DESIGNATED CONFIRMER SIGNATURES: MODELLING, DESIGN AND ANALYSIS

by

FUBIAO XIA

A thesis submitted to the

University of Birmingham

for the degree of

DOCTOR OF PHILOSOPHY

School of Computer Science

College of Engineering and Physical Science

University of Birmingham

January 2013

Abstract

Digital signatures are one of the most significant achievements of public-key cryptography and constitute a fundamental tool to ensure data authentication. However, the public verifiability of digital signatures may have undesirable consequences when manipulating sensitive and private information. Undeniable signatures, whose verification requires the cooperation of the signer in an interactive way, were invented due to such considerations. Whereafter, designated confirmer signatures (DCS) were introduced as an improved cryptographic primitive when the signer becomes unavailable in undeniable signatures.

This thesis is mainly devoted to the modelling, design and analysis of designated confirmer signatures. By exploiting the existing security notions, we theoretically analyse the relations among unimpersonation, invisibility, non-transferability and transcript-simulatability. To this end, we develop formal proofs to demonstrate the implications of those properties.

After providing the theoretical results related to the security model, we develop both concrete and generic DCS constructions that adapts to a full verification setting. On one hand, by supporting the signer's ability to disavow, we can achieve an efficient designated confirmer signatures by using bilinear maps, and such a construction is secure in the random oracle model under a new computational assumption, called Decisional Co-efficient Linear (D-co-L) assumption, whose intractability in pairing settings is analysed in the generic group model. The proposed scheme is constructed by encrypting Boneh, Lynn and Shacham's pairing based short signatures with signed ElGamal encryption. On the other hand, we build a generic transformation that is inspired by Gentry, Molnar, Ramzan's DCS scheme. The new generic DCS scheme is proved to be secure in the standard model, and can be implemented to obtain an efficient instantiation with a Persesen Commitment, a Camenisch and Shoup's Paillier-based encryption scheme and a Boneh, Lynn and Shacham's short signature scheme.

Acknowledgements

At the very beginning, I would like to thank my supervisors, Dr. Guilin Wang, and Dr. Volker Sorge. Without their guidance and support my thesis would not have been possible. In particular, I devote my special gratitude to Dr. Wang, for his great patience and long-term encouragement to my remote supervision in Australia in the last two and a half year, and to Dr. Sorge, for his valuable advices and help from being a remarkable lecturer in the fundamental cryptography in my Master's study.

Besides my supervisors, I'm deeply grateful to my thesis group members, Prof. Mark Ryan and Dr. Eike Ritter, for their insightful comments and challenging questions throughout my PhD study.

I also would like to thank all the staffs and PhD students in the Computer Security Reading Group. The group members and its weekly meetings have been a great source of feedback, and for years I'm benefiting and learning from the discussions. I am particularly grateful to: Myrto Arapinis, Tom Chothia, Eike Ritter, Mark Ryan, Ben Smyth, Shiwei Xu, Rehana Yasmine.

It gives me a great pleasure in acknowledging the support from the external researchers. They are either a collaborator of my previous research or the person educated me generously in cryptography and information security. They are: Liqun Chen, Wei Gao, Qiong Huang, Qi Xie, Rui Xue, Yunlei Zhao.

I am indebted to my many Ph.D colleagues for providing a stimulating and fun environment in which to learn and grow. I am especially grateful to Hasan Qunoo, Rodrigo Soares, Rehana Yasmin, Vivek Nallur, Shuo Wang, Haobo Fu, Guanzhou Lu, Shengdong He, Handing Wang.

I would like to thank my girlfriend Zhangqi, for her encouragement, understanding, and patience.

Finally, I am extremely appreciative of my family: Guoyu Xia and Xiaoqin Dai. Their love and support in both mental and financial aspects have gone a long way in helping me to achieve all that I have. They have taught me to always believe I shall never give up. To them I dedicate this thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

It is evident that information technology provides significant change to our daily life, especially the way people communicate. Communication systems have evolved from pigeons carrying messages to emails and instant messages that travel long distances within seconds. With the rapid development in information technology, the security issue in communication systems herewith get a great deal of attention. Fortunately, encryption technique has long been used to protect the confidentiality of messages, while digital signatures have been adopted to guarantee the authenticity and data integrity of messages in communication systems. The concept of digital signatures is initially introduced by Diffie and Hellman in the remarkable work [28], and later has been widely used in communication systems to significantly reduce the costs of conducting business over the Internet. As one of the most fundamental tools of public-key cryptography, digital signatures are generated by applying a mathematical formula or an algorithm, to scramble the information into a string of digits. Only the holder of the private key – **the signer** can produce such an "electronic autograph", and the recipient of the signature with the public key – **the verifier** can verify if the signature came from that individual. For messages distributed through a non-secure channel, a properly implemented digital signature gives the receiver reason to believe the message being sent by the claimed sender. In many scenarios, any change in the message after signature will invalidate that signature, which ensures the integrity of the signed data against tampering or corruption in the transmission.

As illuminated above, ordinary digital signatures are verifiable by anybody holding the signer's public key. Although the universal verifiability (or self-authentication) of digital signatures is very convenient in most applications, however, in some scenarios, the signer may hope the recipient of a signature would not be able to show its validity to other parties. For instance, a signature binding parties to a confidential agreement or a signature on documents carrying private or personal information, shall only be verified by specific verifiers. In these cases limiting the ability of third parties to verify a signature's validity is an important goal. This motivates the introduction of **undeniable signatures**. Chaum and van Antwerpen proposed the concept of undeniable signatures [23] that such a signature can only be verified with the collaboration of the legitimate signer. A distinctive feature of undeniable signatures is there exists a **disavow** protocol that allows to prevent the signer from denying a valid signature.

