb{G}_1, \mathbb{G}_3, \hat{e}, P, P_{pub}, H_1, H_2$. Public and private key pair for user ID is computed as follows:

- TTP or PKG computes $P_{pub} = [s]P$ and $[s]H_1(ID)$ and send to user ID.
- User ID computes $P_{pub\_ID} = [X_{ID}][s]P$ and send as public key then computes $d_{ID} = [X_{ID}][s]H_1(ID)$ as private key.

4.4.1 Certificateless ID-Based Encryption

Riyami and Paterson [43] has adapted the ID-Based Encryption Scheme of Boneh and Franklin [16] based on Bilinear Diffie Hellman problem.

4.4.1.1 Encryption

a. First choose a random $r \in \mathbb{Z}_q^*$

b. Message M encrypted by symmetric key k which is ciphered as $C = \langle [r]P, k \oplus H_2(g_{ID})^r \rangle$ where $g_{ID} = \hat{e}(H_1(ID), P_{pub\_ID})$

4.4.1.2 Decryption

With $\langle U, V \rangle$ k is computed as
$k = H_2(\hat{e}(d_{ID}, U)) \oplus V$

4.4.1.3 Proof of Decryption

Decryption works because; $V \oplus H_2(\hat{e}(d_{ID}, U))$
= $V \oplus H_2(\hat{e}(d_{ID}, [r]P))$
= $V \oplus H_2(\hat{e}([X_{ID}][s]H_1(ID), [r]P))$
= $V \oplus H_2(\hat{e}(H_1(ID), P)^{X_{ID} \cdot s \cdot r}$
= $V \oplus H_2(\hat{e}(H_1(ID), [X_{ID}][s]P)^r$
= $V \oplus H_2(g_{ID})^r$

4.4.2 Certificateless ID-Based Signature

Riyami and Paterson [43] has modified the ID-Based Signature Scheme of F.Hess [24] based on Weak Diffie Hellman problem.

4.4.2.1 Signature

For signing message M user ID, chooses an arbitrary $P_1 \in \mathbb{G}_1$ and a random $k \in \mathbb{Z}_q^*$

a. First compute $r = \hat{e}(P_1, P)^k)$

b. $v = H(M, r)$

c. $u = [v]d_{ID} + [k]P_1$

The signature is the pair $\langle u, v \rangle \in \langle \mathbb{G}_1, \mathbb{Z}_q \rangle$

4.4.2.2 Verification

When receiving a message M and signature $\langle \mathbb{G}_1, \mathbb{Z}_q \rangle$ verifier computes

a. $r = \hat{e}(u, P) \cdot \hat{e}(H_1(ID), -P_{pub\_ID})^v$

b. Accept the signature iff $v = H(M, r)$

4.4.2.3 Proof of Verification

Check if $r \stackrel{?}{=} \hat{e}(P_1, P)^k$ $r = \hat{e}(u, P) \cdot \hat{e}(H_1(ID), -P_{pub\_ID})^v$
$= \hat{e}([v]d_{ID} + [k]P_1, P) \cdot \hat{e}(H_1(ID), -P_{pub\_ID})^v$
$= \hat{e}([v][X_{ID}][s]H_1(ID) + [k]P_1, P) \cdot \hat{e}(H_1(ID), -P_{pub\_ID})^v$
$= \hat{e}([v][X_{ID}][s]H_1(ID), P) \cdot \hat{e}([k]P_1, P) \cdot \hat{e}(H_1(ID), -P_{pub\_ID})^v$
$= \hat{e}([v][X_{ID}][s]H_1(ID), P) \cdot \hat{e}([k]P_1, P) \cdot \hat{e}(H_1(ID), -[X_{ID}][s]P)^v$
$= \hat{e}([k]P_1, P) \cdot \hat{e}(H_1(ID), P)^{v \cdot X_{ID} \cdot s} \cdot \hat{e}(H_1(ID), -[X_{ID}][s]P)^v$
$= \hat{e}([k]P_1, P) \cdot \hat{e}(H_1(ID), P)^{v \cdot X_{ID} \cdot s} \cdot \hat{e}(H_1(ID), P)^{-v \cdot X_{ID} \cdot s}$
$= \hat{e}([k]P_1, P) = \hat{e}(P_1, P)^k$

4.5 Conclusion

We proposed a non-repudiation protocol which has a new structure based on pairing based cryptography. The hybrid structure consists of two rounds described in previous sections, first round runs with an online TTP then second and next rounds run with offline TTP. Although online TTP has been regarded as a bottle-neck for security protocols, this is not a big challenge nowadays with usage of high available servers and broad band internet connection. Our main contribution here is the usage of signed Joux Tri-partite key exchange scheme in the first round as a security enhancing method. Also the key derived from Joux Tri-partite key exchange scheme is used in next rounds which increases efficiency as use of symmetric cryptography instead of asymmetric cryptography. Previous works on non-repudiation protocols have used pairing based cryptography to take advantages of different properties but they also used traditional PKI for encryption and signatures. Differently our protocol is fully based on pairing based cryptography, especially certificateless ID based encryption and signature schemes which prevents some problems of pure ID-based systems.

CHAPTER 5

VERIFIABLY ENCRYPTED SIGNCRYPTION

5.1 Introduction

Contract signing protocols are being widely used over digital environment and treated as an application of non-repudiation protocols. As a kind of non-repudiation protocols, the most important property of contract signing protocols is fairness. Verifiably encrypted signatures are used mainly for fair exchange and contract signing protocols to sustain fairness in cryptograhic manner. Although cofidentiality of the message is not as important as fairness for ordinary contracts, in the case of secret contracts confidentiality will be as important as fairness. Signcryption as a cryptographic method combines signing and encryption usually in sing then encrypt order. In this chapter (which is published in [11]) we propose a new scheme that combines signcryption and verifiably encrypted signatures which we call VE-Signcrypt. To the best of our knowledge, this scheme is the first of its kind in the literature.

In Section 5.3 we first show verifiably encrypted signcryption scheme which is implementation of verifiably encryption to [51]. Then we expand it to multi-recipient version in Section 5.4. In Section 5.5 we present a fair two-party secret contract signing protocol using either single-recipient or multi-recipient version of verifiably encrypted signcryption scheme. And in the last section 5.7 we present another adaptation of verifiably encrypted signature to publicly verifiable signcryption method.

5.2 General Description

5.2.1 Signcryption

Signcryption was first introduced by Zheng [71] and then accrued many different signcryption methods [72]. Signcryption can be constructed in different orders as; sign-then-encrypt, encrypt-then-sign, commit-then-encrypt-and-sign paradigms. Also signcryption can be performed basically for single recepient or for multi-recipients. It is applicable in a wide area where both confidentiality and authenticity is required like e-voting.

5.2.2 Verifiably Encrypted Signatures

Verifiably encrypted signature was first introduced by Boneh et al [70] as a cryptographic primitive to satisfy mainly fairness in fair exchange, contract signing [44], [10] and certified electronic mail

protocols [5]. By using verifiably encrypted signatures in a protocol sender S can send an encrypted signature to a receiver R. The receiver R can check that signature validity but can not get the actual signature without help of an adjudicator. When the receiver requests from the adjudicator with valid reasons to adjudicate, he can recover the actual signature from verifiably encrypted signature.

5.3 Verifiably Encrypted Signcryption Scheme

Y.Han et. al. [51] have developed a signcryption method which was also extended to multi-recipient environment. We adapted this scheme to the verifiably encrypted signature scheme and called it shortly as VE-Signcrypt. The **Setup**, **Extract**, **Signcrypt**, **DeSigncrypt** steps are same as the original work [51]. Here are all the steps;

Setup : Let $\mathbb{G}_1$ be additive group of prime order q which is an n-bit prime and $\mathbb{G}_3$ be multiplicative group of prime order q. Choose an arbitrary generator $P \in \mathbb{G}_1$, a random secret PKG master key $s \in \mathbb{Z}_q^*$ and a random secret adjudicator master key $s_T \in \mathbb{Z}_q^*$. l is the bit length of elements in $\mathbb{G}_1$. Set $Y_T = [s]P$ choose cryptographic hash functions $H_1 : \{0,1\}^m X \mathbb{G}_1 \rightarrow \mathbb{G}_1$ and $H_2 : \mathbb{G}_1^3 \rightarrow \{0,1\}^{m+l}$. Publish the system parameters $(\mathbb{G}_1, \mathbb{G}_3, q, \hat{e}, P, Y_T, H_1, H_2)$

Extract : Public and private key pair for user ID is extracted as follows:

- TTP or PKG computes $Y_T = [s]P$ as public key and s as private key.
- User ID computes $Y_{ID} = [X_{ID}]P$ as public key and $X_{ID} \in \mathbb{Z}_q^*$ as private key.

Signcrypt : Sender ID=S with key pair (Y_S, X_S) sends a signcrypted message m to receiver ID=R with public key Y_R. Sender S picks a random $r \in \mathbb{Z}_q^*$, computes $U = [r]P, V = X_S H_1(m, rY_R), Z = (m||V) \oplus H_2(U, Y_R, rY_R)$ and output the signcryption $(U, Z) \in \mathbb{G}_1 X \{0,1\}^*$.

DeSigncrypt : Given a signcryption (U, Z) and public key of sender S, receiver R computes $(m||V) = Z \oplus H_2(U, Y_R, X_R U), h = H_1(m, X_R U)$ and then check if $\hat{e}(P, V) = \hat{e}(Y_S, h)$
if check passes, output $< m, (U, V, Y_R, X_R U), Y_S >$ as signature.

The correction of verification for a valid signcryption (U, Z) is as follows;
Since $X_R U = X_R rP = rY_R, Z$ signcryption can be decrypted successfully, then,
$\hat{e}(P, V) = \hat{e}(P, X_S H_1(m, rY_R))$
$= \hat{e}(P, V) = \hat{e}(X_S P, H_1(m, rY_R))$
$= \hat{e}(P, V) = \hat{e}(Y_S, h)$

VE-Signcrypt : Sender ID=S with key pair (Y_S, X_S) sends a verifiably encrypted signcrypted message m to receiver ID=R with public key Y_R and with public key Y_T of Adjudicator . Sender S picks two random $r_1$and $r_2 \in \mathbb{Z}_q^*$, computes $U_1 = [r_1]P, U_2 = [r_2]P, V = X_S H_1(m, r_1 Y_R) + r_2 Y_T, Z = (m||V) \oplus H_2(U_1, Y_R, r_1 Y_R)$ and output the signcryption $(U_1, U_2, Z) \in \mathbb{G}_1^2 X \{0,1\}^*$.

De-VE-Signcrypt : Given a verifiably encrypted signcryption (U_1, U_2, Z) and public key of sender S, receiver R computes $(m||V) = Z \oplus H_2(U_1, Y_R, X_R U_1), h = H_1(m, X_R U_1)$ and then check if $\hat{e}(P, V) = \hat{e}(Y_S, h)\hat{e}(U_2, Y_T)$
if check passes, output $< m, (U_1, U_2, V, Y_R, X_R U_1), Y_S >$ as verifiably encrypted signature.

The correction of verification for a valid verifiably encrypted signcryption (U_1, U_2, Z) is as follows;
Since $X_R U = X_R rP = rY_R$, Z signcryption can be decrypted successfully, then,
$\hat{e}(P, V) = \hat{e}(P, [X_S H_1(m, r_1 Y_R) + r_2 Y_T])$
$= \hat{e}(P, V) = \hat{e}(P, X_S H_1(m, r_1 Y_R))\hat{e}(P, r_2 Y_T)$
$= \hat{e}(P, V) = \hat{e}(X_S P, H_1(m, r_1 Y_R))\hat{e}(r_2 P, Y_T)$
$= \hat{e}(P, V) = \hat{e}(Y_S, h)\hat{e}(U_2, Y_T)$

Adjudication : Given the adjudicator's private key s_T and a valid verifiably encrypted signature (U_1, U_2, V) for a message m, compute $V_1 = V - [s_T]U_2$ and output the original signature (U_1, V_1)
The correction of adjudication for a valid verifiably encrypted signature (U_1, U_2, V) is as follows;
$V_1 = V - [s_T]U_2 = V - [s_T][r_2]P = V - [r_2]Y_T = X_S H_1(m, r_1 Y_R) + [r_2]Y_T - [r_2]Y_T = X_S H_1(m, r_1 Y_R)$
Here the receiver can not send the original verifiably encrypted signcryption (U_1, U_2, Z) to adjudicator since Z is encrypted for receiver. The adjudication process can be done by only (U_1, U_2, V) provided, but in a fair protocol adjudicator shall make some verifications, in that case De-VE-Signcrypted tuple $< m, (U_1, U_2, V, Y_R, X_R U_1), Y_S >$ as verifiably encrypted signature can be sent to adjudicator.

5.4 Multi-Recipient Verifiably Encrypted Signcryption Scheme

In this section we extended the verifiably encrypted signature scheme described in the previous section to multi-recipient environment and called it shortly as MR-VE-Signcrypt. The **Setup**, **Extract**, **MR-Signcrypt**, **MR-DeSigncrypt** steps are the same as in the original work [51]. Here are all the steps;

Setup : Let $\mathbb{G}_1$ be additive group of prime order q and $\mathbb{G}_3$ be multiplicative group of prime order q. Choose an arbitrary generator $P \in \mathbb{G}_1$, a random secret PKG master key $s \in \mathbb{Z}_q^*$ and a random secret adjudicator master key $s_T \in \mathbb{Z}_q^*$. Set $Y_T = [s]P$ choose cryptographic hash functions $H_1 : \{0,1\}^* X \mathbb{G}_1^2 \rightarrow \mathbb{G}_1$ and $H_2 : \mathbb{G}_1^3 \rightarrow \{0,1\}^*$. Publish the system parameters $(\mathbb{G}_1, \mathbb{G}_3, q, \hat{e}, P, Y_T, H_1, H_2)$

Extract : Public and private key pair for user ID is extracted as follows:

- TTP or PKG computes $Y_T = [s]P$ as public key and s as private key.
- User ID computes $Y_{ID} = [X_{ID}]P$ as public key and $X_{ID} \in \mathbb{Z}_q^*$ as private key.

MR-Signcrypt : Sender ID=S with key pair (Y_S, X_S) sends a messages m_i to receivers ID=$R_i, i = 1, ..., n$ with public keys Y_{R_i}. Sender S picks a random $r \in \mathbb{Z}_q^*$, computes $U = [r]P$ For i=1 to n;

- $V_i = X_S H_1(m_i, rY_{R_i})$,
- $Z_i = (m_i || V_i) \oplus H_2(U, Y_{R_i}, rY_{R_i})$

EndFor Finally output the signcryptions $(U, Z_i) \in \mathbb{G}_1 X \{0,1\}^{n+1}$.

MR-DeSigncrypt : Given a signcryption for receiver $R_i, (U, Z_i)$ and public key of sender S, receiver R_i computes $(m_i || V_i) = Z_i \oplus H_2(U, Y_{R_i}, X_{R_i} U), h_i = H_1(m_i, X_{R_i} U)$ and then check if

$\hat{e}(P, V_i) = \hat{e}(Y_S, h_i)$
if check passes, output $< m, (U, V_i, Y_{R_i}, X_{R_i}U), Y_S >$ as signature.

The correction of verification for a valid signcryption (U, Z_i) is as follows;
Since $X_{R_i}U = X_{R_i}rP = rY_{R_i}, Z_i$ signcryption can be decrypted successfully, then,
$\hat{e}(P, V_i) = \hat{e}(P, X_S H_1(m_i, rY_{R_i}))$
$= \hat{e}(P, V_i) = \hat{e}(X_S P, H_1(m_i, rY_{R_i}))$
$= \hat{e}(P, V_i) = \hat{e}(Y_S, h_i)$

MR-VE-Signcrypt : Sender ID=S with key pair (Y_S, X_S) sends verifiably encrypted messages m_i to receivers ID=$R_i, i = 1, ..., n$ with public keys Y_{R_i} and with public key Y_T of Adjudicator. Sender S picks two random $r_1$and $r_2 \in \mathbb{Z}_q^*$, computes $U_1 = [r_1]P, U_2 = [r_2]P, V = X_S H_1(m, r_1 Y_R) + r_2 Y_T, Z = (m||V) \oplus H_2(U_1, Y_R, r_1 Y_R)$, For i=1 to n;

- $V_i = X_S H_1(m_i, rY_{R_i}) + r_2 Y_T$,
- $Z_i = (m_i||V_i) \oplus H_2(U, Y_{R_i}, rY_{R_i})$

EndFor Finally output the verifiable encrypted signcryptions $(U_1, U_2, Z_i) \in \mathbb{G}_1^2 X\{0,1\}^{n+1}$.