Undeniable signatures have various applications in cryptography such as licensing softwares, electronic voting and auctions. Considering a typical scenario in licensing softwares, for instance, software vendors might want to sign on their products to provide authenticity to their paying customers. Nevertheless, they strictly disallow dishonest users who have illegally duplicated their softwares to verify the validity of the signatures. Undeniable signature scheme plays an important role here as it allows only legitimate users to verify the validity of the signatures on the softwares.

However, for many practical applications, if the signer becomes unavailable, or refuses to cooperate, the recipient cannot make use of the signature. Due to this reason, **designated confirmer signatures** (DCS) are introduced by Chaum and van Antwerpen

[21] to solve this weakness, as an extension of undeniable signatures. In a designated confirmer signature scheme, the signer is still able to interactively verify the signature with the verifier. However, if the signer is unavailable, a semi-trusted third party called the designated confirmer can also confirm the (in)validity of an alleged signature by running some interactive protocols with the verifier. In general, such a verifier cannot transfer the signature's (in)validity to other parties by convincing them of the same fact. Furthermore, the designated confirmer can convert a designated confirmer signature into a standard signature when this is necessary, so that it becomes a publicly verifiable signature.

Perhaps the most convincing example demonstrating that designated confirmer signatures are better than undeniable signatures, is a job offer scenario. Alice is offered a job by Bob and wishes to receive a formal signed offer at some point, but Bob does not want Alice to show this offer to other potential employers. If Bob signs this offer using an undeniable signature, he may suffer from an embarrassment that Alice wants to expose this signed offer. In that case, Bob has no hope to deny that offer, and the most he can do is to refuse to cooperate. To solve this problem, Carol comes to the rescue. Suppose Bob signs this offer with a designated confirmer signature by using his own secret key and Carol's public key. He could simply convince Alice that the signed offer is legitimate, i.e., he proves to Alice he formed the signature in this way. Such a DCS is special in that it can also be verified directly by Carol, and its distribution is indistinguishable from a distribution that can be computed using only the public keys of Carol and Bob. Bob can assume that nobody can forge a signature for his public key, and that as long as Carol is honest no body learns that he signed an offer. Alice can safely assume that Bob cannot fool her, and that if Bob denies having signed an offer and Carol is honest, then Carol can prove to anybody that Bob is lying by convert that designated confirmer signature into an ordinary signature of Bob that can be verified by anybody.

1.1 Motivations

Since the invention of designated confirmer signatures, a number of schemes with various properties and different underlying mathematical problems have been developed. Although a considerable amount of work has been dedicated to the design of DCS schemes, all of the previous DCS schemes fail to support signer to disavow any invalid signatures. Therefore, the current concept of DCS has not yet fully inherited that functionality from undeniable signatures, as the latter does grant the signer the ability of disavowal. As a result, one motivation of this thesis is concerned with the design of designated confirmer signatures that fully support signer's verifiability, or more precisely, "DCS with full verification". The formal definition of this notion will be presented in Chapter 5.

Another motivation is the confusions of the security notions in the existing DCS models. As far as we know, In a DCS scheme, the signer's security only requires a security notion called unforgeability, which informally means no body except the signer can produce a valid signature on any unsigned message. However, to achieve the confirmer's security, several security notions were proposed in the literature, including unimpersonation [45, 64], invisibility [18, 32], non-transferability [18] and transcript-simulatability [38]. Intuitively, unimpersonation means no body can impersonate the confirmer to verify a DCS by running the confirmer's verification protocol; invisibility requires no body can see the validity of a DCS without the verification, in other words, an adversary who receives a DCS has no advantage than a random guess to find the signature is valid or not; non-transferability is relates to the verification protocols, which means one cannot get more information out of the verification protocols than whether a signature is valid or not; transcript-simulatability guarantees the confirmation or disavowal of a DCS should not be transferable, that is, any transcript of the verification protocols is simulatable. Naturally, one may raise such a question: "*can a DCS cryptosystem achieves the confirmer's security by satisfying only one or two of them?*" It is noticed that non-transferability and transcript-simulatability with different definitions but capture the similar security requirement, and hence there may exist an implication between these two notions. In addition, it seems also feasible to figure out an equivalence between three security notions, that is, a DCS cryptosystem achieves the security of transcript-simulatability, if and only if it achieves invisibility and non-transferability. Since the relations between these security notions are not clear and have never been formally discussed after their propositions, it is a fundamental question to study the equivalences/implications among these security notions.

1.2 Contributions and Results

Based on the above two motivations, the key contributions of this thesis can be summed up in two aspects. Firstly, from a modelling perspective, we prove that to achieve provable security, a DCS cryptosystem only requires transcript-simulatability or alternatively invisibility plus non-transferability. In other words, the security model proposed by Camenisch and Michels [18] and the security model proposed by Gentry et al [38] are equivalent under a proper assumption, that is if the verification protocols for DCS schemes are based on zero knowledge proofs. However, we also prove that unimpersonation is implied by invisibility, which concludes that Goldwasser and Waisbard's security model [45] is weaker than Camenisch and Michels' security model.

Then from the perspective of designing and analysing DCS schemes, we show that it is feasible to construct a DCS scheme that supports the signer's ability of disavowing invalid signatures. In more details, we provide the following contributions:

- Two existing DCS schemes, i.e., Zhang et al's DCS scheme and Wei et al's society-oriented DCS scheme, are insecure, and attacks against invisibility can be identified. In particular, we fix the security flaw in one of the two schemes, i.e., Wei et al's scheme [83].

- A concrete DCS scheme with full verification can be constructed by using bilinear maps. Such a DCS scheme is provably secure in the random oracle model under a newly introduced computational assumption. And the intractability of this computational assumption in pairing settings is analyzed in generic group model. Moreover, the proposed scheme can be further transformed into a unified verification version, which allows the signer and the confirmer to run the same verification protocols. We also propose a very efficient way that transforms our verification protocols into concurrent zero knowledge protocols which is invulnerable to any adversary during the concurrent executions of verification protocols.

- A generic DCS scheme with full verification can be constructed by extending the constructions of the improved GMR scheme in [78]. The building blocks of this generic DCS scheme are, a statistically hiding computationally binding commitment scheme, an IND-CCA2 secure public key encryption scheme which supports the use of "labels", and an EUF-CMA digital signature scheme. In addition, we give a formal security analysis, and implement an instantiation of the generic construction by using BLS short signatures [14], Pedersen commitments [67] and CS-Paillier cryptosystem [19].

This thesis has resulted in two publications:

- [86] Fubiao Xia, Guilin Wang, and Rui Xue. On the Invisibility of Designated Confirmer Signatures. In Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security (ASIACCS'11), pp. 268-276, ACM Press. March 22-24, 2011, Hong Kong, China.

- [80] Guilin Wang, Fubiao Xia, and Yunlei Zhao. Designated Confirmer Signatures With Unified Verification. In Proceedings of the 13th IMA International Conference on Cryptography and Coding (IMACC'11), LNCS 7089, pp. 469-495, Springer-Verlag, 2011. December 12-15, 2011; University of Oxford, UK.

Also the following work is published during my PhD study on some topic which is not related to designated confirmer signatures.

- [87] Qi Xie, Guilin Wang, Fubiao Xia, and Deren Chen. Provably Secure Self-Certified Proxy Convertible Authenticated Encryption Scheme. In: Proc. of the 4th International Conference on Intelligent Networking and Collaborative Systems (INCoS-2012), Bucharest, Romania, September 19-21, 2012.

1.3 Thesis Outline

Chapter 2 aims at providing a brief overview of the cryptographic background required to understand the sequel of this work. After recalling the concept of "provable security", we discuss two commonly used idealisations of security models, namely, the random oracle model and the generic model. Then we present a survey of interactive proofs and zero-knowledge proofs, which are necessary in the verification protocols of a designated confirmer signature scheme. We also introduce the pairing based cryptography by recalling the definition of bilinear maps, which will be used as a building block in the concrete DCS scheme in Chapter 5. The last section of this chapter is dedicated to a review of the definitions and security notions of several cryptographic primitives, including public key encryptions, digital signatures and undeniable signatures.

Chapter 3 is devoted to general aspects of designated confirmer signatures. First of all, we present the context and motivations in an introduction to this cryptographic primitive, and discuss some important works among the previous DCS schemes. The subsequent section provides a formal definition of the security model. Next to this, we introduce two existing concrete DCS schemes which is vulnerable to some considerable attacks.

Chapter 4 exposes the modelling aspect of designated confirmer signatures, namely we clarified different security notions under proper assumptions, associated with both intuitive discussions and formal security proofs. Firstly, comparing with the properties introduced in the previous chapter, we recall two different security notions, namely, **unimpersonation** (appears in [64, 45]) and **transcript-simulatability** (firstly introduced in [38]), and we proved the property called unimpersonation, is naturally satisfied if a stronger property called **invisibility** (initially introduced in [18]) is satisfied. Next, we analyse the relations between transcript-simulatability, invisibility, and **non-transferability** (appears in [58, 18]) with formal proofs. Our result shows that transcript-simulatability in Gentry et al.'s model [38] is also implicitly covered by Camenisch and Michels' [18] DCS model.

Chapter 5 is dedicated to the design and security analysis of a new concrete designated confirmer signature scheme with unified verification. We first put forward some building blocks of our new scheme, including bilinear pairings, and BLS signatures [14]. Next, we develop the transformation of concurrent zero knowledge (CZK) proofs from honest verifier zero knowledge proofs, because CZK protocols are required since an adversary in DCS schemes may act as arbitrary cheating verifiers during the concurrent execution of verification protocols. Subsequently, we introduce the updated model for DCS with unified (full) verification. Based on these results, we propose a DCS scheme with unified verification by using BLS signature and signed ElGamal encryption, and prove security results according to the definitions in the updated security model. Furthermore, such a construction can be simply transformed into a full-verification version.

Chapter 6 deals with the design and analysis of a generic designated confirmer signature scheme with full verification. After presenting the context and motivation of this work, we introduce two cryptographic primitives, that is, the commitment scheme which is used as a "layer of indirection", to achieve the efficient instantiations, as well as the public key encryption scheme that supports the use of labels to enhance the security. Next to this, we propose a generic transformation to convert any digital signatures into designated confirmer signatures with full verification. A formal security analysis of the proposed scheme is provided with regarding to the definitions in the previous chapter. Subsequently, we show how to efficiently instantiate the generic construction by properly choosing a digital signature scheme, a commitment scheme and a public key encryption scheme.

Chapter 7 concludes this work and suggests some future research directions in both theoretical and practical aspects concern with our results, which are worth to investigate from our viewpoint.

Chapter 2

Preliminaries in Cryptography

2.1 General Definitions

To denote the set of different numbers, we use the “blackboard” font such as the set of positive integers $\mathbb{N}$, the integers $\mathbb{Z}$, the real numbers $\mathbb{R}$, the non-negative real numbers $\mathbb{R}^+$. Throughout this thesis, $\lambda \in \mathbb{N}$ denotes the security parameter. The set $\{0,1\}^*$ stands for the set of the bitstrings of arbitrary length. For any bitstring $x \in \{0,1\}^*$, we use the symbol $|x|$ to denote its length, i.e., the number of bits it is composed of.

We consider only algorithms $\mathcal{A}$ which are probabilistic Turing machine that run in time polynomial in λ unless indicated otherwise. If $\mathcal{S}$ is a set, then $x \leftarrow \mathcal{S}$ indicates that x is chosen uniformly at random over $\mathcal{S}$. We remark that, even if occasionally not mentioned, all algorithms in this thesis receive the security parameter λ as an additional input.