MR-De-VE-Signcrypt : Given a verifiably encrypted signcryption (U_1, U_2, Z_i) and public key of sender S, receiver R_i computes $(m_i||V_i) = Z_i \oplus H_2(U_1, Y_{R_i}, X_R U_1), h_i = H_1(m_i, X_{R_i}U_1)$ and then check if $\hat{e}(P, V_i) = \hat{e}(Y_S, h_i)\hat{e}(U_2, Y_T)$
if check passes, output $< m_i, (U_1, U_2, V_i, Y_{R_i}, X_{R_i}U_1), Y_S >$ as verifiably encrypted signature.

The correction of verification for a valid verifiably encrypted signcryption (U_1, U_2, Z_i) is as follows;
Since $X_{R_i}U = X_{R_i}rP = rY_{R_i}, Z_i$ signcryption can be decrypted successfully, then,
$\hat{e}(P, V_i) = \hat{e}(P, [X_S H_1(m_i, r_1 Y_{R_i}) + r_2 Y_T])$
$= \hat{e}(P, V_i) = \hat{e}(P, X_S H_1(m_i, r_1 Y_{R_i}))\hat{e}(P, r_2 Y_T)$
$= \hat{e}(P, V_i) = \hat{e}(X_S P, H_1(m_i, r_1 Y_{R_i}))\hat{e}(r_2 P, Y_T)$
$= \hat{e}(P, V_i) = \hat{e}(Y_S, h_i)\hat{e}(U_2, Y_T)$

Adjudication : Given the adjudicator's private key s_T and a valid verifiably encrypted signature (U_1, U_2, V_i) for a message m_i, compute $V_{1_i} = V_i - [s_T]U_2$ and output the original signature (U_1, V_{1_i})
The correction of adjudication for a valid verifiably encrypted signature (U_1, U_2, V_i) is as follows;
$V_{1_i} = V_i - [s_T]U_2 = V_i - [s_T][r_2]P = V_i - [r_2]Y_T = X_S H_1(m_i, r_1 Y_{R_i}) + [r_2]Y_T - [r_2]Y_T$
$= X_S H_1(m_i, r_1 Y_{R_i})$
Here the receiver can not send the original verifiably encrypted signcryption (U_1, U_2, Z_i) to adjudicator since Z_i is encrypted for receiver. The adjudication process can be done by only (U_1, U_2, V_i) provided, but in a fair protocol adjudicator shall make some verifications, in that case MR-De-VE-Signcrypted tuple $< m, (U_1, U_2, V_i, Y_{R_i}, X_{R_i}U_1), Y_S >$ as verifiably encrypted signature can be sent to adjudicator.
As stated in [51], this scheme supports multi message to multi recipient. When $m_1 = m_2 = ...m_n = m$ then this scheme becomes a single message to multi recipient. When $R_1 = R_2 = ...R_n$ then this scheme becomes a single message to signle recipient.

5.5 Fair Two-Party Secret Contract Signing Protocol

In this section we propose a fair two-party optimistic secret contract signing protocol. We propose two alternative ways to define protocol; in the first case we use single recipient verifiably encrypted signcryption and in the second case we use multi recipient verifiably encrypted signcryption defined in previous sections.

5.5.1 First Case

Here is the steps for the first case with single recipient verifiably encrypted signcryption.

Step 1 $S \rightarrow R : ID_S, ID_R, VESigncrypt\{ID_S, ID_R, m\}$

Step 2 $R \rightarrow S : ID_R, ID_S, Signcrypt\{ID_R, ID_S, m\}$

Step 3 $S \rightarrow R : ID_S, ID_R, Signcrypt\{ID_S, ID_R, m\}$

- Step 1: Sender S computes verifiably encrypted signcryption (U_1, U_2, Z) of $\{ID_S, ID_R, m\}$ where m is the single message as secret contract. And sends to receiver $R < ID_S, ID_R, (U_1, U_2, Z) >$
- Step 2: Receiver R checks the validity of $< ID_S, ID_R, (U_1, U_2, Z) >$ by De-VE-Signcrypt (U_1, U_2, Z). If De-VE-Signcrypt successes then output and keeps $< ID_S, ID_R, m, (U_1, U_2, V, Y_R, X_R U_1), Y_S >$ as verifiably encrypted signature and sends back to Sender $S < ID_R, ID_S, Signcrypt\{ID_R, ID_S, m\}$, otherwise aborts the protocol.
- Step 3: Sender S checks the validity of $< ID_R, ID_S, (U, Z) >$ by De-Signcrypt (U, Z). if check passes, output and keeps $< m, (U, V, Y_R, X_R U), Y_S >$ as signature and sends back to Receiver R $ID_S, ID_R, Signcrypt\{ID_S, ID_R, m\}$, otherwise aborts the protocol.

If Receiver gets signcryption computed in step three and verification that signcryption passes than the protocol ends by success, otherwise Receiver can request arbitrament from adjudicator. Here is the steps for Adjudication;

- Step 1: Receiver R sends De-VE-Signcrypted $< ID_S, ID_R, m, (U_1, U_2, V, Y_R, X_R U_1), Y_S >$ to Adjudicator as verifiably encrypted signature. And computes an ordinary signature as $U = [r]P, V = X_R H_1(m, rY_S)$ and sends also ordinary signature $< m, (U, V, Y_S, rY_S), Y_R >$
- Step 2: Adjudicator checks the validity of ordinary signature $< m, (U, V, Y_S, rY_S), Y_R >$ as $\hat{e}(P, V) = \hat{e}(Y_R, h)$ where $h = H_1(m, rY_S)$ if check fails then aborts the protocol. Otherwise Adjudicates $(U_1, U_2, V, Y_R, X_R U_1)$ outputs the original signature (U_1, V_1) and checks the contract and identities and sends back to Receiver R (U_1, V_1). Then sends to Sender S ordinary signature $< m, (U, V, Y_S, rY_S), Y_R >$

5.5.2 Second Case

Here is the steps for the second case with multi recipient verifiably encrypted signcryption.

Step 1 $S \rightarrow R : ID_S, ID_R,$
$MR - VESigncrypt\{ID_S, ID_R, m\},$
$MR - VESigncrypt\{ID_S, ID_ADJ, m\}$

Step 2 $R \rightarrow S : ID_R, ID_S,$
$MR - Signcrypt\{ID_R, ID_S, m\},$
$MR - Signcrypt\{ID_R, ID_ADJ, m\}$

Step 3 $S \rightarrow R : ID_S, ID_R,$
$MR - Signcrypt\{ID_S, ID_R, m\},$
$MR - Signcrypt\{ID_S, ID_ADJ, m\}$

- Step 1: Sender S computes multi-recipient verifiably encrypted signcryption (U_1, U_2, Z_1, Z_2) of $\{ID_S, ID_R, ID_{ADJ}, m\}$ where m is the single message as secret contract. And sends to receiver R $< ID_S, ID_R, (U_1, U_2, Z_1, Z_2) >$
- Step 2: Receiver R checks the validity of $< ID_S, ID_R, (U_1, U_2, Z_1, Z_2) >$ by MR-De-VE-Signcrypt (U_1, U_2, Z_1, Z_2). If MR-De-VE-Signcrypt successes then output and keeps $< ID_S, ID_R, m, (U_1, U_2, V_1, Y_R, X_R U_1), Y_S >$ as verifiably encrypted signature and sends back to Sender S $< ID_R, ID_S, MR - Signcrypt\{ID_R, ID_S, ID_{ADJ}, m\}$, otherwise aborts the protocol.
- Step 3: Sender S checks the validity of $< ID_R, ID_S, ID_{ADJ}, (U, Z_1, Z_2) >$ by MR-De-Signcrypt (U, Z_1, Z_2). if check passes, output and keeps $< m, (U, V_1, Y_R, X_R U), Y_S >$ as signature and sends back to Receiver R ID_S, ID_R, ID_{ADJ},$MR-Signcrypt\{ID_S, ID_R, ID_{ADJ}, m\}$, otherwise aborts the protocol.

If Receiver gets signcryption computed in step three and verification that signcryption passes than the protocol ends by success, otherwise Receiver can request arbitrament from adjudicator. Here is the steps for Adjudication;

- Step 1: Receiver R sends original message sent in Step 1 to adjudicator as (U_1, U_2, Z_1, Z_2) of $\{ID_S, ID_R, ID_{ADJ}, m\}$. And also sends original message sent in step 2 to adjudicator as $< ID_R, ID_S, MR - Signcrypt\{ID_R, ID_S, ID_{ADJ}, m\}$
- Step 2: Adjudicator checks the validity of $< ID_S, ID_R, ID_{ADJ}, (U_1, U_2, Z_1, Z_2) >$ by MR-De-VE-Signcrypt (U_1, U_2, Z_1, Z_2) and checks the validity of $< ID_R, ID_S, ID_{ADJ}, (U, Z_1, Z_2) >$ by MR-De-Signcrypt (U, Z_1, Z_2). If the second check fails then aborts the protocol but if the first check fails or the contract in two messages are different than requests MR-De-VE-Signcrypted version of (U_1, U_2, Z_1, Z_2) from Receiver. If MR-De-VE-Signcrypted version as $< ID_S, ID_R, m, (U_1, U_2, V_1, Y_R, X_R U_1), Y_S >$ validates then adjudicates the first message $< ID_S, ID_R, ID_{ADJ}, (U_1, U_2, Z_1, Z_2) >$ outputs the original signature (U_1, V_1) and checks the contract and identities and sends back to Receiver R (U_1, V_1). Then sends to Sender S ordinary signature $< m, (U, V, Y_S, rY_S), Y_R >$

5.6 Security and Performance Analysis

There are three security notions that a verifiably encrypted signcryption should satisfy, namely confidentiality, unforgeability and opacity. Confidentiality and unforgeability is required for both signcryption and verifiably encrypted signcryption while opacity is required for only verifiably encrypted signcryption.

Confidentiality and unforgeability for signcryption has been shown in the random oracle model under the hardness of CDH in [51]. Since our scheme's signcryption part is same as the original work, we will present security analysis regarding confidentiality and unforgeability for verifiably encrypted signcryption.

Opacity means that, given a verifiably encrypted signature, it is not possible to get a valid signature on the same message and the same recepient. By this respect we can define opacity for verifiably encrypted signcryption scheme as; given a verifiably encrypted signcryption text, it is not possible to get a valid signcryption on the same message and the same recepient.

5.6.1 Confidentiality of VESigncrypt

Theorem 5.6.1. *In the random oracle model, if there is an adversary $\mathbb{A}_0$ that performs an attack against IND-CCA2 of our VE-Signcrypt with non-negligible advantage ϵ running time in* t *and performing q_{VeSC} verifiably encrypted signcryption queries, q_{DeVeSC} verifiably encrypted designcryption queries, and q_{H1} and q_{H2} queries to oracles H_1 and H_2, then there is an algorithm $\mathbb{A}_1$ that solves the CDH problem in G_1 with probability $\epsilon' \geq \epsilon - q_{DeVESC}(q_{H_1}/2^{n-1} + q_{H_2}/2^{m+l})$ with running time $t' = t + (5q_{De-VESC} + 2q_{H2})t_p + 4q_{VESC}t_{sm}$.*

Proof. With help of $\mathbb{A}_0$ we can construct an adversary $\mathbb{A}_1$ for solving the CDH problem. When $\mathbb{A}_1$ is given with (P, aP, bP), he runs $\mathbb{A}_0$ as a subalgorithm to find the solution abP. Since VE-Signcrypt processes are based on signcryption processes of [51], hash, VE-Signcrypt and De-VE-Signcrypt queries are similar to work [51]. $\mathbb{A}_1$ constructs three lists L_1, L_2, L_3 for oracle queries H_1, H_2 and to simulations of VE-Signcrypt and De-VE-Signcrypt.
H_1 and H_2 simulations are same as [51] except when returning hP from H_1 oracle, $\mathbb{A}_1$ maintains another list L_3 as (h, r_2P, r_2Y_T) as r_2 picked randomly for each query.
VE-Signcrypt Simulation: When a VE-Signcrypt query for (m, Y_R) chosen by $\mathbb{A}_0$, $\mathbb{A}_1$ checks first if $Y_R \notin \mathbb{G}_1$ or $Y_R = Y_S$ or $Y_R = Y_T$, then rejects the query. Otherwise $\mathbb{A}_1$ picks randomly $r_1 \in \mathbb{Z}_q^*$, computes the result of $U_1 = r_1P$, then simulates $H_1(m, r_1Y_R)$ and gets hP from list L_1 and (r_2P, r_2Y_T) from L_3. Sets $U_2 = r_2P$ and $V = X_SH_1(m, r_1Y_R) + r_2Y_T = hY_S + r_2Y_T = h(bP) + r_2Y_T$ and computes the result of $Z = (m||V) \oplus H_2(U_1, Y_R, r_1Y_R)$ and output the signcryption (U_1, U_2, Z) with sender's public key $Y_S = bP$.
De-VE-Signcrypt Simulation: When a VE-Signcrypted test (U_1, U_2, Z) arrives, $\mathbb{A}_1$ checks first if (U_1, Y_R, F_i, v_i) is in the list L_2, for $0 \leq i \leq q_{H_2}$, such that $Z \oplus v_i = m_i||V_i$ for the corresponding elements (m_i, F_i, h_i) in list L_1 and corresponding r_2Y_T in list L_3, which satisfies $V_i = h_ibP + r_2Y_T$. If one of them satisfies $\hat{e}(P, F_i) = \hat{e}(U_1, Y_R)$ and $\hat{e}(P, V_i) = \hat{e}(Y_{S_i}, h_i)\hat{e}(U_2, Y_T$ then returns (m_i, U_1, U_2, V_i) to $\mathbb{A}_0$, else reutrns 0.
Second stage of proof is same as [51] except the probability and running time as follows. For the queries on H_1 the probability is no more than $q_{H_1}/2^n + q_{H_1}/2^n = q_{H_1}/2^{n-1}$ and for the queries on H_2 the probability is no more than $q_{H_2}/2^{m+l}$. Hence the probability of adversary $\mathbb{A}_1$ wins is $\epsilon' \geq \epsilon - q_{DeVESC}(q_{H_1}/2^{n-1} + q_{H_2}/2^{m+l})$

For the running time of adversary $\mathbb{A}_1$, we only count pairing and scalar multiplication operations. Its running time is evaluated as, 5 pairing operations for each De-VE-Signcrypt simulation, 2 pairing operation 4 scalar multiplication operations for each VE-Signcrypt simulation which includes H_1 and H_2 oracles. so the overall running time is $t' = t + (5q_{De-VESC} + 2q_{H2})t_p + 4q_{VESC}t_{sm}$ where t_p stands for pairing evaluation time and t_{sm} stands for scalar multiplication evaluation time. □

5.6.2 Unforgeability of VESigncrypt

Theorem 5.6.2. *In the random oracle model, if there is a forger $\mathbb{F}_0$ that forges a valid VE-Signcryption text with non-negligible advantage ϵ running time in* t *and performing q_{VeSC} verifiably encrypted signcryption queries, q_{DeVeSC} verifiably encrypted designcryption queries, and q_{H1} and q_{H2} queries to oracles H_1 and H_2, then there is an algorithm $\mathbb{F}_1$ that solves the CDH problem in G_1 with probability $\epsilon' \geq \epsilon - (q_{VESC}(q_{H_1} + 1)/2^n$ with running time $t' = t + q_{VESC}(2q_{H2})t_p + (2q_{VESC} + 3q_{VESC}q_{H1})t_{sm}$.*

Proof. With help of $\mathbb{F}_0$ we can construct an adversary $\mathbb{F}_1$ for solving the CDH problem. When $\mathbb{F}_1$ is given with (P, aP, bP), he runs $\mathbb{F}_0$ as a subalgorithm to find the solution abP. $\mathbb{F}_1$ constructs three lists L_1, L_2, L_3 for oracle queries H_1, H_2 and to simulation of VE-Signcrypt except H_1 returns haP instead of hP. In the second stage $\mathbb{F}_0$ produces signcryption text (U_1', U_2', Z'). $\mathbb{F}_1$ validates the text as $\hat{e}(P, V') = \hat{e}(Y_S, H')\hat{e}(U_2', Y_T$ if it is a valid verifiably encrypted signcryption text. And if $H_1(m', r_1Y_R)$ is in the list L_1 and (r_2P, r_2Y_T) is in the list L_3 it is easy to see that $V' = habP + r_2Y_T$, then $\mathbb{F}_1$ can compute $abP = h^{-1}(V' - r_2Y_T)$.
The probability of adversary $\mathbb{F}_1$ wins is not different than the probability of [51] as $\epsilon' \geq \epsilon - (q_{VESC}(q_{H_1} + 1)/2^n$. The running time of adversary $\mathbb{F}_1$ sums up, 2 pairing operation for each H_2 query, 3 scalar multiplication operations for each H_1 query and 2 scalar multiplication operations for each VE-Signcrypt simulation. So the running time of $\mathbb{F}_1$ is $t' = t + q_{VESC}(2q_{H2})t_p + (2q_{VESC} + 3q_{VESC}q_{H1})t_{sm}$ where t_p stands for pairing evaluation time and t_{sm} stands for scalar multiplication evaluation time. □

5.6.3 Opacity of VESigncrypt

Since adjudication can only be applied to verifiably encrypted signature (U_1, U_2, V) we can consider opacity attack like forgery in Theorem 2 except list L_3 is not provided and $Y_T = aP, U_2 = (bP - hP)$, then $V' = V - r_2P_T = V - s_TU_2$ and $V' = haP - a(bP - hP)$ so $\mathbb{F}_1$ can compute $abP = -1(V')$ with same propability and running time as in Theorem 2.