Definition 2.1. (A Negligible Function). *we say that a function, $\epsilon : \mathbb{N} \rightarrow \mathbb{R}^+$ is called negligible if for every constant $c \geq 0$, there exists an integer k_c such that $\epsilon(k) \leq k^{-c}$ for all $k \geq k_c$. We will denote by $negl(\cdot)$ any negligible function.*

2.2 Provable Security

After the first revolution in the 1970s, when the notion of public key cryptography is invented by Diffie and Hellman [28], and made possible by Rivest, Shamir and Adleman [71], cryptography underwent another revolution in the 1980s, namely, the discovery that one could provide formal definitions of security for cryptographic problems, and such definitions were achievable under complexity assumptions.

The term “provable security” has been criticized since security is not being proved; only a reduction from security to some other unproved assumptions. A significant line of research has turned out to construct proofs in the framework of complexity theory, also known as “reductionist security proofs” [3]: the proofs provide reductions from a well-studied problem (RSA or the discrete logarithm) to an attack against a cryptographic protocol.

The initial attempts of defining security notions were actually trying to minimize the required assumptions on the primitives like one-way functions without considering practicality. Therefore, one just needs to design a scheme with some polynomial algorithms, and to present a polynomial reductions from the basic assumption on the primitive to an attack of the security notion, in an asymptotic way. However, those results may indeed have no practical impact on actual security, because of the **tightness** of a reduction. More specifically, in a non-tight reduction, one may still construct an adversarial algorithm that breaks the cryptographic protocol within a few hours, while the reduction leads to an algorithm against the underlying problem that requires many years.

For a few years, people have tried to provide both practical schemes, with practical reductions and exact complexity, which prove the security for realistic parameters, under a well-defined assumption: exact reduction in the standard model (which means in the complexity-theoretic framework). Unfortunately, practical or even just efficient reductions in the standard model can rarely be conjugated with practical implementations. Therefore, one needs to make some hypotheses on the adversary, namely, the attack is generic, independent of the actual implementation of some components:

1. hash functions, in the random oracle model;
2. symmetric block ciphers, in the ideal-cipher model;
3. algebraic groups, in the generic model.

we give more detailed explanations about random oracle model and generic model, which are necessary background for the sequel of this thesis. In particular, both two models are referred in the security proofs of our DCS constructions.

2.2.1 The Random Oracle Model

In the early attempts of designing cryptographic protocols with provable security, very few practical schemes can be proved in this **"standard model"**, in which the adversary is only limited by the amount of time and computational power. In 1993, Bellare and Rogaway [5] proposed a trade-off to achieve some kind of security validation for cryptographic protocols, by identifying some concrete crypto-objects with ideal random ones. The most famous identification appeared in the so-called "random-oracle model".

Various cryptographic schemes have adopted a hash function $\mathcal{H}$, such as MD5 [70] and SHA-1 [77]. This use of hash functions was originally motivated by the wish to sign long messages with a single short signature. In order to achieve **non-repudiation**, a minimal requirement on the hash function is the impossibility for the signer to find two different messages providing the same hash value. This property is called **collision-resistance**.

Hash functions are found as an essential ingredient for the security of digital signature schemes, and even for the security of the most cryptographic schemes. In order to obtain security arguments, while keeping the efficiency of the designs which use hash functions. A few authors suggested using the hypothesis that "$\mathcal{H}$ behaves like a random function", to obtain security proofs (or more precisely "security arguments"), while reserving the efficiency of the designs. Fiat and Shamir [31] applied the random oracle heuristically to construct a signature scheme which is "as secure as" factorization. Later, Bellare and Rogaway [6, 7] formalised this concept for digital signature and public-key encryptions.

In the random-oracle model, the hash function can be formalised by an oracle that outputs a truly random value for each new query. Certainly, identical answers are received if the same query is asked twice. This is precisely the context of complexity theory with "oracles", and hence the name.

We give a formalised description as below.

Definition 2.2. The Random Oracle Model (ROM). *Let G be a group of prime order q with a generator g, a range $M = \{0, 1\}^*$ of messages, and let $\mathbb{Z}_q$ denote the field of integers modulo q. Let $\mathcal{H}$ be an ideal hash function with range $\mathbb{Z}_q$, modelled as an oracle that given an input (query) in $G \times M$, outputs a random number in $\mathbb{Z}_q$. Formally, $\mathcal{H}$ is a random function $\mathcal{H}$: $G \times M \rightarrow \mathbb{Z}_q$ chosen at random over all functions of that type with uniform probability distribution.*

Canetti, Goldreich and Halevi hold a rather negative view on the ROM-based security proofs [20]. They demonstrate that there exists signature and encryption schemes which are provably secure under the ROM, but cannot reserve the security in the real world implementations. Their basic idea is to devise nasty schemes. Such a scheme usually behaves properly as a signature scheme or an encryption scheme. However, upon holding of a certain condition such as non-randomness is sensed, the scheme becomes nasty and outputs the private signing key if it is a signature scheme, or the plaintext message if it is an encryption scheme.

Another interesting view of ROM is given by Mao, the author of the book “Modern Cryptography: Theory and Practice” [55]. He gives his argument in the subsection 15.2.7, under the fact revealed by the ROM-based security proof for the RSA-OAEP (Optimal Asymmetric Encryption Padding is a padding scheme introduced by Bellare and Rogaway [7], often used together with RSA encryption). That is, if the padding scheme uses a truly random function, then the padding result output from OAEP is a “plaintext” in an ideal world: it has a uniformly random distribution in the plaintext space of the RSA function. Thus, his investigation on the strength of the RSA function being used in the ideal world concludes that the easiest way to break the IND-CCA2 security is to solve the RSA problem first and then to do what the decryption algorithm does. Furthermore, the ROM-based proof suggests that for a real world padding-based encryption scheme which uses real world hash functions rather than ROs, the most vulnerable point to mount an attack is the hash functions used in the scheme. From this point of view, he considers that a ROM-based technique for a security proof manifests its importance such that it suggests where to focus the attention for careful design. For instance, in order to reach a high confidence about a padding based encryption scheme, people should pay much attention on the design of hash function and its inputting randomness.