5.6.4 Performance Analysis

We compare our VE-Signcrypt with [51] to give computational overheads of adding verifiably encryption to signcryption. Here SM, PC, PA, FM, H_1, H_2 denotes scalar multiplication, pairing computation, point addition in $\mathbb{G}_1$, field multiplication in $\mathbb{G}_3$, hash functions 1 and 2, respectively. In Table 6.1 and Table 5.2 there is a minor computational overhead of adding verifiably encryption. Since the **Setup**, **Extract**, **Signcrypt**, **DeSigncrypt**, **MR-Signcrypt**, **MR-DeSigncrypt** steps are same as the original work there is no overhead in these steps. For single recipient case extra 2 SM, 1 PA and 1 PC, 1 FM is added for **VE-Signcrypt** and **VE-DeSigncrypt**, respectively. For multi-recipient case extra 2 SM, n PA and n PC, n FM is added for **MR-VE-Signcrypt** and **MR-VE-DeSigncrypt**, respectively.

Table5.1: Comparison of our scheme with [51] for single recipient

	[51]	Proposed
Key Gen	1 SM for each user	1 SM for each user
Sign	3 SM, 1 H1, 1 H2	3 SM, 1 H1, 1 H2
Design	1 SM, 1 H1, 1 H2, 2 PC	1 SM, 1 H1, 1 H2, 2 PC
VE-Sign	-	5 SM, 1 H1, 1 H2, 1 PA
VE-Desig	-	1 SM, 1 H1, 1 H2, 3 PC, 1 FM
Adj	-	1 SM, 1 PA

Table5.2: Comparison of our scheme with [51] for multi-recipient

	[51]	Proposed
Key Gen	n SM	n SM
Sign	(2n+1) SM, n H1, n H2	(2n+1) SM, n H1, n H2
Design	n SM, n H1, n H2, 2n PC	n SM, n H1, n H2, 2n PC
VE-Sign	-	(2n+3) SM, n H1, n H2, n PA
VE-Design	-	n SM,n H1,n H2,3n PC,n FM
Adj	-	1 SM, n PA

Below we present our new Verifiably Encrypted Signcryption scheme running times in seconds. Lynn's [105] PBC library is used to implement proposed signcryption scheme. Proposed signcryption scheme is implemented over Cygwin on a PC running Windows 10 with specifications; Intel Atom CPU N450, 1,66 GHz with 2 GB RAM. Code of signcryption is compiled by GNU C Compiler with library versions GMP 6.1.2/PBC 0.5.14. Type a parameters are constructed on the curve $y^2 = x^3 + x$ over the field $\mathbb{F}_q$ where q is a 512 bit prime with embedding degree 2, so the $\mathbb{G}_3$ is a subgroup of $\mathbb{F}_q^2$. r is a 160 bit Solinas prime. Type a1 uses the the same equation but with different field of size 1033 bit prime and large group size of 1022 bit length. Type e are constructed on the curve $y^2 = x^3 + ax + b$ over the field $\mathbb{F}_q$ where q is a 1024 bit prime with embedding degree 1, with 160 bit Solinas prime r. As Lynn reported in manual there is not optimizations in this type thus results in a slower pairing implementation.

Table5.3: Running times of our scheme with different supersingular curves

	a param	a.1 param	e param
Key Gen	0.076624	0.621663	0.151811
Sign	0.111440	1.199413	0.204752
Design	0.086951	1.527697	0.252265
VE-Sign	0.135589	1.542669	0.265134
VE-Desig	0.102912	1.999532	0.316904
Adj	0.124244	1.591876	0.245098
All time	0.637760	8.482850	1.435964

5.7 Public Verifiable Verifiably Encrypted Signcryption Scheme

In previous section we showed a verifiably encrypted signcryption method and used in a fair two-party optimistic secret contract signing protocol. In fair secret contract signing protocol, adjudication is done by sending the verifiable encrypted signature to the trusted third party instead of signcryption. Although; sending plaintext of the contract is acceptable as shared with only trusted third party, sometimes it may not be desired. This is a result of using signcryption which is verifiably encrypted (((or signcrypted for only))) for only receiver. To overcome this sharing problem here we present another adaptation of verifiably encrypted signature to publicly verifiable signcryption method.
Selvi et al [52] have developed a signcryption method which is publicly verifiable. We adapted this scheme to the verifiably encrypted signature scheme and called it shortly as PV-VE-Signcrypt. The **Setup**, **Extract**, **Signcrypt**, **DeSigncrypt** steps are same as the original work [52]. Here are all the steps;

Setup : Let $\mathbb{G}_1$ be additive group of prime order q which is an n-bit prime and $\mathbb{G}_3$ be multiplicative group of prime order q. Choose an arbitrary generator $P \in \mathbb{G}_1$, a random secret PKG master key $s \in \mathbb{Z}_q^*$ and a random secret adjudicator master key $s_{Adj} \in \mathbb{Z}_q^*$. l is the bit length of elements in $\mathbb{G}_1$. Set $P_{Pub} = [s]P$, choose cryptographic hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$, $H_2 : \mathbb{G}_2 \rightarrow \{0,1\}^{\|symetricencrkey\|}$, $H_3 : \{0,1\}^{\|ciphertext\|} X \mathbb{G}_1^3 \rightarrow \mathbb{G}_1$, $H_4 : \{0,1\}^{\|plaintext\|} X \mathbb{G}_2 X \mathbb{G}_1^4 \rightarrow \mathbb{G}_1$. Publish the system parameters $(\mathbb{G}_1, \mathbb{G}_3, q, \hat{e}, P, P_{Pub}, P_{Adj}, H_1, H_2, H_3, H_4)$

Extract : Public and private key pair for system, adjudicator and user ID is extracted by TTP or PKG as follows:

- $P_{Pub} = [s]P$ as public key and s as master private key.
- for user ID $Q_{ID} = H_1(ID)$ as public key and $D_{ID} = [s]Q_{ID} \in \mathbb{G}_1$ as private key.
- for adjudicator $P_{Adj} = [s_{Adj}]P$ as public key and s_{Adj} as adjudicator private key. Here we regard adjudicator as another authority but it can be the same authority as TTP/PKG. In that case P_{Pub} can be used instead of P_{Adj}.

Signcrypt : Sender ID=S with key pair (Q_S, D_S) sends a signcrypted message m to receiver ID=R with public key Q_R. Sender S picks a random $x \in \mathbb{Z}_q^*$, computes $U = [x]P, \alpha_1 = \hat{e}(P_{Pub}, Q_R)^x$, $\alpha_2 = H_2(\alpha_1), r = H_4(m, \alpha_1, U, Q_S, Q_R), c = Enc_{\alpha_2}(m||r), R = H_3(c, U, Q_S, Q_R), V = [x]R + D_A$ and output the signcryption $\sigma =< U, V, c > \in \mathbb{G}_1^2 X \{0,1\}^*$.

UnSigncrypt : Receiver ID=R with key pair (Q_R, D_R) and public key Q_S of sender ID=S unsingcrypts a signcryption $\sigma =< U, V, c >$. First check if $Public_Verify(\sigma, S, R)$ is valid or invalid. If signcryption is valid continues to unsigncryption as follows. Computes; $\alpha_1 = \hat{e}(U, D_R), \alpha_2 = H_2(\alpha_1), m||r = Dec_{\alpha_2}(c)$ Output $\phi =< m, r, \alpha_1, \sigma >$ iff $r = H_4(m, \alpha_1, U, Q_S, Q_R)$.

PublicVerify : Any user can verify a signcryption σ with public keys Q_S, Q_R, P_{Pub}. Compute $R = H_3(c, U, Q_S, Q_R)$, if $\hat{e}(V, P) = \hat{e}(U, R)\hat{e}(Q_A, P_{Pub})$ then it is a valid signcryption, else it is an invalid signcryption.

TP-Verify : Trusted party can verify a designcrypted signcryption $\phi =< m, r, \alpha_1, \sigma >$ with public keys Q_S, Q_R, P_{Pub}. First check if $Public_Verify(\sigma, S, R)$ is valid or invalid. If signcryption is valid continues to TP-Verify as follows. Computes; $\alpha_2 = H_2(\alpha_1)$, $m'||r' = Dec_{\alpha_2}(c)\phi$ is valid iff $r' = H_4(m', \alpha_1, U, Q_S, Q_R)$ and $r' = r$ received.

PVVerEnc-Signcrypt : Sender ID=S with key pair (Q_S, D_S) sends a publicly verifiable verifiably encrypted signcrypted message m to receiver ID=R with public key Q_R and adjudicator public key P_{Adj}. Sender S picks two random $x_1, x_2 \in \mathbb{Z}_q^*$, computes $U_1 = [x_1]P, U_2 = [x_2]P, \alpha_1 = \hat{e}(P_{Pub}, Q_R)_1^x, \alpha_2 = H_2(\alpha_1), r = H_4(m, \alpha_1, U_1, Q_S, Q_R), c = Enc_{\alpha_2}(m||r), R = H_3(c, U_1, Q_S, Q_R), V = [x_1]R + D_A + [x_2]P_{Adj}$ and output the signcryption $\sigma =< U_1, U_2, V, c > \in \mathbb{G}_1^3 X\{0,1\}^*$.

PVVerEnc-UnSigncrypt : Receiver ID=R with key pair (Q_R, D_R) and public key Q_S of sender ID=S unsingcrypts a signcryption $\sigma =< U_1, U_2, V, c >$. First check if $VerEnc-PublicVerify(\sigma, S, R)$ is valid or invalid. If signcryption is valid continues to unsigncryption as follows. Computes; $\alpha_1 = \hat{e}(U_1, D_R), \alpha_2 = H_2(\alpha_1), m||r = Dec_{\alpha_2}(c)$ Output $\phi =< m, r, \alpha_1, \sigma >$ iff $r = H_4(m, \alpha_1, U_1, Q_S, Q_R)$. This ϕ is different than the previous signature as it is a verifiably encrypted signature.

VerEnc-PublicVerify : Any user can verify a publicly verifiable verifiably encrypted signcryption σ with public keys $Q_S, Q_R, P_{Pub}, PAdj$. Compute $R = H_3(c, U_1, Q_S, Q_R)$, if $\hat{e}(V, P) = \hat{e}(U_1, R)\hat{e}(Q_A, P_{Pub})\hat{e}(U_2, P_{Adj})$ then it is a valid signcryption, else it is an invalid signcryption.

VerEnc-TP-Verify : Trusted party can verify a designcrypted verifiably encrypted signcryption $\phi =< m, r, \alpha_1, \sigma >$ with public keys Q_S, Q_R, P_{Pub}. First check if $Public_Verify(\sigma, S, R)$ is valid or invalid. If signcryption is valid continues to TP-Verify as follows. Computes; $\alpha_2 = H_2(\alpha_1)$, $m'||r' = Dec_{\alpha_2}(c)\phi$ is valid iff $r' = H_4(m', \alpha_1, U_1, Q_S, Q_R)$ and $r' = r$ received.

Adjudication : Adjudicator starts the process first checking VerEnc-PublicVerify of $\sigma =< U_1, U_2, V, c >$. If it is valid then with his private key s_{Adj}, computes $V_1 = V - [s_{Adj}]U_2$ and output the original public verifiable verifiably encrypted signcryption (U_1, V_1, c). The correction of adjudication for a valid verifiably encrypted signature (U_1, U_2, V) is as follows;
$V_1 = V - [s_{Adj}]U_2 = V - [s_{Adj}][x_2]P = V - [x_2]P_{Adj} = [x_1]R + D_A$
After adjudication, adjudicator should get a signcrypted message only and he can send it to receiver who can UnSigncrypt. By this way adjudication can be done without reaching the message, by authenticating the sender only. If message authentication is also needed and adjudicator can reach the message the adjudicator can first check with VerEnc-TP-Verify. Then adjudicates same as above.

5.8 Conclusion

In this chapter we propose a new scheme (up to our knowledge the first) which combines signcryption and verifiably encrypted signatures which we call VESigncrypt with a proof of security. Then extent it to multi-recipient environment and called it shortly as MR-VE-Signcrypt and use this scheme in a fair two-party optimistic secret contract signing protocol. Finally we propose a publicly verifiable verifiably encrypted signcryption scheme, to the best of our knowledge, this scheme is the first of its kind in the literature.

CHAPTER 6

POST-QUANTUM ASPECTS OF PAIRINGS

6.1 Post-Quantum Cryptography

Post-Quantum Cryptography is an emerging subject in cryptology as quantum computing is seen more realistic with higher computing power. Post-Quantum Cryptography contains Lattice, Code, Multivariate, Hidden Field Extension, Isogeny and other difficult problems which are not based on classical problems of public key algorithms that can be broken by quantum algorithms like Shor [73] and Grover's.

6.2 Isogeny Based Cryptography

Isogeny Based Cryptography uses hardness of finding isogeny ϕ between different elliptic curves as $E_1 = \phi(E_0)$ without knowing kernels G_0 and G_1 of isogenies which are subgroups of elliptic curves as E_0, E_1 respectively. After the first proposal of Isogeny Based Cryptosystems by Couvegnes [74] and other developments, it attracted the researchers by the introduction of supersingular isogeny Diffie-Hellman (SIDH) key exchange scheme of Jao and De Feo [75]. Originally, isogeny based key exchange protocol was previously proposed by [76] with ordinary elliptic curves, but because of its commutativity, it is subject to subexponential quantum attack [77]. Afterwards many isogeny based crypto systems have been proposed such as undeniable signature [78], identification and encryption scheme [79], designated verifier signature [80], Undeniable blind signature scheme [82].

6.2.1 Problems for Isogeny

Problem 1: Supersingular Isogeny Problem
$E_0/K, E_1/K$ are two supersingular elliptic curves such that $|E_0| = |E_1|$, compute an isogeny $\phi : E_0 \to E_1$

Problem 2: Endomorphism Ring Computation
Compute endomorphism ring of a given elliptic curve E defined over a field K.

Problem 3: Computational Supersingular Isogeny Problem
Let p be a prime of the form $l_A^a l_B^b f \pm 1$, E_0 be a supersingular curve over $\mathbb{F}_{p^2}$, $< P_A, Q_A >$ and $< P_B, Q_B >$ be bases for $E_0[l_A^a]$ and $E_0[l_B^b]$ respectively. Let $\phi : E_0 \to E_1$ be an isogeny with kernel $< [m]P_A + [n]Q_A >$, where m,n are chosen uniformly random and not both divisible by l_A. Compute

a generator of $< [m]P_A + [n]Q_A >$, given E_1 and the values $\phi(P_B), \phi(Q_B)$.

Problem 4: Supersingular Computational Diffie-Hellman Problem
Let $< [m_a]P_A + [n_a]Q_A >$ be kernel for an isogeny $\phi_A : E \to E_A$ and $< [m_b]P_B + [n_b]Q_B >$ be kernel for an isogeny $\phi_B : E \to E_B$ where m_a, n_a are chosen randomly from $\mathbb{Z}/l_A^a\mathbb{Z}$ and not both divisible by l_A (m_b, n_b are chosen respectively). Compute the j-invariant of $E/ < [m_a]P_A + [n_a]Q_A, [m_b]P_B + [n_b]Q_B >$ by using $E_A, E_B, \phi_A(P_B), \phi_A(Q_B), \phi_B(P_A), \phi_B(Q_A)$.

Problem 5: Supersingular Decisional Diffie-Hellman Problem
From the following two distributions;
- $E_A, E_B, E_{AB}, \phi_B(P_A), \phi_B(Q_A), \phi_A(P_B), \phi_A(Q_B)$ where $E_A, E_B, \phi_B(P_A), \phi_B(Q_A), \phi_A(P_B), \phi_A(Q_B)$ are like in the Supersingular Computational Diffie-Hellman Problem and
$E_{AB} \cong E/ < [m_a]P_A + [n_a]Q_A, [m_b]P_B + [n_b]Q_B >$.
- $E_A, E_B, E_X, \phi_B(P_A), \phi_B(Q_A), \phi_A(P_B), \phi_A(Q_B)$ where all the variables are same as the first case except E_X
$E_X \cong E/ < [m'_a]P_A + [n'_a]Q_A, [m'_b]P_B + [n'_b]Q_B >$.
determine from which distribution the tuple is sampled when tuples are sampled with same probablity. Jao and De Feo [75] key exchange scheme and encryption scheme are secure under Supersingular Decisional Diffie-Hellman Problem.

6.2.2 Isogeny Based Schemes

6.2.2.1 Key Exchange Scheme

Supersingular Isogeny Diffie-Hellman (SIDH) key exchange scheme of Jao and De Feo [75] is a breakthrough for Isogeny Based Cryptography as here we discuss.

Scheme 1: Jao-De Feo Key Exchange Scheme

Setup:
p is a prime of the form $2^a.3^b.f - 1$ where $2^a \approx 3^b$ and f is small. E is a supersingular elliptic curve over the field $\mathbb{F}_{p^2}$. Choose linearly independent points $P_A, Q_A \in E[2^a]$ and $P_B, Q_B \in E[3^b]$ which are bases for $E[2^a]$ and $E[3^b]$ respectively. Thus $| < P_A, Q_A > | = 2^2a$ and $| < P_B, Q_B > | = 3^2b$.
Protocol:

1. User A picks random integers $m_a, n_a < 2^a$ (Not both divisible by 2). User B picks random integers $m_b, n_b < 3^b$ (Not both divisible by 3). User A computes $G_A =< [m_a]P_A + [n_a]Q_A >$, User B computes $G_B =< [m_b]P_B + [n_b]Q_B >$.

2. By using Velu's formula they then compute ϕ_A and ϕ_B respectively. User A computes $E_A = \phi_A(E) = E/G_A, \phi_A(P_B), \phi_A(Q_B)$ and send to user B $E_A, \phi_A(P_B), \phi_A(Q_B)$. User B computes $E_B = \phi_B(E) = E/G_B, \phi_B(P_A), \phi_B(Q_A)$ and send to user A $E_B, \phi_B(P_A), \phi_B(Q_A)$.

3. After receiving each others messages, user A computes $< [m_a]\phi_B(P_A) + [n_a]\phi_B(Q_A) >=< \phi_B([m_a]P_A + [n_a]Q_A) >= \phi_B(G_A)$, user B computes $< [m_b]\phi_A(P_B) + [n_b]\phi_A(Q_B) >=< \phi_A([m_b]P_A + [n_b]Q_B) >= \phi_A(G_B)$.

4. Then user A calculates isogeny $\phi'_A : E_B \to E_{BA} = E_B/\phi_B(G_A)$ and user B calculates isogeny $\phi'_B : E_A \to E_{AB} = E_A/\phi_A(G_B)$. Finally they use their common j-invariant $j(E_{AB}) = j(E_{BA})$ as their secret key.

The following diagram summarizes the steps

$$\begin{array}{ccc} E & \xrightarrow{\phi_A} & E_A \\ {\scriptstyle \phi_B}\downarrow & & \downarrow{\scriptstyle \phi'_B} \\ E_B & \xrightarrow{\phi'_A} & E_{BA} = E_{AB} \end{array}$$

6.2.2.2 Man-In-The-Middle Attack

A **Attacker** **B**

Input: A,B,P_B, Q_B
Compute: m_a, n_a
$G_A =< [m_a]P_A + [n_a]Q_A >$

$E_A = \phi_A(E) = E/G_A$
$\phi_A(P_B)$
$\phi_A(Q_B)$

$\xrightarrow{Send}$

Input: A,B,P_A, Q_A
Compute: m_b, n_b
$G_B =< [m_b]P_B + [n_b]Q_B >$

Compute: m_a^*, n_a^*
$G_A^* =< [m_a^*]P_A + [n_a^*]Q_A >$

$E_A^* = \phi_A^*(E) = E/G_A^*$
$\phi_A^*(P_B)$
$\phi_A^*(Q_B)$

$\xrightarrow{Send}$

$E_B = \phi_B(E) = E/G_B$
$\phi_B(P_A)$
$\phi_B(Q_A)$

$\xleftarrow{Send}$

Compute: m_b^*, n_b^*
$G_B^* =< [m_b^*]P_B + [n_b^*]Q_B >$

$E_B^* = \phi_B * (E) = E/G_B^*$
$\phi_B^*(P_A)$
$\phi_B^*(Q_A)$

$\xleftarrow{Send}$

A:
$< [m_a]\phi_B^*(P_A) + [n_a]\phi_B^*(Q_A) >=$
$= \phi_B^*(G_A)$
$\phi'_A : E_B^* \to E_{B^*A} = E_B^*/\phi_B^*(G_A)$
Compute $j(E_{B^*A})$

Attacker:
$< [m_b^*]\phi_A(P_B) + [n_b^*]\phi_A(Q_B) >=$

$= \phi_A(G_B^*)$
$\phi_A^* : E_A \rightarrow E_{AB^*} = E_A/\phi_A(G_B^*)$
Compute $j(E_{AB^*}) = j(E_{B^*A})$

B:
$< [m_b]\phi_A^*(P_B)+[n_b]\phi_A^*(Q_B) >=$
$= \phi_A^*(G_B)$
$\phi_B' : E_A^* \rightarrow E_{A^*B} = E_A^*/\phi_A^*(G_B)$
Compute $j(E_{A^*B})$

Attacker:
$< [m_a^*]\phi_B(P_A) + [n_a^*]\phi_B(Q_A) >=$
$= \phi_B(G_A^*)$
$\phi_B^* : E_B \rightarrow E_{BA^*} = E_B/\phi_B(G_A^*)$
Compute $j(E_{BA^*}) = j(E_{A^*B})$

To overcome MITM attack, we propose a simple method;
First setup general public points P_C, Q_C,
Append public key of A, $\phi_A(P_C), \phi_A(Q_C)$
Append public key of B, $\phi_B(P_C), \phi_B(Q_C)$
By public key of B, after exchange of parameters, user A computes $\phi_A'(\phi_B(P_C)), \phi_A'(\phi_B(Q_C))$
By public key of A, after exchange of parameters, user B computes $\phi_B'(\phi_A(P_C)), \phi_B'(\phi_A(Q_C))$
Shared secret key is hash of $j(E_{BA})||\phi_A'(\phi_B(P_C))||\phi_A'(\phi_B(Q_C))$ which is equal to
$j(E_{AB})||\phi_B'(\phi_A(P_C))||\phi_B'(\phi_A(Q_C))$

6.2.2.3 Public Key Encryption Scheme

Jao and De Feo [75] has adapted Supersingular Isogeny Diffie-Hellman (SIDH) key exchange to a public key encryption scheme like adaptation of Diffie-Hellman key exchange to Elgamal encryption.

Scheme 2: Jao-De Feo Public Key Encryption Scheme

Setup:
Choose prime of the form $2^a.3^b.f - 1$, supersingular elliptic curve E over the field $\mathbb{F}_{p^2}$, linearly independent points $P_A, Q_A \in E[2^a]$ and $P_B, Q_B \in E[3^b]$. H_k be a keyed hash function from $\mathbb{F}_{p^2}$ to $(0,1)^w$
Key Generation:

1. User A picks random integers $m_a, n_a < 2^a$ (Not both divisible by 2).
2. User A computes $E_A = \phi_A(E), \phi_A(P_B), \phi_A(Q_B)$
3. Then user A chooses a random k as ephemeral key for hash. The public and private keys are the tuples $(E_A, \phi_A(P_B), \phi_A(Q_B), k)$, (m_a, n_a, k) respectively.

Encryption:

1. User B picks random integers $m_b, n_b < 3^b$ (Not both divisible by 3). Computes $E_B = \phi_B(E), \phi_B(P_A), \phi_B(Q_A)$ as above.

2. User B computes $h = H_k(j(E_{AB}))$, and $c = h \oplus m$

3. Output the tuple $(E_B, \phi_B(P_A), \phi_B(Q_A), c)$

Decryption:
User A computes $h = H_k(j(E_{AB}))$, and $m = h \oplus c$ by using $(E_B, \phi_B(P_A), \phi_B(Q_A), c)$ and private key (m_a, n_a, k).

6.2.2.4 Strong Designated Verifier Signature Scheme

In [80] a Strong Designated Verifier Signature Scheme was proposed based on Isogenies. In a Designated Verifier Signature Scheme, the signature can only be verified by intended verifier. And in a Strong Designated Verifier Signature Scheme this verification is done by verifier's private key. They combined the [75] key exchange scheme and Huang et al's [81] method to form Strong Designated Verifier Signature from a Diffie-Hellmann key exchange scheme.

Scheme 3: Strong Designated Verifier Signature Scheme

Setup:
Like in [75] key exchange scheme choose prime of the form $2^a.3^b.f - 1$, supersingular elliptic curve E over the field $\mathbb{F}_{p^2}$, linearly independent points $P_A, Q_A \in E[2^a]$ and $P_B, Q_B \in E[3^b]$. H_k be a keyed hash function from $\mathbb{F}_{p^2}$ to $(0,1)^w$
Key Extraction:

1. Signer picks random integers $m_s, n_s < 2^a$ (Not both divisible by 2).

2. Signer computes $E_S = \phi_S(E), \phi_S(P_B), \phi_S(Q_B)$ as public key. The private key is (m_s, n_s).

3. Similarly verifier picks random integers $m_v, n_v < 3^b$ (Not both divisible by 3) as private key.

4. Verifier computes $E_V = \phi_V(E), \phi_V(P_A), \phi_V(Q_A)$ as public key.

Sign:

1. Signer computes kernel $< [m_s]\phi_V(P_A) + [n_s]\phi_V(Q_A) >$ and corresponding isogeny ϕ'_S from E_S to E_{SV}.

2. The notation is miswritten here, ϕ'_S should be from E_V to E_{VS} as stated in [75]. Respectively for verifier.

3. Signature is $\sigma = H(m||j(E_{SV}))$.

Verification:

1. Verifier computes kernel $< [m_v]\phi_S(P_B) + [n_v]\phi_S(Q_B) >$ and corresponding isogeny ϕ'_V from E_V to E_{VS}.
2. Computes $\sigma' = H(m||j(E_{VS}))$.
3. Check if $\sigma = \sigma'$

6.2.2.5 Undeniable Signature Scheme

[78] used zero knowledge style verification. Any user can verify an undeniable signature, only by interaction with the signer. There is also a disawoval protocol for the signer to interactively prove when there is a forgery.

Scheme 4: Undeniable Signature Scheme

Setup:
Let p is a prime of the form $p_a^a.p_b^b.p_c^c.f \pm 1$. E is a supersingular elliptic curve over the field $\mathbb{F}_{p^2}$. Choose linearly independent points $P_A, Q_A \in E[p_a^a]$, $P_B, Q_B \in E[p_b^b]$ and $P_C, Q_C \in E[p_c^c]$ be bases for $E[p_a^a]$, $E[p_b^b]$ and $E[p_c^c]$ respectively.

Key Generation:

1. Signer picks random integers m_a, n_a. And computes $G_A =< [m_a]P_A + [n_a]Q_A >$,
2. By using Velu's formula signer then computes ϕ_A. And computes $E_A = \phi_A(E) = E/G_A$
3. The public key for signer is $E_A, \phi_A(P_C), \phi_A(Q_C)$ and the private key is m_a, n_a

Signing:

1. For signing a message M, signer computes the hash $h = H(M)$ and $G_M = P_B + [h]Q_B$ and computes the isogenies
2. $E_M = \phi_M(E) = E/G_M$
3. $E_{AM} = \phi_{M,AM}(E_M) = E_M/\phi_M(G_A)$
4. $E_{AM} = \phi_{A,AM}(E_A) = E_A/\phi_A(G_M)$
5. signature is E_{AM} and auxiliary points $\phi_{M,AM}(\phi_M(P_C))$ and $\phi_{M,AM}(\phi_M(Q_C))$.

Confirmation:

1. Signer picks random integers m_c, n_c. And computes $G_C =< [m_c]P_C + [n_c]Q_C >$, and also the curves and isogenies;

2. $E_C = \phi_C(E) = E/G_C$, $E_{MC} = E_M/\phi_M(G_C) = E_C/\phi_C(G_M)$,
3. $E_{AC} = E_A/\phi_A(G_C) = E_C/\phi_C(G_A)$ and $E_{AMC} = E_{MC}/\phi_{C,MC}(G_A)$
4. The signer sends $E_C, E_{AC}, E_{MC}, E_{AMC}$ and $ker(\phi_{C,MC})$ as the commitment.
5. $b \in 0, 1$ is randomly selected by verifier.
6. if $b = 0$ then signer outputs $ker(\phi_C)$ verifier computes $ker(\phi_{A,AC})$ and $\phi_{M,MC}$ and using the auxiliary points verifier computes $\phi_{AM,AMC}$.
7. verifier checks whether each isogeny maps between the corresponding commitment curves. The verifier also re-computes $\phi_{C,MC}$ and checks with commitment.
8. if $b = 1$ then signer outputs $ker(\phi_{C,AC})$. verifier computes $\phi_{MC,AMC}$ and $\phi_{AC,AMC}$ and checks whether each isogeny maps between the corresponding commitment curves.

Disavowal:
Let a third party sent a falsified signature (E_T, T_P, T_Q) to the signer where E_T is falsified curve E_{AM} and (T_P, T_Q) stands for auxiliary points $\phi_{M,AM}(\phi_M(P_C))$ and $\phi_{M,AM}(\phi_M(Q_C))$ respectively. Now the verifier should be able to compute E_{FC} and check that $E_{FC} \neq E_{AMC}$.