Although there is an ongoing debate on whether the assumption of a random hash function is realistic or too generous, this model has been strongly accepted by the community, and is considered as a good one, in which security analysis give a good taste of the actual security level. The problem is that random functions can in principle not be implemented by public algorithms. However, even if it does not provide a formal security proof, comparing with the proofs in the standard model without any ideal assumption), it is argued that proofs in this model guarantee the security of the overall design of the scheme provided that the hash function has no weakness. On the other hand, proofs in the random oracle model are still widely used today as they lead to better reductions than any other proof technique.

This model can also be regarded as a restriction on the adversary's capabilities. It simply means that the attack is generic without considering any particular instantiation of the hash functions. Therefore, an actual attack would necessarily use a weakness or a specific feature of the hash function. The replacement of the hash function by another one would rule out this attack. On the other hand, assuming the tamper-resistance of some devices, such as smart cards, the random-oracle model is equivalent to the standard model, which simply requires the existence of pseudo-random functions [40, 60]. As a consequence, almost all the designs of cryptographic protocols by now require provably security, at least in the random oracle model.

2.2.2 The Generic Model

The generic model is also an idealised cryptographic model like the random oracle model. Researchers use this model to analyse the computational hardness assumptions, namely, to prove a lower bound on the complexity of computing the corresponding intractable problems. Nechaev [62] proves that the discrete logarithm problem is hard in such a model. The generic model of algorithms was further elaborated on by Shoup [74].

In the generic model, it is assumed that the properties of the representation of the elements of the algebraic structure (e.g. a group) under consideration cannot be exploited. The adversary is only given access to a randomly chosen encoding of a group, instead of efficient encodings, such as those used by the finite field or elliptic curve groups used in practice.. In fact, for some problems like the discrete logarithm problem on general elliptic curves, exploiting the representation is not known to be of any help, and hence generic algorithms are the best known, such an assumption is reasonable from a practical point of view.

Generic Algorithms In order to motivate the later security proof to be introduced, we briefly discuss generic algorithms in a cyclic group.

Let G be a multiplicative cyclic group of integers mod n, and let S be a set of bit strings of cardinality at least n. An encoding function of G is an **injective map** ξ from G into S.

A generic algorithm (adversary) $\mathcal{A}$ for G is a probabilistic algorithm that behaves as follows. It takes an encoding list of the group elements as input. When the algorithm executes, it may make up oracle queries from time to time. More precisely, it specifies two exponents ρ_i and ρ_j, and a bit of operator into the encoding list. The oracle computes $\xi(\rho_i \pm \rho_j)$, according to the operator bit, and this string is appended to the encoding list.

Note that the algorithm $\mathcal{A}$ depends on S and n, but not on ξ; $\mathcal{A}$ can only retrieve the information about ξ through the oracle. The term generic means that one cannot exploit non-trivial properties of the representation of group elements, except for two generic properties that any representation has. Firstly, one can have the equality test of elements, and secondly one can impose a total order relation $\preceq$ on any representation.

2.3 Interactive Proofs

2.3.1 Interactive Proof Systems

This section introduces the notion of interactive proof systems, which is a fundamental tool for designing cryptographic protocols. It is intended to recall some basic material and to restrict to the necessary background for the sequel of this work. Most of the subsequent results are taken from the book of Goldreich [39] and the paper of Goldwasser et al. [43].

The concept of interactive proof systems (or interactive proofs) was motivated by the need of secure cryptographic protocols, such as identification protocols. In an interactive proof system, a player, called the **prover**, needs to interactively prove the validity of a given statement to another player, called the **verifier**. Formally, the prover and the verifier are modelled by some interactive Turing machines. An interactive Turing machine (ITM) is a Turing machine equipped with a read-only input tape, a work tape, a random tape, one read-only communication tape, and one write-only communication tape. The random tape contains an infinite sequence of random bits, and can be scanned only from left to right. We say that an interactive machine flips a coin, meaning that it reads the next bit in its own random tape. An interactive machine M expects some input at the beginning of its execution, and from then on alternately sends and receives messages. It may finally terminate with some output. Such a machine may be probabilistic (i.e., use randomnesses).

Definition 2.3. *We say an interactive Turing machine M polynomial time if there is a polynomial p, such that the machine M runs at most $p(|x|)$ steps upon input x, no matter what and how many messages it receives.*

Let $\mathcal{R}$ be an efficiently computable relation on pairs of bitstrings. We say that x is a true statement and ω is a witness for x if $(x, \omega) \in \mathcal{R}$ (or $x\mathcal{R}\omega$). For instance, if we want to prove a "big integer" is not prime, we would use the relation $\mathcal{R}$ with $x\mathcal{R}(p, q)$ iff $x = pq$ and $p, q > 1$. The witness ω corresponds to a proof that x is a true statement.

Let $L_{\mathcal{R}}$ denote the language of all true statements, and we have, $L_{\mathcal{R}} := \{x : \exists\omega.(x, \omega) \in \mathcal{R}\}$. We define the concept of interactive proof systems by using the above notation.

Definition 2.4. *Let P denote the prover and V denote the verifier. An interactive proof system $\langle P, V\rangle$ (or interactive proof) for a relation $\mathcal{R}$ with completeness bound c and soundness bound s is a pair of polynomial-time interactive Turing machine P and V, such that the following properties are satisfied.*

- *Completeness: If $(x, \omega) \in \mathcal{R}$ then $Pr[\langle P(x, \omega), V(x)\rangle = 1] \geq c(|x|)$, where the probability is over the random tapes of P and V. (Intuitively, the honest prover will succeed in proving a true statement with probability at least $c(|x|)$, provided it knows a witness.)*

- *Soundness: For any (possibly computationally unbounded) interactive machine P^* (the cheating prover), and for any $x \notin L_{\mathcal{R}}$, we have that $Pr[\langle P^*, V(x)\rangle = 1] \leq s(|x|)$, where the probability is over the random tapes of P^* and V. (Intuitively, even a dishonest, unbounded prover will not succeed in convincing the honest verifier of a wrong statement.)*

Obviously, an interactive proof is better if c is close to 1 (it almost always succeeds), and s is close to 0 (one almost never proves anything wrong). If $c = 1$, we say the proof has **perfect completeness**. If the soundness property only holds against a polynomial-time (in $|x|$) malicious prover, we say that the interactive proof is **computationally sound**.