1. Signer picks random integers m_c, n_c. And computes $G_C =< [m_c]P_C + [n_c]Q_C >$, and also the following curves and isogenies;
2. $E_C = \phi_C(E) = E/G_C$, $E_{MC} = E_M/\phi_M(G_C) = E_C/\phi_C(G_M)$,
3. $E_{AC} = E_A/\phi_A(G_C) = E_C/\phi_C(G_A)$ and $E_{AMC} = E_{MC}/\phi_{C,MC}(G_A)$
4. The signer sends $E_C, E_{AC}, E_{MC}, E_{AMC}$ and $ker(\phi_{C,MC})$ as the commitment.
5. the verifier selects random $b \in 0, 1$
6. if $b = 0$ then signer outputs $ker(\phi_C)$ verifier computes $ker(\phi_{A,AC})$ and $\phi_{M,MC}$ and $\phi_T : E_{TC} = \phi_T(E_T) = E_T/ < [m_c]T_P + [n_c]T_Q >$.
7. verifier checks whether each isogeny maps between the corresponding commitment curves. The verifier also re-computes $\phi_{C,MC}$ and checks with commitment. And verifier also checks that $E_{FC} \neq E_{AMC}$.
8. if $b = 1$ then signer outputs $ker(\phi_{C,AC})$. verifier computes $\phi_{MC,AMC}$ and $\phi_{AC,AMC}$ and checks whether each isogeny maps between the corresponding commitment curves and map to E_{AMC}.

6.2.2.6 Undeniable Blind Signature Scheme

[82] They have extended Jao-Soukharev [78] scheme into an Undeniable Blind Signature scheme.

Scheme 5: Undeniable Blind Signature Scheme

Setup:
Let p is a prime of the form $p_a^a.p_b^b.p_c^c.p_d^d.f \pm 1$. E_0 is a supersingular elliptic curve over the field $\mathbb{F}_{p^2}$. Choose linearly independent points $P_A, Q_A \in E[p_a^a]$, $P_B, Q_B \in E[p_b^b]$, $P_C, Q_C \in E[p_c^c]$ and $P_D, Q_D \in E[p_d^d]$ be bases for $E_0[p_a^a]$, $E[p_b^b]$, $E[p_c^c]$ and $E[p_d^d]$ respectively. Finally choose a hash function $H : 0, 1* \rightarrow \mathbb{Z}/p_b^b\mathbb{Z}$.

Key Generation:

1. Signer picks random integers m_a, n_a. And computes $G_A =< [m_a]P_A + [n_a]Q_A >$,
2. By using Velu's formula signer then computes ϕ_A. And computes $E_A = \phi_A(E_0) = E_0/G_A$
3. The public key for signer is $E_A, \phi_A(P_C), \phi_A(Q_C)$ and the private key is m_a, n_a, G_A

Blinding:

1. To blind a message M, requester computes the hash $h = H(M)$ and $G_M =< P_B + [h]Q_B >$ and computes the isogeny ϕ_M
2. $E_M = \phi_M(E_0) = E_0/G_M$
3. And computes the points $\phi_M(P_A), \phi_M(Q_A), \phi_M(P_C), \phi_M(Q_C), \phi_M(P_R), \phi_M(Q_R)$,
4. To blind the curve E_M choose a random r and compute the isogeny; $E_{RM} = \phi_{M,RM}(E_M) = E_M/(\phi_M(P_R) + [r]\phi_M(Q_R))$ and the blinded curve E_{RM}.
5. To unblind the curve back, user must compute the dual isogeny $\hat{\phi}_{M,RM}$, first find a point $K \in E_M[p_d^d]$ of order p_d^d such that $K \notin Ker\phi_{M,RM}$ for example $K = \phi_M(Q_D)$, compute the point $\phi_{M,RM}(K) \in E_{RM}$. The dual isogeny $\hat{\phi}_{M,RM}$ is with the kernel $< \phi_{M,RM}(K) >$
6. To generate $E_{RM}[p_d^d]$ choose basis $P'_D, Q'_D \in E_{RM}$,
7. Find suitable $m, n \in \mathbb{Z}/p_d^d$ such that $\phi_{M,RM}(K) = [m]P'_D + [n]Q'_D$. This is done by solving discrete logarithm problem by using generalized Pohlig-Hellman algorithm given by Teske.
8. All the points $P'_A = \phi_{M,RM}(\phi_M(P_A))$, $Q'_A = \phi_{M,RM}(\phi_M(Q_A))$, $P'_C = \phi_{M,RM}(\phi_M(P_C))$, $Q'_C = \phi_{M,RM}(\phi_M(Q_C))$, P'_D, Q'_D and blinded curve E_{RM} are sent to signer.

Signing:

1. To sign a blinded curve E_{RM} with his private key m_a, n_a, signer computes $< [m_a]P'_A + [n_a]Q'_A >$ and computes the isogeny $\phi_{RM,ARM}$
2. $E_{ARM} = \phi_{RM,ARM}(E_{RM}) = E_{RM}/ < [m_a]P'_A + [n_a]Q'_A >$
3. And computes the points $\phi_{RM,ARM}(P'_C), \phi_{RM,ARM}(Q'_C), \phi_{RM,ARM}(P'_D), \phi_{RM,ARM}(Q'_D)$ and sends all values to requester.

Unblinding:

1. To unblind a blinded signature requester computes $\hat{\phi}_{AM,ARM}$, and the curve;
2. $E_{AM} = E_{ARM}/ < [m]\phi_{RM,ARM}(P'_D) + [n]\phi_{RM,ARM}(Q'_D) >$.
3. And computes the points $P_S = \hat{\phi}_{AM,ARM}(\phi_{RM,ARM}(P'_C)), Q_S = \hat{\phi}_{AM,ARM}(\phi_{RM,ARM}(Q'_C))$ The signature is $\sigma = (E_{AM}, P_S, Q_S)$.

Verification: The curve unblinded is isomorphic to undeniable signature generated in [78], thus confirmation and disavowal protocols are same as in 6.2.2.5.

6.2.2.7 Isogeny Based Signature Schemes

Jao-Venkanetasan [85] which is under Canadian Patent.In [82] they say that [85] speculate the use of hardness assumptions related to isogeny problems in constructing blind signature. But when we eximine it, they have proposed a signature scheme not blind, but dependant on pairings in verification phase. Only isogenies are used in signing phase.

Scheme 6: Signature Scheme 1

Setup and Key Generation:
Let E_1 be an elliptic curve defined over $\mathbb{F}_{ıı}$, E_2 be an elliptic curve defined over $\mathbb{F}_{ıı}^{\aleph}$ and ϕ be an isogeny from E_1 to E_2. Select random $P \in E_1$ and compute $Q = \phi(P)$ and publish P,Q as public key. The private key is dual isogeny $\hat{\phi}$. (isogeny ϕ should also be stated as private key).

Signing:
First choose a hash function $H : 0, 1* \rightarrow E_2[k]$. Signature is;
$\sum_{i=0}^{n-1} \pi^i \hat{\phi}(H(m))$
where the summation is done over elliptic curve E_1, π is the q^{th} power Frobenius map. Note that $\sum_{i=0}^{n-1} \pi^i$ is denoted by trace Tr, output signature $S \in E_1(F_q)$ and the Galois group of F_{q^n}/F_q is $1, \pi, ..., \pi^{n-1}$ so S is Galois invariant and defined over F_q.

Verification:
Given a message m, signature S and public key (P, Q), check whether;
$e_1(P, S) = \prod_{i=0}^{n-1} \pi^i e_2(Q, H(m))$
where e_1, e_2 denote Weil pairing on $E_1[k] and E_2[k]$ respectively. The correction this equation is as follows;
$e_1(P, S) = e_1(\sum_{i=0}^{n-1} \pi^i \hat{\phi}(H(m))) = \prod_{i=0}^{n-1} e_1(P, \pi^i \hat{\phi}(H(m))) = \prod_{i=0}^{n-1} \pi^i e_1(P, \hat{\phi}(H(m))) = \prod_{i=0}^{n-1} \pi^i e_2(\phi(P), H(m)) = \prod_{i=0}^{n-1} \pi^i e_2(Q, H(m))$
Verificaiton is using the following property of Weil pairing:
$e_n(S, \hat{\phi}(T)) = e_n(\phi(S), T)$, where $S \in E_1[n], T \in E_2[n]$

In [85] another signature scheme with multiple isogenies for different elliptic curves has been defined, to increase the strength of short signatures.

Scheme 7: Signature Scheme 1

Setup and Key Generation:
Let ϕ_i be isogenies from E to E_i. Select random $P \in E$ and compute $Q_i = \phi_i(P)$ and publish P, Q_i as public key. The private key is isogenies ϕ_i.

Signing:
Hash functions are defined as $H_i : 0, 1* \rightarrow E_i$. Signature is;
$\sum_{i=1}^{n} \hat{\phi}_i(H_i(m))$

Verification:
Given a message m, signature S and public key (P, Q_i), check whether;
$e(P, S) = \prod_{i=1}^{n} e(Q_i, H_i(m))$
The correction of this equation is as follows;
$e(P, S) = e(P, \sum_{i=1}^{n} \hat{\phi}_i(H_i(m))) = \prod_{i=1}^{n} e(P, \hat{\phi}_i(H_i(m))) = \prod_{i=1}^{n} e(Q_i, H_i(m))$

Galbraith-Petit-Silva Signature Schemes Based on Supersingular Isogeny Problems In [86] two signature schemes are described. The first one is a Fiat-Shamir transformation from interactive to non-interactive version of De Feo, Jao and Plut protocol. The second signature scheme relies on the problem of computing the endomorphism ring of a supersingular elliptic curve

---

Scheme 8: Signature Scheme 1

Setup and Key Generation:

1. Let p is a prime of the form $p_a^a.p_b^b.f \pm 1$.
2. Let E_0 be a supersingular elliptic curve with j-invariant j_0.
3. Let $R_2, S_2 \in E_0(\mathbb{F}_{p^2})[p_b^b]$.
4. Select a random primitive p_a^a-torsion point $P_1 \in E_0[p_a^a]$.
5. Cmpute the isogeny $\phi : E_1 = \phi(E_0) = E_0/ < P_1 >$ and j_1 is the j-invariant of E_1.
6. Set $R_2' = \phi(R_2), S_2' = \phi(S_2)$
7. With hash function H with t bits of output, the public key for signer is $p, j_0, j_1, R_2, S_2, R_2', S_2', H$ and the private key is P_1

Signing:

1. Select random integers $0 \leq \alpha_i < p_b^b$ for $i = 1, ..., t$.
2. Compute the isogeny $\omega_i : E_{2,i} = \omega_i(E_0)$ with j-invariant $j_{2,i} = j(E_{2,i})$ with kernel generated by $R_2 + [\alpha_i]S_2$.

3. Compute the isogeny $\omega_i' : E_{3,i} = \omega_i'(E_1)$ with j-invariant $j_{3,i} = j(E_{3,i})$ with kernel generated by $R_2' + [\alpha_i]S_2'$.
4. Compute $h = H(m, j_{2,1}, ..., j_{2,t}j_{3,1}, ..., j_{3,t})$.
5. For t bits of h, b_i for $i = 1, ..., t$
6. if $b_i = 0$ then set $z_i = \alpha_i$.
7. if $b_i = 1$ then compute $\omega_i(P_1)$ and set z_i to a representation of $j_{2,i}$ and the isogeny with kernel $\omega_i(P_1)$.
8. Send signature $\sigma = (h, z_1, ..., z_t)$

Verification:

1. First verifier recovers the parameters p, E_0, E_1.
2. Then verifier recomputes the j-invariants $j_{2,1}, ..., j_{2,t}j_{3,1}, ..., j_{3,t}$ by using the information z_i for $1 \leq i \leq t$.
3. if $b_i = 0$ then by using $z_i = \alpha_i$, computes the isogenies from E_0 and E_1, with kernels $< R_2 + [\alpha_i]S_2 >$ and $< R_2' + [\alpha_i]S_2' >$ respectively.
4. if $b_i = 1$ then z_i is a representation of $j_{2,i}$ and the description of isogeny from $E_{2,i}$ to $E_{3,i}$ to find $j_{3,i}$.
5. Verifier then computes $h' = H(m, j_{2,1}, ..., j_{2,t}j_{3,1}, ..., j_{3,t})$ with hsi own computed values and checks whether $h' = h$.

Yoo-Azarderakhsh-Jalali-Jao-Soukharev A Post-Quantum Digital Signature Scheme Based on Supersingular Isogenies In [88] a signature scheme is proposed which is conceptually identical to [86]. They again obtained it from De Feo, Jao and Plut protocol by a method of Unruh's non-interactive transformation.

Scheme 9: Digital Signature Scheme

Setup and Key Generation:

1. Let p is a prime of the form $p_a^a.p_b^b.f \pm 1$.
2. Let E be a supersingular curve over $\mathbb{F}_{p^2}$.
3. And P_B, Q_B be generators of $E[p_b^b]$.
4. Select a random p_a^a-torsion point S.

5. Compute the isogeny $\phi : E_S = \phi(E) = E/ < S >$.
6. Set public key $(E/ < S >, \phi(P_B), \phi(Q_B))$ and private key S.

Signing:

1. For i=1 to 2λ
2. Select a random p_b^b-torsion point S.
3. Compute the isogeny $\omega : E_R = \omega(E) = E/ < R >$.
4. Compute one of the isogenies $\omega' : E_{<R,S>} = \omega'(E_S)$ or $\phi' : E_{<R,S>} = \phi'(E_S)$.
5. Then call $E_1 = E_R$, $E_2 = E_{<R,S>}$ and $Commit_i = (E_1, E_2)$ and $Challenge_{i,0} = 0/1$ chosen randomly
6. if $Challenge_{i,0} = 0$ then set $(Response_{i,0}, Response_{i,1}) = ((R, \phi(R)), \omega(S))$.
7. if $Challenge_{i,0} = 1$ then set $(Response_{i,0}, Response_{i,1}) = (\omega(S), (R, \phi(R)))$.
8. $h_{i,j} = G(Response_{i,j})$
9. End For
10. $J_1||...||J_{2\lambda} = H(Pub.Key, m, (Commit_i)_i), (Challenge_{i,j})_{i,j}, (h_{i,j})_{i,j}$
11. Send signature $\sigma = ((Commit_i)_i), (Challenge_{i,j})_{i,j}, (h_{i,j})_{i,j}, (Response_{i,J_i})_i$

Verification:

1. First verifier computes $J_1||...||J_{2\lambda} = H(m, x, (Commit_i)_i), (Challenge_{i,j})_{i,j}, (h_{i,j})_{i,j}$
2. For i=1 to 2λ
3. verify $h_{i,J_i} = G(Response_{i,J_i})$
4. If $Challenge_{i,J_i} = 0$ then
5. Extract $(R, \phi(R))$ from $Response_{i,J_i}$.
6. Verify if R generates the kernel of isogeny $E \to E_1$ and has order p_b^b
7. Verify if $\phi(R)$ generates the kernel of isogeny $E/ < S > \to E_2$
8. Else $Challenge_{i,J_i} = 1$ then
9. Extract $\omega(S)$ from $Response_{i,J_i}$
10. Verify if $\omega(S)$ generates the kernel of isogeny $E_1 \to E_2$ and has order p_a^a
11. If it succeeds for all then verify the signature.

In [87] they eximine the The Security of Supersingular Isogeny Cryptosystems.

6.2.2.8 Comparison of Isogeny Based Signature Schemes

Although previously described signature schemes are proposed for different purposes, we try to give a comparison for parameter sizes. Computing parameter sizes is difficult because the sizes are dependant on the representation of elliptic curves, points, coefficients and also the security level needed (λ bits of post quantum security level may be accepted as common). For most of the proposed schemes there is not a description for representation except [88].

Table6.1: Comparison of parameter sizes

	Public Key	Private Key	Signature
SDVS [80]	1EC, 2P	2C	2λ(for hash)
Undeniable [78]	1EC, 2P	2C	1EC, 2P
UBSS [82]	1EC, 2P	2C	1EC, 6P for blinding, 4P for signing, 1EC, 2P for unblind
Jao [85]	2P	2C	1P
Galbraith [86]	2C, 2J, 4P	1P	2λ(2+1C+1J+1P)/2
Yoo [88]	1EC, 2P	1P	2λ(2EC+1+3λ+(2C+1P)/2)

EC: Elliptic Curve representation, which requires 1 field element size (12λ)
P : Elliptic Curve Point representation, which requires 1 field element size(12λ)
C : Coefficient representation for m,n or α, which requires half field element size (12λ)
J : J-Invariant representation, which requires 2 (both A and B) field elements size (24λ)
Calculations presented in the table are based on [88], which accepts to achieve λ bits of post quantum security level, base prime $p = p_a^a.p_b^b.f \pm 1$ should be 6λ bit length where each prime p_a and p_b of bit length 3λ and an element in base field $\mathbb{F}_{p^2}$ should be 6λ bit length. For supersingular elliptic curve representation one coefficient (A or B) and for points on curves one coordinate is accepted sufficient.

6.2.2.9 New VES Construction Based on Isogeny Signature Scheme

By using Jao-De-Feo key exchange based public key encryption scheme inside Yoo et al [88] isogeny based signature scheme we propose here a new verifiably encrypted signature construction. In this scheme conventional signing and verification of signature are same as [88], so only the Isog-VES, Isog-VES-Verify and Adjudication cases are described:

Scheme 10: Isogeny Based Verifiably Encrypted Signature Construction

---

Setup and Key Generation:

1. Let p is a prime of the form $p_a^a.p_b^b.f \pm 1$.

2. Let E be a supersingular curve over $\mathbb{F}_{p^2}$.
3. P_A, Q_A be generators of $E[p_a^a] and P_B, Q_B$ be generators of $E[p_b^b]$.
4. Select a random p_a^a-torsion point S and a random p_b^b-torsion point S_{Adj}.
5. Compute the isogenies $\phi : E_S = \phi(E) = E/ < S >$ and $\phi_{Adj} : E_{Adj} = \phi_{Adj}(E) = E/ < S_{Adj} >$.
6. Set public key of signer $(E/ < S >, \phi(P_B), \phi(Q_B))$ and private key of signer S.
7. Set public key of Adjudicator $(E/ < S_{Adj} >, \phi_{Adj}(P_A), \phi(Q_A))$ and private key of Adjudicator S_{Adj}.

VE-Sign:
Start with Jao-De-Feo encryption scheme, to produce key for encrypting all responses.

1. Signer computes $h = H_k(j(E_{S-Adj}))$, and $\phi(P_A), \phi(Q_A)$ with randomly selected k. We will use this key h for encryption in the Yoo scheme.

1. For i=1 to 2λ
2. Select a random p_b^b-torsion point S.
3. Compute the isogeny $\omega : E_R = \omega(E) = E/ < R >$.
4. Compute one of the isogenies $\omega' : E_{<R,S>} = \omega'(E_S)$ or $\phi' : E_{<R,S>} = \phi'(E_S)$.
5. Then call $E_1 = E_R$, $E_2 = E_{<R,S>}$ and $Commit_i = (E_1, E_2)$ and $Challenge_{i,0} = 0/1$ chosen randomly
6. if $Challenge_{i,0} = 0$ then set $(Response_{i,0}, Response_{i,1}) = ((R, \phi(R)), \omega(S))$.
7. if $Challenge_{i,0} = 1$ then set $(Response_{i,0}, Response_{i,1}) = (\omega(S), (R, \phi(R)))$.
8. $h_{i,j} = G(Response_{i,j})$
9. End For
10. First encrypt all responses with key h produced above, $c = h \oplus (Responses_{i,j})$
11. The ciphertext is the tuple $(E_S, \phi(P_A), \phi(Q_A), c, k)$
12. Add Adjudicator public key and ciphertext into hash input:
$J_1||...||J_{2\lambda} = H(Pub.Key, m, (Commit_i)_i), (Challenge_{i,j})_{i,j}, (h_{i,j})_{i,j}, PK_{Adj}, c)$
13. Add also the ciphertext c to signature;
$\sigma = ((Commit_i)_i), (Challenge_{i,j})_{i,j}, (h_{i,j})_{i,j}, (Response_{i,J_i})_i, c)$

By inserting public key of adjudicator and ciphertext into hash H, used responses are changed randomly.

VES-Verify:

1. First verifier computes
   $J_1||...||J_{2\lambda} = H(m, x, (Commit_i)_i), (Challenge_{i,j})_{i,j}, (h_{i,j})_{i,j}, PK_{Adj}, c)$

2. For i=1 to 2λ

3. verify $h_{i,J_i} = G(Response_{i,J_i})$

4. If $Challenge_{i,J_i} = 0$ then

5. Extract $(R, \phi(R))$ from $Response_{i,J_i}$.

6. Verify if R generates the kernel of isogeny $E \rightarrow E_1$ and has order p_b^b

7. Verify if $\phi(R)$ generates the kernel of isogeny $E/ < S > \rightarrow E_2$

8. Else $Challenge_{i,J_i} = 1$ then

9. Extract $\omega(S)$ from $Response_{i,J_i}$

10. Verify if $\omega(S)$ generates the kernel of isogeny $E_1 \rightarrow E_2$ and has order p_a^a

11. If it succeeds for all then verify the signature.

Adjudicate:

1. Adjudicator computes $h = H_k(j(E_{Adj-S}))$, where $j(E_{S-Adj}) = j(E_{Adj-S})$ decrypt c as; $Responses_{i,j} = h \oplus c$.

2. Continue like in the original scheme [88]
   $J_1||...||J_{2\lambda} = H(Pub.Key, m, (Commit_i)_i), (Challenge_{i,j})_{i,j}, (h_{i,j})_{i,j}$.

3. Insert the corresponding responses into the signature
   $\sigma = ((Commit_i)_i), (Challenge_{i,j})_{i,j}, (h_{i,j})_{i,j}, (Response_{i,J_i})_i$

6.3 Isogeny Pairing Groups

Koshiba and Takashima [83] has defined a new framework which they called Isogeny Pairing Groups. In this framework they propose to use isogenies with classical pairings to defend against quantum computing as a means of pre-challenge. Since pairing crypto is also subject to attack of quantum computing like other classical public key systems, their work helps pairing-based crypto to survive in post-quantum crypto world. Their framework proposed to bring only pre-challange resistance against post-quantum since their schemes use isogeny in the computation of master key/user keys, but other cryptographic computations are done with classical pairings.

6.3.1 Problems for Isogeny Pairing Groups

Problem 1: Isogeny Computational Bilinear Diffie Hellman (Isog-CBDH)
Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be cyclic symmetric pairing groups defined over two different elliptic curves (as in Section 2) and let ϕ be a randomly chosen isogeny from $\mathbb{G}_1$ to $\mathbb{G}_2$. Any ppt machine computes $e_1(\phi(g), \phi(h))$ only with a negligible probability given $g_1, \phi(g_1), g, \phi(h)$ for randomly generated $g_1, g, h \in \mathbb{G}_1$

Problem 2: Isogeny Decisional Bilinear Diffie Hellman (Isog-DBDH)
Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be cyclic symmetric pairing groups defined over two different elliptic curves (as in Section 2) and let ϕ be a randomly chosen isogeny from $\mathbb{G}_1$ to $\mathbb{G}_2$. Any ppt machine guesses whether $g_T = e_1(\phi(g), \phi(h))$ or a random element in $\mathbb{G}_T$ only with a negligible probability given $g_1, \phi(g_1), g, \phi(h), g_T$ for randomly generated $g_1, g, h \in \mathbb{G}_1$ and $g_T \in \mathbb{G}_T$

6.3.2 Isogenous Pairing Groups

By bringing together the trapdoor homomorphisms and binding with Isog-DBDH, Koshiba and Takashima [83] has proposed Isogenous Pairing Groups framework which has the compatibility property as follows; $e_1(g, h) = e_2(\phi(g), \phi(h))$ where $e_1(respectively e_2)$ is an efficiently computable pairing on $\mathbb{G}_1$ (respectively $\mathbb{G}_2$), on the curve $E_1(respectively E_2)$.
Compatiblity between pairing and isogeny is a key property for construction of IPG. The Weil pairing is compatible with isogenies as [84] states the following proposition: Proposition 1: For any $P, Q \in E_0[q]$ and any non-trivial isogeny $\phi : E_0 \to E_1$, then $e_{weil,1}(\phi(P), \phi(Q)) = e_{weil,0}(P, Q)^{deg\phi}$ where $e_{weil,1}$ is the Weil pairing on E_1 and respectively $e_{weil,0}$ is the Weil pairing on E_0.

Definition 1 Isogenous Pairing Groups : An instance of Isogenous Pairing Group is generated as follows; $Gen^{IPG}(1^\lambda, d) \xrightarrow{R} (pk^{IPG} := (< \mathbb{G}_t, \hat{\mathbb{G}}_t, g_t, \hat{g}_t, e_t >_{t\in[1,d]}, \mathbb{G}_T), sk^{IPG} := (\phi_t)_{t\in[1,d]})$ where $< \mathbb{G}_t, \hat{\mathbb{G}}_t, e_t, \mathbb{G}_T >$ are asymmetric pairing groups with pairings $e_t : \mathbb{G}_t \times \hat{\mathbb{G}}_t \to \mathbb{G}_T, g_t \in \mathbb{G}_t, \hat{g}_t \in \hat{\mathbb{G}}_t$ given by isogenies between different elliptic curves $\phi_t : \mathbb{G}_1 \to \mathbb{G}_{t+1}$ and $g_{t+1} = \phi_t(g_1)$.
Isogenous pairing groups staisfy the extension of abovementioned compatibility property as;
$e_1(g_1, \hat{g}_1) = e_t(g_t, \hat{g}_t) = e_t(\phi_{t-1}(g_t), \hat{g}_t)$

The following diagram summarizes the Isogenous Pairing Groups

$$
\begin{array}{ccc}
\mathbb{G}_1 \times \hat{\mathbb{G}}_1 & \xrightarrow{\quad e_1 \quad} & \mathbb{G}_T \; e_1(g_1, \hat{g}_1) = \\
\downarrow & & \uparrow \; e_2(g_2, \hat{g}_2) = \\
\xrightarrow{\phi_1} \mathbb{G}_2 \times \hat{\mathbb{G}}_2 & \xrightarrow{\quad e_2 \quad} & e_2(\phi_1(g_1), \hat{g}_2) = \\
\downarrow \; ... \quad ... \quad ... & & \uparrow \; e_d(g_d, \hat{g}_d) = \\
\xrightarrow{\phi_{d-1}} \mathbb{G}_d \times \hat{\mathbb{G}}_d & \xrightarrow{\quad e_d \quad} & e_d(\phi_{d-1}(g_d), \hat{g}_d) = g_T
\end{array}
$$

Here, $e_1(g_1, \hat{g}_1) = g_T$ is the non-trivial pairing value, points on $\mathbb{G}_t, \hat{\mathbb{G}}_t$ generate the group q-torsion points on tne t-th elliptic curve where $\mathbb{G}_t \neq \hat{\mathbb{G}}_t$

Definition 2 Isog-DBDH Assumption on IPG:
For $(pk^{IPG} := (< \mathbb{G}_t, \hat{\mathbb{G}}_t, g_t, \hat{g}_t, e_t >_{t\in[1,2]}, \mathbb{G}_T), sk^{IPG} := (\phi_1)) \xleftarrow{R} Gen^{IPG}(1^\lambda, d)$ and $\alpha, \beta, \gamma \xleftarrow{U}$

$\mathbb{Z}_q$ a classical ppt machine adversary receives χ_b, $b \xleftarrow{U} 0, 1$ that is defined by;
$\chi_0 := (pk^{IPG}, g_1^{\alpha}, \hat{g}_2^{\beta}, g_T^{\alpha\beta}$ and $\chi_1 := (pk^{IPG}, g_1^{\alpha}, \hat{g}_2^{\beta}, g_T^{\gamma}$
where $g_T := e_1(g_1, \hat{g}_1)$. Adversary outputs a guess bit b'. If b=b', adversary wins. For any ppt adversary the advantage of adversary is negligible in λ.
Next, qIsog-DBDH assumption on IPG is defined against quantum adversary similar way Isog-DBDH Assumption on IPG. And d-qIsog-DBDH assumption on IPG is defined against quantum adversary similar way qIsog-DBDH Assumption on IPG but it is given d isogenies $sk^{IPG} := (\phi_t)_{t\in[1,d]}$ with d points $< \hat{g}_t^{\beta} >_{t\in[1,d]}$ instead of one isogeny and one point.

6.3.2.1 Identity Based Encryption Scheme

Koshiba and Takashima [83] has proposed Anonymous Identity Based Encryption Scheme with isogenous pairing group groups which is secure against pre-challenge quantum adversaries.

Scheme 11: Anonymous Identity Based Encryption Scheme with IPG

Setup:
$(pk^{IPG} := (< \mathbb{G}_t, \hat{\mathbb{G}}_t, g_t, \hat{g}_t, e_t >_{t\in[1,2]}, \mathbb{G}_T), sk^{IPG} := (\phi_1)) \xleftarrow{R} Gen^{IPG}(1^{\lambda}, 1)$. A random hash function $H : \mathbb{F}_q \to \mathbb{G}_1$.

Key Generation:(sk^{IPG},ID)

1. $h_1 = H(ID) \in \mathbb{G}_1$.
2. $h_2 = \phi_1(h_1)$

Return secret key for user ID $sk_{ID} = h_2$.

Encryption:(pk^{IPG},m,ID)

1. $h_1 = H(ID) \in \mathbb{G}_1$.
2. $\alpha \xleftarrow{R} \mathbb{F}_q^*$
3. $c = \hat{g}_2^{\alpha}$
4. $z = e_1(h_1, \hat{g}_1)^{\alpha}$
5. $c_T = z.m$
6. The ciphertext is the tuple $CT_{ID} =< c, c_T >$

Decryption:($pk^{IPG}, sk_{ID} = h_2, CT_{ID} =< c, c_T >$)

1. $z' = e_2(h_2, c)$
2. $m' = c_T.z'^{-1}$ Return m'.

The correction of decryption is dependant on the compatibility property $e_1(h_1, \hat{g}_1) = e_2(\phi_1(h_1), \hat{g}_2)$.

Remark
The drawbacks of their scheme are; They use the compatibility of isogeny and pairing as $e_1(P_1, \hat{Q}_1) = e_1(\phi(P_0), \hat{Q}_1) = e_0(P_0, \hat{Q}_0)$ where ϕ is defined as an isogeny $E_1 = \phi(E_0)$. But there is not any relation between $\hat{Q}_1$ and $\hat{Q}_0$ which holds the equation. They said in Remark 2 case 2 that ϕ_t is defined from the rank two torsion group $\mathbb{G}_0 \oplus \hat{\mathbb{G}}_0$ to $\mathbb{G}_t \oplus \hat{\mathbb{G}}_t$ such that $\mathbb{G}_t = \phi_t(\mathbb{G}_0)$ and $\hat{\mathbb{G}}_t = \phi_t(\hat{\mathbb{G}}_0)$. However ϕ_t is enough to be defined only on $\mathbb{G}_0$ for their schemes, they did not defined ϕ_t over $\hat{\mathbb{G}}_0$.

6.3.2.2 New Identity Based Signature Scheme

Here we convert the Identity Based Signature Scheme of K.G.Paterson [89] to isogenous pairing group framework which is secure against pre-challenge quantum adversaries. The scheme is definde in additive format so we follow the original work.

Scheme 12: Identity Based Signature Scheme with IPG

Setup:
$(pk^{IPG} := (< \mathbb{G}_t, \hat{\mathbb{G}}_t, g_t, \hat{g}_t, e_t >_{t\in[1,2]}, \mathbb{G}_T), sk^{IPG} := (\phi_1)) \xleftarrow{R} Gen^{IPG}(1^\lambda, 1)$. Random hash functions $H_1 : \mathbb{F}_q \to \mathbb{G}_1$, $H_2 : ((0,1)^* || \mathbb{Z}_q^*) \to \mathbb{Z}_q^*$, $H_3 : \hat{G}_1 \to \mathbb{Z}_q^*$. And the points $P_1 \in \mathbb{G}_1, P_2 \in \mathbb{G}_2, \hat{P}_1 \in \hat{\mathbb{G}}_1, \hat{P}_2 \in \hat{\mathbb{G}}_2$ are defined globally.