2.3.2 Zero-knowledge Proofs

In mathematics and in life, if we want to convince you that we know X is true, usually, we commonly try to present all facts we know and the inferences from that facts that imply X is true. For instance, if you want to prove a “big integer” is not prime, the straight way is you expose the factorization, that is, the big integer is a product of two integers. This approach gives a typical byproduct that you gained some knowledge, other than that you are convinced that the statement is true.

a zero-knowledge proof (ZKP) is an interactive method to address that issue. In a zero-knowledge proof, when Alice prove to Bobs that a statement X is true, Bob will completely convinced that X is true, but will learn nothing else as a result of this process. That is, Bob will gain **zero knowledge**.

Firstly conceived by Goldwasser, Micali and Rackoff [43] in the "GMR" paper in $FOCS'85$, zero-knowledge proofs turned out to be one of the most interesting and influential topics in computer science, with applications ranging from practical signature schemes to complexity proofs for many NP-complete problems.

a zero-knowledge proof must satisfy three properties:

- **Completeness**: if the statement is true, the honest verifier will be convinced of this fact with a non-negligible probability, by a honest prover who knows the witness.

- **Soundness**: if the statement is false, even a dishonest prover will not succeed in convincing the honest verifier of a wrong statement.

- **Zero-knowledge**: if the statement is true, no cheating verifier learns anything other than this fact. This notion is formalised by showing that, for every cheating verifier, there always exists a simulator such that given only the statement to be proved (and no access to the prover), can output a transcript that "indistinguishable" from the transcript of an interaction between the honest prover and the cheating verifier.

Initially formalised in [43], we present a formal definition of zero-knowledge as follows. Suppose all the messages exchanged between a prover P and a verifier V in an interactive proof form the transcript of the protocol. We denote by $View(\langle P, V\rangle)$ the transcript (random variable) of the interactive proof $\langle P, V\rangle$.

Definition 2.5. (Zero-knowledge). *We say that an interactive proof system $\langle P, V\rangle$ for the language $L_{\mathcal{R}}$ (with corresponding relation $\mathcal{R}$) is (perfect, statistical, or computational) zero-knowledge if for any probabilistic polynomial-time interactive machine V^* and any polynomial p, there exists a probabilistic polynomial-time algorithm $\mathcal{S}$ called the simulator, such that the ensembles:*

$$\{View(\langle P(\omega), V^*(y)\rangle\,(x))\}_{(x,\omega)\in\mathcal{R}, y\in\{0,1\}^{p(|x|)}} \textit{ and } \{\mathcal{S}(x,y)\}_{(x,\omega)\in\mathcal{R}, y\in\{0,1\}^{p(|x|)}}$$

are (perfectly, statistically, or computationally) indistinguishable, where$\mathcal{S}(x,y)$ denotes the random variable of $\mathcal{S}$'s output with the inputs x, y.

This definition of zero-knowledge with an auxiliary input to the verifier is an extension of the classical definition, where the verifier is only given the common input x. The auxiliary input can be some information known by the verifier before the beginning of the interaction.

2.3.3 Proofs of Knowledge

Basically this is a stronger form of soundness of interactive proofs, which guarantee that no statement x is accepted by the verifier such that there is no witness ω with $(x,\omega) \in R$. Such a proof can hence be interpreted as proving the statement of "there is a witness ω for x". In some scenarios, however, we require the prover to show something stronger, namely, "I know a witness ω for x".

In the following, when we say a machine K has oracle access with rewinding to an interactive machine P, we mean that, K can interact with P in an arbitrary fashion and at any point reset P's state to any state in the history of P's execution. In particular, the machine K may send inputs of the following forms to P:(msg, m) and $(rewind, i)$. Upon receiving input (msg, m), the code of P is executed on the current state of P to compute the reaction to the message m. The message output by P is returned to K. The new state of P is appended to a list H of states. Upon receiving input$(rewind, i)$, P's state is set to the i-th element of H.

The notion of **a proof of knowledge** is defined as below.

Definition 2.6. (A Proof of Knowledge). *We call (P, V) a proof of knowledge with completeness bound c, soundness bound s, and knowledge error κ if the following holds:*

- *$(P, V)$is a proof with completeness bound c, soundness bound s.*
- *Validity: There exists a constant $d > 0$ and a polynomial-time oracle machine K with rewinding such that for any interactive machine P^* and any $x \in L_{\mathcal{R}}$, we have the following:*

$$Pr[\langle P^*, V(x)\rangle = 1] \geq \kappa(|x|) \Rightarrow Pr[(x, \omega) \in \mathcal{R} : \omega \leftarrow K^{P^*}(x)]$$
$$\geq (Pr[\langle P^*, V(x)\rangle = 1] - \kappa(|x|))^d.$$

2.4 Pairing-based Cryptography

Elliptic curves naturally occur in the study of congruent numbers and Diophantine equations, which has been a research area for a long time already. At the beginning, researchers found the study of curves over finite fields seem to forms rather boring Abelian groups. However, in 1985, Miller [59] found an application of elliptic curves over finite fields in cryptology. Two years later, Koblitz [52], alternatively figured, elliptic curves could provide a similar level of security while using shorter keys. Thereafter a lot of research works [54, 56, 73, 75] have been put in elliptic curve cryptography (ECC) and many cryptosystems [35, 17, 46] have been proposed. Pairings are cryptanalysis tools which were initially used for attacking the existing cryptosystems. In this context, a pairing may be treated as a function that takes two points on an elliptic curve as input, and outputs an element of some multiplicative group. Basically, a pairing meets some special properties, like **bilinearity** and **non-degeneracy**, and is naturally hard to construct.

Weil pairing [84] and Tate pairing [76] are two well-studied symmetric pairings, while some other pairings have been given more and more attentions, like Eta pairing [2] and Ate Pairing [34]. Until 2000, pairings are found by Joux [51] that they can be contributed to cryptographic building blocks as well, whose discovery spurred an extensive research into pairings and their new applications.

Definition 2.7. (Bilinear Maps). *Suppose that G and G_t be two multiplicative cyclic groups of prime order q, while g is a generators of G. A bilinear pairing on (G, G_t) is a map $e : G \times G \rightarrow G_t$, which satisfies the following properties:*

- *Bilinearity: For all $u, v \in G$, and for all $a, b \in \mathbb{Z}_q$, $e(u^a, v^b) = e(u, v)^{ab}$.*
- *Non-degeneracy: $e(g, g) \neq 1$, where 1 is the multiplicative identity of group G_t.*
- *Computability: e can be efficiently computed.*

Considering the DL(discrete logarithm)-problem, if (P, Q) is an instance of the DL-Problem in G where $Q = P^x$, then $e(P, Q) = e(P, P^x) = [e(P, P)]^x$. Thus $log_P Q = log_g h$, where $g = e(P, P)$ and $h = e(P, Q)$ are elements of G_t. Hence, the DL-problem in G can be efficiently reduced to the DL-problem in G_t.

Considering the bilinear Diffie-Hellman problem (BDHP), which is introduced by Boneh and Franklin [12]: given P, P^a, P^b, P^c, compute $e(P, P)^{abc}$. The following reasons state that the hardness of the BDHP implies the hardness of the Diffie-Hellman Problem (DHP) [28] in both G and G_t . Firstly, if the DHP in G can be efficiently solved, then one could solve an instance of the BDHP by computing abP and then $e(P^{ab}, P^c) = e(P, P)^{abc}$. Also, if the DHP in G_t can be efficiently solved, then the BDHP instance could be solved by computing $g = e(P, P)$, $g^{ab} = e(P^a, P^b)$, $g^c = e(P, P^c)$ and then g^{abc}.

2.5 Some Cryptographic Primitives

2.5.1 Public Key Encryption Schemes

The concept of public-key cryptography was firstly developed in the ground-breaking article of Diffie and Hellman [28] without proposing a concrete public-key encryption (PKE) scheme. And possibly the most famous PKE is RSA cryptosystem [71], which was introduced by Rivest, Shamir and Adleman. In general, a public key encryption scheme allows two parties to communicate in a confidential way with the help of an authenticated channel, which is used to transmit the public key. Such a cryptographic system requires two separate keys, one of which is secret and one of which is public. This ensures any message encrypted with respect to the public key will only be decrypted by the owner of the corresponding secret key (or called *private key* alternatively).

We denote the message space by $\mathcal{M}$ and the ciphertext (encrypted message) space by $\mathcal{C}$. A public key encryption scheme is composed of three polynomial time algorithms.

Setup: *This PPT algorithm takes the security parameter λ, denoted by 1^λ as input, and outputs a pair of matching public and private key. We have $(pk, sk) \leftarrow Setup(1^\lambda)$.*

Enc: *The encryption algorithm is a PPT algorithm that takes a message $m \in \mathcal{M}$ and a public key pk as input, and outputs a ciphertext c. We have $c \leftarrow Enc(m, pk)$.*

Dec: *The decryption algorithm is a deterministic algorithm that takes a ciphertext*

$c \in \mathcal{C}$ and a private key sk as input, and outputs a message $m \in \mathcal{M}$. We have $m \leftarrow Dec(c, sk)$. In some cases, this algorithm returns a special symbol $\perp$ to indicate that the ciphertext was invalid.

We require that for all (pk, sk) which can be output by $Setup(1^\lambda)$, for all $m \in \mathcal{M}$, and for all c that can be output by $Enc(m, pk)$, we have that $Dec(c, sk) = m$. We also require that $Setup$, Enc and Dec can be computed in polynomial time.

The classical goal of secure encryption is to protect the privacy of messages: an adversary should not be able to learn from a ciphertext information about its plaintext beyond the length of that plaintext, which is formalised as *indistinguishability of encryptions*, due to Goldwasser and Micali [42].

Along the other axis, we consider three different attacks. These are *chosen-plaintext attack* (CPA), *non-adaptive chosen-ciphertext attack* (CCA1), and *adaptive chosen-ciphertext attack* (CCA2), in order of increasing strength. Under CPA the adversary can obtain ciphertexts of plaintexts of its choice. In the public-key setting, giving the adversary the public key suffices to capture these attacks. Under CCA1, formalised by Naor and Yung [61], the adversary gets, in addition to the public key, access to an oracle for the decryption function. Under CCA2, due to Racko and Simon [69], the adversary again gets access to an oracle for the decryption function, but this time it may use this decryption function even on ciphertexts chosen after obtaining the challenge ciphertext c, the only restriction being that the adversary may not ask for the decryption of c itself.

One can "mix-and-match" the goal– IND and the attacks {CPA,CCA1,CCA2} in three combinations, giving rise to three notions of security: IND-CPA, IND-CCA1, and IND-CCA2. Below, we present these three security notions based on the definition in [4] with minor changes.

Let the string atk be instantiated by any of the formal symbols cpa, cca1, cca2;

while ATK is then the corresponding formal symbol from CPA, CCA1, CCA2. When we say $\mathcal{O}_i = \varepsilon$, where $i \in \{1, 2\}$, we mean $\mathcal{O}_i$ is the function which, on any input, returns the empty string ε.