Key Generation:
(sk^{IPG},ID)

1. $h_1 = H_1(ID) \in \mathbb{G}_1$.
2. $h_2 = \phi_1(h_1)$

Return secret key for user ID $sk_{ID} = h_2$.

Signature:
(sk^{IPG},m,ID)

1. $h_1 = H_1(ID) \in \mathbb{G}_1$.
2. Choose random $k, r \in \mathbb{Z}_q^*$
3. $R = k\hat{P}_2$
4. $S = k^{-1}(H_2(m, r)P_2 + H_3(R)h_2)$
5. The signature is the tuple $\sigma =< R, S, r >$

Verification:
($pk^{IPG}, \sigma =< R, S, r >$) verify if;

- $e_2(S, R) = e_1(P_1, \hat{P}_1)^{H_2(m,r)}.e_1(h_1, \hat{P}_1)^{H_3(R)}$

The correction of verification is dependant on the compatibility property $e_1(h_1, \hat{g}_1) = e_2(\phi_1(h_1), \hat{g}_2)$. And if we replace S,R in the equation $e_2(S, R) = e_2(k^{-1}(H_2(m, r)P_2 + H_3(R)h_2), k\hat{P}_2) = e_2((H_2(m, r)P_2 + H_3(R)h_2), \hat{P}_2) = e_2((H_2(m, r)P_2, \hat{P}_2).e_2(H_3(R)h_2, \hat{P}_2) = e_2(P_2, \hat{P}_2)^{H_2(m,r)}.e_2(h_2, \hat{P}_2)^{H_3(R)} = e_1(P_1, \hat{P}_1)^{H_2(m,r)}.e_1(h_1, \hat{P}_1)^{H_3(R)}$, then verification works.

Security:
Security is considered for two cases; classical and quantum random oracle model. For the first case our scheme is very similar to [89]. By contrast; in our scheme verification is done by pairing computation on the isogenius curve on the left hand side of the equation ($e_2(S, R)$), secret key generation is computed by isogeny of public key ($\phi_1(H_1(ID))$) instead of scalar multiplication (($s(H_1(ID))$)) in original version and hash function $H_2(m)$ is changed as $H_2(m, r)$ where r is a random number in $\mathbb{Z}_q^*$. The second change is the main reason to resist against quantum adversaries which changes the assumption of security rely on isogenies instead of discrete logarithm on elliptic curves. The first change is related with second change and using IPG as building block. Last change is needed because of security proof in quantum random oracle model.

Thus the security notion is also similar to original work in classical ROM. If there exists an adversary $\mathcal{A}$with advantage ϵ our scheme then there will be an adversary $\mathcal{B}$against El-Gamal signature scheme who forges with advantage ϵ/cn_1. Where n_1 is the number of H_1 queries made by adversary $\mathcal{A}$, H_1 is the random function instead of hash H_1, c is the constant.
For the case of quantum random oracle model, we use the security definition of [90];

Theorem 6.3.1. *The proposed new signature scheme is EUF-qCMA secure in the quantum random oracle model where EUF-qCMA security is defined as in [90].*

Proof of Sketch: This proof relies on the Theorem 5 defined in [90] which states if the classical signature scheme is secure enough (UUF-RMA Universally Unforgeability under a random message attack) then a signature scheme constructed like in Construction 4 in [90] will be EUF-qCMA secure. When we compare our scheme with construction 4; selecting k as random, regarding H_3 as randomness s used by signing, and $H_(m, r)$ can be replaced by $H(m, r)$ in the construction. Then our scheme is EUF-qCMA secure under the q-Isog-DBDH assumption.

6.3.2.3 New Identity Based Verifiably Encrypted Signature Scheme

Here we propose a new verifiably encrypted signature scheme which is modification of the scheme proposed in previous section. This scheme is also based on isogenous pairing group framework which is secure against pre-challenge quantum adversaries.

Scheme 13: Identity Based Verifiably Encrypted Signature Scheme with IPG

Setup:
$(pk^{IPG} := (< \mathbb{G}_t, \hat{\mathbb{G}}_t, g_t, \hat{g}_t, e_t >_{t\in[1,2]}, \mathbb{G}_T), sk^{IPG} := (\phi_1)) \xleftarrow{R} Gen^{IPG}(1^\lambda, 1)$. Random hash functions $H_1 : \mathbb{F}_q \to \mathbb{G}_1$, $H_2 : 0, 1^* \to \mathbb{Z}_q^*$, $H_3 : \hat{G}_1 \to \mathbb{Z}_q^*$. And the points $P_1, P_1' \in \mathbb{G}_1, P_2, P_2' \in \mathbb{G}_2, \hat{P}_1 \in \hat{\mathbb{G}}_1, \hat{P}_2 \in \hat{\mathbb{G}}_2$ are defined globally.

Key Generation:
(sk^{IPG},ID)

1. $h_1 = H_1(ID) \in \mathbb{G}_1$.
2. $h_2 = \phi_1(h_1)$

Return secret key for user ID $sk_{ID} = h_2$.

Signature:
(sk^{IPG},m,ID)

1. $h_1 = H_1(ID) \in \mathbb{G}_1$.
2. Choose random $\alpha \in \mathbb{Z}_q^*$.
3. Compute $k = e_1(h_1, \hat{P}_1)^{\alpha}$
4. $R = k\hat{P}_2$
5. $S = k^{-1}(H_2(m)P_2 + H_3(R)h_2)$
6. The signature is the pair $\sigma =< R, S >$

Verification:
($pk^{IPG}, \sigma =< R, S >$) verify if;

- $e_2(S, R) = e_1(P_1, \hat{P}_1)^{H_2(m)}.e_1(h_1, \hat{P}_1)^{H_3(R)}$

VE-Sign:
(sk^{IPG},m,ID)

1. $h_1 = H_1(ID) \in \mathbb{G}_1$.
2. Choose random $\alpha \in \mathbb{Z}_q^*$.
3. Compute $k = e_1(h_1, \hat{P}_1)^{\alpha}$
4. $R = k\hat{P}_2$
5. $R_2 = \alpha\hat{P}_2$
6. $S = k^{-1}(H_2(m)P_2 + H_3(R)h_2 + H_3(R_2)P_2')$
7. The VE-signature is the tuple $\sigma =< R, R_2, S >$

VE-Sign-Verification:
($pk^{IPG}, \sigma =< R, R_2, S >$) verify if;

- $e_2(S, R) = e_1(P_1, \hat{P}_1)^{H_2(m)}.e_1(h_1, \hat{P}_1)^{H_3(R)}.e_1(P_1', \hat{P}_1)^{H_3(R_2)}$

Adjudication:
$(pk^{IPG}, \sigma =< R, R_2, S >)$;

- Compute $k = e_2(\phi_1(h_1), R_2)$.
- Compute $S' = S - k^{-1}H_3(R_2)P'_2$, adjudicated signature is the pair $< R, S' = S >$.

The correction of verification is dependant on the compatibility property $e_1(h_1, \hat{g}_1) = e_2(\phi_1(h_1), \hat{g}_2)$.
And if we replace R_2 in the equation; $e_2(\phi_1(h_1), R_2) = e_2(\phi_1(h_1), \alpha\hat{P}_2) = e_1(h_1, \alpha\hat{P}_1) = e_1(h_1, \hat{P}_1)^\alpha = k$ adjudication works.

REFERENCES

[1] R. Kissel, *Glossary of Key Information Security Terms*, NISTIR 7298 Rev.2, `http://nvlpubs.nist.gov/nistpubs/ir/2013/NIST.IR.7298r2.pdf`, 2013.

[2] S. Kremer, O. Markowitch, J. Zhou*An Intensive Survey of Non-repudiation Protocols*, Computer Communications 25 (2002)1606-1621, 2002.

[3] J.L.F. Gomilla, J.A. Onieva, M. Payeras*Certified Electronic Mail: Propersties Revisited*, Computer & Security (2009) 1-13, 2009.

[4] R. Dutta, P. Barua, P.Sarkar *Pairing Based Cryptography: A Survey*, 2004.

[5] Ç. Çalık, Ö. Sever, H.M. Yıldırım, Z. Yüce *A Survey of Certified Electronic Mail Protocols* 4^{th} ISC Turkey, 2010.

[6] S. Akleylek, B.B. Kırlar, Ö. Sever, Z. Yüce, *Pairing Based Cryptography: A Survey* 3^{rd} ISC Turkey, 2008.

[7] S. Akleylek, B.B. Kırlar, Ö. Sever, Z. Yüce *Arithmetic on Pairing Friendly Fields* 3^{rd} ISC Turkey, 2008.

[8] S. Akleylek, B.B. Kırlar, Ö. Sever, Z. Yüce, *A New Short Signature Scheme with Random Oracle From Bilinear Pairings* Journal Of Telecommunications and Information Technology, 2011.

[9] Ö. Sever, E. Akyıldız, *Hybrid Non-Repudiation Protocol with Pairing Based Cryptography* 3rd ISDFS, 9-12 May 2015.

[10] Ö. Sever, E. Akyıldız, *Improved Contract Signing Protocol Based on Certificateless Hybrid Verifiably Encrypted Signature Scheme* 8^{th} ISC Turkey, 2015.

[11] Ö. Sever, *Verifiably Encrypted Signcryption Scheme Based on Pairings* International Journal of Information Security Science, IJISS, Vol.6, No.1, pp.1-10, 2017.

[12] M.Joye and G.Neven, *Identity-Based Cryptography* IOS Press, 2009.

[13] C. Galdi, R. Giordano *Certified email with temporal authentication: An improved optimistic protocol* Proceedings of International Conference on Trust and Privacy in Digital Business (TrustBus04), LNCS, vol.3184, Springer, Berlin, 2004, pp.181-190.

[14] R. Oppliger, P. Stadlin *A certified mail system (CMS) for the Internet*, Comput.Commun.27 2004 1229-1235.

[15] M. Franklin, G. Price *A comparison between traditional Public Key Infrastructures and Identity-Based Cryptography*, 2002.

[16] D. Boneh, M. Franklin *Identity Based Encryption from Weil Pairing.* SIAM J.of Computing Vol.32 No.3, 2003, Extended Abstract in Crypto 2001.

[17] A. Boldyreva, *Efficient Threshold Signature, Multisignature and Blind Signature Schemes Based on the Gap-Diffie-Hellman-Group Signature Scheme.* PKC 2003,LNCS 2139, pp.31-46 Springer-Verlag 2003.

[18] D. Boneh, B. Lynn, H. Shacham. *Short Signatures from the Weil Pairing.* in Proceedings of Asiacrypt 2001.

[19] F. Zhang, K. Kim. *ID-Based Blind Signature and Ring Signature from Pairings.* Advances in Cryptology in AsiaCrypt 2002, LNCS Vol.2510, Springer-Verlag, 2002.

[20] C. Cocks and R.G.E. Pinch, *Identity-based cryptosystems based on the Weil pairing*, Unpublished manuscript, 2001.

[21] M. Scott, P.S.L.M. Barreto, *Generating more MNT elliptic curves.* Des. Codes Cryptography 38,209-217, 2006.

[22] F. Brezing, A. Weng, *Elliptic curves suitable for pairing based cryptography* Des. Codes Cryptography 37,133-141, 2005.

[23] F. Zhang, R. Safavi-Naini, W. Susilo. *Efficient Verifiably Encrypted Signature and Partially Blind Signature from Bilinear Pairings.* In Proceedings of IndoCrypt 2003, Springer-Verlag, 2003.

[24] F. Hess, *Efficient Identity Based Signature Schemes Based on Pairings*, SAC 2002, LNCS 2595 Springer Verlag, 2000.

[25] J.A. Onieva, J. Zhou and J. Lopez *Multi-Party Non-Repudiation: A Survey* ACM Computing Surveys, 2008.

[26] C. Galdi, R. Giordano *Certified E-mail with temporal authentication: An improved optimistic protocol* LNCS Vol.3184, 2004.

[27] A. Bahreman and J. D. Tygar. *Certified Electronic Mail*, In Proceedings of the Internet Society Symposium on Network and Distributed System Security, pages 3-19, San Diego, CA, February 1994.

[28] S. Cimato, C. Galdi, R. Giordina, B. Masucci, G. Tomasco, *Design and Implementation of an Inline Certified E-mail Service*, CANS 2005, LNCS 3810, pp.186-199, 2005.

[29] J. Zhou, D. Gollman, 1996, *A fair non-repudiation protocol*, In Proceedings of the 1996 IEEE Symposium on Security and Privacy (May 06 - 08, 1996). SP. IEEE Computer Society, Washington, DC, 55.

[30] N. Zhang, Q. Shi, *Achieving non-repudiation of receipt*, The Computer Journal, 39(10):844–853, 1996.

[31] Rolf Oppliger, Peter Stadlin, *A certified mail system (CMS) for the Internet*, Computer Communications 27(13): 1229-123 (2004)

[32] M. Shao, G. Wang, J. Zhou, 2006, *Some common attacks against certified email protocols and the countermeasures*, Comput. Commun. 29, 15 (Sep. 2006), 2759-2769. DOI= http://dx.doi.org/10.1016/j.comcom.2005.10.027

[33] R. H. Deng, L. Gong, A. Lazar, and W. Wang. *Practical protocols for certified electronic mail*, Journal of Network and System Management, 4(3):279–297, 1996.

[34] N. Asokan, M. Schunter, M. Waidner, *Optimistic Protocols for Fair Exchange*, ACM Conference on Computer and Communications Security, 7-17, 1997.

[35] F. Bao, R. H. Deng, W. Mao, *Efficient and Practical Fair Exchange Protocols with Off-Line TTP*, IEEE Symposium on Security and Privacy 1998: 77-85

[36] Aruba Posta Elettronica Certificata, available at http://www.pec.it/.

[37] G. Ateniese, *Verifiable Encryption of Digital Signatures and Applications*, ACM Transactions on Information and System Security, Vol. 7, No. 1, February 2004, Pages 1-20.

[38] T. Coffey, P. Saidha, 1996. *Non-repudiation with mandatory proof of receipt* SIGCOMM Comput. Commun. Rev. 26, 1 (Jan. 1996), 6-17. DOI= http://doi.acm.org/10.1145/232335.232338

[39] S. Goldwasser, S. Micali and R. Rivest, *A digital signature scheme secure against adaptive chosen-message attacks*, SIAM Journal of Computing, vol. 17, no. 2, pp. 281-308, 1988.

[40] M. Bellare and P. Rogaway, *The exact security of digital signatures - How to sign with RSA and Rabin*, Eurocrypt 1996, Lecture Notes in Computer Science, vol. 1070, Springer-Verlag, pp. 399-416, 1996.

[41] S. Mitsunari, R. Sakai and M. Kasahara, *A new traitor tracing*, IEICE Trans. Fundamentals, vol. E85-A, no. 2, pp. 481-484, 2002.

[42] A. Joux *One Round Protocol for Tripartite Diffie Hellman* LNCS Vol.1838, 2000.

[43] S.S. Al-Riyami, K.G. Paterson *Certificateless Public Key Cryptography* AsiaCrypt 2003.

[44] C. Bamboriya, S.R. Yadav *A Survey of Different Contract Signing Protocols*, Ijetae V.1, I:4, January 2014.

[45] L. Chen, C. Gu *Optimistic Contract Signing Protocol Based on Hybrid Verifiably Encrypted Signature* Advances in Information Sciences and Service Sciences(AISS) V.4, N:12, July 2012.

[46] I. Blake, G. Seroussi, N. Smart *Advances in Elliptic Curves in Cryptography* Number 317 in London Mathematical Society Lecture Note Series. Cambridge University Press. ISBN 0-521-60415-X, 2005.

[47] S.D. Galbraith, K.G. Paterson, N.P. Smart *Pairings for Cryptographers* Discrete Applied Mathematics, Volume 156, Issue 16, 2008, Pages 3113-3121, ISSN 0166-218X, Also available Cryptology ePrint Archive, Report 2006/165.

[48] E.R. Verheul *Evidence that XTR is more secure than supersingular elliptic curve cryptosystems.* in EuroCrypt 2001, 195-210.

[49] R. Barbulescu, P. Gaudry, A. Joux, E. Tomme *A Quasi-polynomial Algorithm for Discrete Logarithm in Finite Fields of Small Characteristic* in EuroCrypt 2014.

[50] R. Granger, T. Kleinjung, J. Zumbragel *Breaking 128 bit Secure Supersingular Binary Curves* Advances in Cryptology - CRYPTO 2014, pp. 126-145.

[51] Y. Han, X. Gui, *Multi-Recipient Signcryption for Secure Wireless Group Communication*, 2009 4th IEEE Conference on Industrial Electronics and Applications, Xian, 2009, pp. 161-165.

[52] S.S.D. Selvi, S.S. Vivek, C.P. Rangan, *Identity Based Public Verifiable Signcryption Scheme* In: Heng SH., Kurosawa K. (eds) Provable Security. ProvSec 2010. Lecture Notes in Computer Science, vol 6402. Springer, Berlin, Heidelberg, 2010.

[53] US Legal Definition *http://definitions.uslegal.com/e/e-contract/*

[54] *https://www.e-contract.be/*

[55] *http://www.signable.co.uk/legal*

[56] *https://www.ctmecontracts.com/eContracts/wp/index.htm*

[57] *http://www.telefonica.com/en/about_telefonica/html/suppliers/soluciones/econtracts.shtml*

[58] R. Oppliger, P. Stadlin *A certified mail system (CMS) for the Internet*, Comput.Commun.27 2004 1229-1235.

[59] A. Menezes, T. Okamoto , S. Vanstone *Reducing elliptic curve logarithms to logarithms in a finite field*, The Annual ACM Symposium on Theory of Computing (STOC), ACM Press, pp. 8089, 1991.

[60] A. Menezes, S. Vanstone *Isomorphism classes of elliptic curves over finite fields of characteristic 2*, Util.Math.38 , 135-153, 1990.

[61] F. Morain *Classes disomorphismes des courbes elliptiques supersingulieres en caracteristique greater than and equal to 3*, Util.Math.52 , 241-253, 1997.

[62] S. Lang *Elliptic Functions*, Springer, Berlin, 1987.

[63] A. Miyaji, M. Nakabayashi, S. Takano *New explicit conditions of elliptic curve traces for FR-reduction*, IEICE Trans.Funda. E84-A(5),1234-1243, 2001.

[64] D. Freeman *Constructing pairing-friendly elliptic curves with embedding degree 10*, Algorithmic Number Theory Symposium - ANTS VII. Lecture Notes in Computer Science, Vol.4076 pp.452-465,Springer, 2006.

[65] D. Freeman, M. Scott, E. Teske *A Taxonomy of Pairing-Friendly Elliptic Curves* J.of Cryptology 23, pp.224-280, 2010.

[66] A.Nenadic, N.Zhang, B.Cheetham, C.Goble *RSA Based Certified Delivery of E-Goods Using Verifiable and Recoverable Signature Encryption* Journal of Computer Science vol.11, no.1(2005), 175-192, 2005.

[67] A.Kupcu, A.Lysyanskaya: *Usable optimistic fair exchange* In:Pieprzyk,J.(ed.)CT-RSA 2010.LNCS,vol.5985,pp.252-267.Springer,Heidelberg(2010), http://eprint.iacr.org/2008/431

[68] A.M.Alaraj, *Optimizing One Fair Document Exchange Protocol* International Journal of Network Security & Its Applications (IJNSA), Vol.4, No.1, January 2012.

[69] J.A. Garay, M. Jakobsson, P. MacKenzie, *Abuse-Free Optimistic Contract Signing*, In: Wiener M. (eds) Advances in Cryptology, CRYPTO 1999. Lecture Notes in Computer Science, vol 1666. Springer, Berlin, Heidelberg.

[70] D.Boneh, C.Gentry, B.Lynn, and H.Shacham, *Aggregate and Verifiably Encrypted Signatures from Bilinear Maps*, Advances in Cryptology - Eurocrypt 2003 ,LNCS, vol.2656, Springer, 2003, pp.416432.

[71] Y. Zheng, *Digital Signcryption or How to Achieve Cost, Advances in Cryptology - CRYPTO97* (B.S.KaliskiJr.,ed.),LNCS,vol.1294,Springer,1997,pp.165.179

[72] I.Jeong,H.Jeong,H.Rhee,D.Lee,andJ.Lim, *Provably Secure Encrypt-then-Sign Composition in Hybrid Signcryption*, Information security and cryptology ICISC 2002, pp.16.34.

[73] P.W. Shor, *Polynomialtime algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAMJ.Comput.26(5),1484 1509, 1997.

[74] J.M. Couveignes, *Hard homogeneous spaces*, Cryptology ePrint Archive, Report 2006/291, 2006.

[75] D. Jao, L.De Feo, *Towards quantum resistant cryptosystems from supersingular elliptic curve isogenies*, PQCrypto 2011 Proceedings, 2011,pp.19-34, 2011.

[76] A. Rostovtsev, A. Stolbunov, *Public key cryptosystem based on isogenies*, Cryptology ePrint Archive, Report 2006/145, 2006.

[77] A. Childs, D. Jao,V. Soukharev, *Constructing elliptic curve isogenies in quantum subexponential time*, J.Math.Crypt.8,1-29, 2014.

[78] D. Jao, V. Soukharev, *Isogeny based quantum resistant undeniable signatures*, In: PQCrypto 2014.pp.160-179, 2014.

[79] L.De Feo, D.Jao, J.Plut, *Towards quantum resistant cryptosystems from super-singular elliptic curve isogenies*, J.Math.Crypt.8(3),209-247(2014),preliminary version: IACR Cryptology eprint Archiv, 2011:506, 2011.

[80] S. Xi, H. Tian, Y. Wang, *Towards quantum resistant strong designated verifier signature from isogenies*, International Journal of Grid and Utility Computing 5 (2012),no.2,292-296, 2012.

[81] X.Huang, W.Susilo, Y.Mu and F.Zhang, *Short Designated Verifier Signature Scheme and Its Identity-based Variant*, International Journal of Network Security, Vol.6, No.1, pp.82-93, 2008.

[82] M.S. Srinath, V. Chandrasekaran, *Isogeny based Quantum resistant Undeniable Blind Signature Scheme*, IACR Cryptology eprint Archiv, 2016:148, 2016

[83] T. Koshiba, K. Takashima, *Pairing Cryptography Meets Isogeny: A New Framework of Isogenous Pairing Groups*, IACR Cryptology eprint Archiv, 2016:1138, 2016

[84] J. Silverman, *The Arithmetic of Elliptic Curves*, GTM Vol.106 Springer Verlag, 2009.

[85] D.Y. Jao and R. Venkatesan, *Use of Isogenies For Design of Cryptosystems*, December 24 2013. CA Patent 2,483,486.

[86] S.D.Galbraith, C.Petit and J.Silva, *Signature Schemes Based on Supersingular Isogeny Problems*, Cryptology eprint Archiv, 2016:1154, 2016

[87] S.D.Galbraith, C.Petit, B.Shani and Y.B.Ti, *On The Security of Supersingular Isogeny Cryptosystems*, AsiaCrypt 2016, Cryptology eprint Archiv, 2016:859, 2016

[88] Y.Yoo, R.Azarderakhsh, A.Jalali,D.Jao and V.Soukharev, *A Post-Quantum Digital Signature Scheme Based on Supersingular Isogenies*, IACR Cryptology eprint Archiv, 2017:186, 2017

[89] K.G.Paterson, *ID-Based Signatures from Pairings on Elliptic Curves*, Electronic Letters, Vol.38, No:18, pp.1025-1026, 2002, http://eprint.iacr.org/2002/004.

[90] D.Boneh, M.Zhandry, *Secure signatures and chosen cipher text security in a quantum computing world*, CRYPTO 2013, Part II. pp.361-379, 2013.

[91] D.Boneh, C.Gentry, B.Waters, *Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys*, CRYPTO 2005, LNCS, vol.3621,pp.258-275. Springer, Heidelberg(2005)

[92] R.Dubois, A.Guillevic, Le B.M.Sengelin *Improved Broadcast Encryption Scheme with Constant-Size Ciphertext*, In: Abdalla M., Lange T. (eds) Pairing-Based Cryptography - Pairing 2012. Pairing 2012. Lecture Notes in Computer Science, vol 7708. Springer, Berlin, Heidelberg. https://eprint.iacr.org/2012/370.pdf

[93] R.L. Rivest, A. Shamir, Y. Tauman, *How to leak a secret*, ASIACRYPT 2001. LNCS,vol.2248,pp.552-565. Springer,Heidelberg(2001)

[94] C.Y. Lin, T.C. Wu, *An identity-based ring signature scheme from bilinear pairings*, Cryptology ePrint Archive, Report 2003/117.

[95] X. Chen, F. Zhang, K. Kim, *A New ID-based Group Signature Scheme from Bilinear Pairings*, Cryptology ePrint Archive, Report 2003/116.

[96] D.Boneh, X.Boyen and H.Shacham, *Short group signatures*, in Advances in Cryptology-CRYPTO2004. Springer, 2004, pp.227-242.

[97] X. Boyen *Mesh signatures*, In: Naor,M.(ed.) EUROCRYPT 2007.LNCS, vol.4515,pp. 210-227.Springer,Heidelberg(2007) also available Cryptology ePrint Archive, Report 2007/094.

[98] X. Boyen, *Unconditionally Anonymous Ring and Mesh Signatures*, J. of Cryptology, October 2016, Volume 29, Issue 4, pp. 729-774.

[99] H.K. Maji, M. Prabhakaran, M. Rosulek *Attribute-Based Signatures*, In: Kiayias A. (eds) Topics in Cryptology. CT-RSA 2011. Lecture Notes in Computer Science, vol 6558. Springer, Berlin, Heidelberg.

[100] M. Barua, X. Liang, R. Lu, and X. Shen, *ESPAC: enabling security and patient-centric access control for eHealth in cloud computing*, International Journal of Security and Networks, vol. 6, no. 2/3, pp. 67-76, Nov. 2011.

[101] L. Ibraimi, M. Asim, and M. Petkovic, *Secure management of personal health records by applying attribute-based encryption*, 6th International Workshop on Wearable Micro and Nano Technologies for Personalized Health, pHealth09, Oslo, Norway, Jun. 2009, pp. 71-74.

[102] A. Lounis, A. Hadjidj, A. Bouabdallah, Y. Challal, *Secure and Scalable Cloud-based Architecture for e-Health Wireless Sensor Networks*, International Conference on Computer Communication Networks (ICCCN), Jul 2012, Munich, Germany. pp.1, 2012.

[103] Z. Wan, J. Liu and R. H. Deng, *HASBE: A Hierarchical Attribute-Based Solution for Flexible and Scalable Access Control in Cloud Computing*, in IEEE Transactions on Information Forensics and Security, vol. 7, no. 2, pp. 743-754, April 2012. doi: 10.1109/TIFS.2011.2172209

[104] J. Su, D. Cao, B. Zhao, X. Wang, I. You, *ePASS: An expressive attribute-based signature scheme with privacy and an unforgeability guarantee for the Internet of Things*, Future Generation Computer Systems, Volume 33, April 2014, Pages 11-18, ISSN 0167-739X, http://doi.org/10.1016/j.future.2013.10.016.

[105] B. Lynn *Phd Thesis, On the Implementation of Pairing-Based Cryptography*, https://crypto.stanford.edu/pbc/thesis.html.

[106] J. Bethencourt, A. Sahai, B. Waters *Ciphertext-policy attribute-based encryption*, Proceedings of the 2007 IEEE Symposium on Security and Privacy, SP'07, IEEE Computer Society, Washington, DC, USA (2007), pp. 321-334.

[107] A. De Caro and V. Iovino. 2011. *jPBC: Java pairing based cryptography*, In Proceedings of the 2011 IEEE Symposium on Computers and Communications (ISCC '11). IEEE Computer Society, Washington, DC, USA, 850-855. DOI=http://dx.doi.org/10.1109/ISCC.2011.5983948

[108] S.Y. Tan, S.H. Heng, B.M. Goi, *Java Implementation for Pairing-Based Cryptosystems*, In ICCSA'10, LNCS6019, Springer-Verlag.

[109] M. Stoegbauer, *Diploma thesis: Efficient algorithms for pairing-based cryptosystems*, 2004. http://www.cdc.informatik.tu-darmstadt.de/reports/reports/KP/Marcus Stoegbauer.diplom.pdf

[110] Y. Kawahara, T. Takagi, E. Okamoto, *Efficient implementation of Tate pairing on a mobile phone using Java*, CIS'06. LNCS(LNAI),vol.4456,pp.396-405. Springer,Heidelberg(2007)

[111] L. Malina, J. Hajny and V. Zeman, *Usability of pairing-based cryptography on smartphones*, 2015 38th International Conference on Telecommunications and Signal Processing (TSP), Prague, 2015, pp. 617-621.

[112] G.Grewal, R.Azarderakhsh, P.Longa, S.Hu and D.Jao, *Efficient implementation of bilinear pairings on arm processors*, in Selected Areas in Cryptography. Springer, 2013, pp.149-165.

[113] S.Canard, N.Desmoulins, J.Devigne and J.Traore, *On the implementation of a pairing-based cryptographic protocol in a constrained device*, in Pairing-Based Cryptography-Pairing 2012. Springer, 2013, pp.210-217.

[114] W. Liu, J. Liu, Q. Wu and B. Qin, *Android PBC: A Pairing Based Cryptography toolkit for android platform*, 2014 Communications Security Conference (CSC 2014), Beijing, 2014, pp. 1-6. doi: 10.1049/cp.2014.0739

[115] S.Y. Tan, C.S. Wong and H.H. Ng, *An Optimized Pairing-Based Cryptography Library for Android*, International Journal of Cryptology Research 6(1): 16 - 30, 2016.

[116] T. Hyla, I. El Fray, W. Mackow, J. Pejas, *Implementation of Pairing-Based Cryptographic Trust Infrastructure in Mobile Environment*, Przeglad Elektrotechniczny, 2015, R. 91, nr 2, pp.93-97.

[117] PBC Libraries: https://gist.github.com/artjomb/f2d720010506569d3a39

[118] M. Rückert, D. Schröder, *Security of Verifiably Encrypted Signatures and a Construction without Random Oracles*, Pairing 2009. Pairing 2009. Lecture Notes in Computer Science, vol 5671. Springer, Berlin, Heidelberg. Cryptology ePrint Archive, Report 2009/027.

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Sever, Ömer
Nationality: Turkish (TC)
Date and Place of Birth: 15 September 1980, Kilis
Marital Status: Married

EDUCATION

Degree	Institution	Year of Graduation
M.S.	Department of Cryptography, METU	2007
B.S.	Department of Computer Engineering, METU	2002

PROFESSIONAL EXPERIENCE

Year	Place	Enrollment
2002-	Turkish Navy	Officer

PUBLICATIONS

Ö. Sever, *Verifiably Encrypted Signcryption Scheme Based on Pairings* International Journal of Information Security Science, IJISS, Vol.6, No.1, pp.1-10, 2017.
Ö. Sever, E. Akyıldız, *Improved Contract Signing Protocol Based on Certificateless Hybrid Verifiably Encrypted Signature Scheme* 8^{th} ISC Turkey, 2015.
Ö. Sever, E. Akyıldız, *Hybrid Non-Repudiation Protocol with Pairing Based Cryptography* 3rd ISDFS, 9-12 May 2015.
S. Akleylek, B.B. Kırlar, Ö. Sever, Z. Yüce, *A New Short Signature Scheme with Random Oracle From Bilinear Pairings* Journal Of Telecommunications and Information Technology, 2011.
Ç. Çalık, Ö. Sever, H.M. Yıldırım, Z. Yüce *A Survey of Certified Electronic Mail Protocols* 4^{th} ISC Turkey, 2010.
S. Akleylek, B.B. Kırlar, Ö. Sever, Z. Yüce, *Pairing Based Cryptography: A Survey* 3^{rd} ISC Turkey, 2008.
S. Akleylek, B.B. Kırlar, Ö. Sever, Z. Yüce, *Arithmatic on Pairing Friendly Fields* 3^{rd} ISC Turkey, 2008.