Definition 2.8. (IND-CPA, IND-CCA1, IND-CCA2). Let $\Pi = (Setup, Enc, Dec)$ be an encryption scheme and let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary. For atk $\in${cpa, cca1,cca2} and $\lambda \in N$, let $adv_{\mathcal{A},\Pi}^{ind-atk}(\lambda)$ denote the following probability:

$$2 \cdot \Pr[(pk, sk) \leftarrow Setup(1^\lambda); (m_0, m_1, s) \leftarrow \mathcal{A}_1^{\mathcal{O}_1}(pk);$$
$$b \leftarrow \{0,1\}; c \leftarrow Enc(m_b); \mathcal{A}_2^{\mathcal{O}_2}(m_0, m_1, s, c) = b] - 1$$

where

if atk=cpa, then $\mathcal{O}_1 = \varepsilon$ and $\mathcal{O}_2 = \varepsilon$;

if atk=cca1, then $\mathcal{O}_1 = Dec(\cdot)$ and $\mathcal{O}_2 = \varepsilon$;

if atk=cpa, then $\mathcal{O}_1 = Dec(\cdot)$ and $\mathcal{O}_2 = Dec(\cdot)$;

In the case of CCA2, we insist that $\mathcal{A}_2$ does not ask its oracle to decrypt c. We say that Π is secure in the sense of IND-ATK if $\mathcal{A}$ being polynomial-time implies that $adv_{\mathcal{A},\Pi}^{ind-atk}(\cdot)$ is negligible.

2.5.2 Digital Signatures

The main goal of digital signatures is to reproduce the electronic version of handwritten signatures, i.e., the signature on a message is a string which binds the message, and public and secret data specific to the user; anyone can check the validity of the signature by using public data only. Digital signatures become more and more crucial since their invention [28], and are now used in numerous cryptographic protocols. In 1988, Goldwasser, Micali and Rivest became the first [44] to rigorously define the security requirements of digital signature schemes. We introduce the formal definition of signature schemes, with minor changes, following David Pointcheval's proposal [68] that formally presented the precise definition of digital signatures and of the possible attacks against them.

A digital signature scheme usually consists of the following probabilistic polynomial-time (PPT) algorithms. Let $\mathcal{M}$ and Σ denote the message space and the signature space respectively. In general, $\mathcal{M}$ is the set of messages to which the signature algorithm may be applied, and Σ is the set of signatures can be produced with an instance of the digital signature scheme.

Setup: *This PPT algorithm takes the security parameter λ, denoted by 1^λ as input, and outputs a key pair which is associated with the signer, $(pk, sk) \leftarrow Setup(1^\lambda)$.*

Sign: *This PPT algorithm generates a signature σ on a given message $m \in \mathcal{M}$, $\sigma \leftarrow Sign(m, sk)$.*

Verify: *This verification algorithm is usually deterministic and takes a message-signature pair $(m, \sigma) \in \mathcal{M} \times \Sigma$ and an associated public key pk as input, outputs a bit 0 or 1. We have $b \leftarrow Verify(m, \sigma, pk)$, where $b \in \{0, 1\}$. This algorithm tells whether the pair (m, σ) is valid with respect to the key pair (pk, sk). The output bit 1 means that the signature is valid.*

In practice, signer's public key pk needs to be sent through an authenticated channel so that the verifiers are ensured that pk really corresponds to the right signer. Once this operation is performed, the signer can authenticate the messages to a verifier using digital signatures even through via an insecure channel.

The security of the signature schemes usually depends on what kind of attack results the adversary could achieve. A hierarchy of attack results are discussed in [68, 44]. One might say that the enemy has "broken" a signer's signature scheme if his attack allows him to do any of the following with a non-negligible probability:

- Disclose the private key/data of the signer, which is the most serious attack, so-called *total break*.

- Construct an efficient algorithm to sign messages with good probability of success, so called *universal forgery*.
- Forge a signature for a particular message chosen a prior by the enemy, so called *selective forgery*.
- Provide a new message-signature pair. This is named as *existential forgery*.

Considering the security requirements, an adversary can have some access to the signature protocols with different power levels. For instance, there could be an attacker which has the only access to public key of the signer while another attacker may have the access to a list of valid message-signature pairs. The attacker in the latter scenario is obviously much powerful, and is named as "*adaptive chosen-message attack*". In an adaptive attacking situation, the attacker can ask the signer to sign any message of its choice and it can adjust its queries according to previous answers. Currently, the strongest security notion is the security against existential forgery under an adaptive chosen-message attack. So when designing a signature scheme, one may want to computationally prevent at least existential forgeries, under adaptive chosen-message attacks, which is widely accepted as the standard definition. We give a formalisation as below.

Definition 2.9. (Existential Unforgeability) *Let $\mathcal{O}$ be a signing oracle that implements the algorithm $Sign$. We denote by L the list of all messages queried to $\mathcal{O}$. A signature scheme is secure against an existential forgery under an adaptive chosen-message attack, if for any probabilistic polynomial time forger (algorithm) $\mathcal{F}$, we have*

$$\Pr[1 \leftarrow Verify(m, \sigma, pk) \wedge m \notin L | (pk, sk) \leftarrow Setup(1^{\lambda}), (m, \sigma) \leftarrow \mathcal{F}^{\mathcal{O}}] = neg(\lambda)$$

where the probability is taken over the random tapes of the involved algorithms. Each invocation to the oracle $\mathcal{O}$ is counted in the complexity of $\mathcal{F}$, and the number of queries made to $\mathcal{O}$ must also be polynomially bounded in λ.

2.5.3 Undeniable Signatures

The conflict between authenticity (non-repudiation) and privacy (controlled verifiability) always exists in the digital signature world. As mentioned in Chapter 1, undeniable signature was firstly introduced to solve such a problem by letting the signer interactively prove to any verifier it selected. Undeniable signature as an fundamental cryptographic primitive, is the origin of designated confirmer signatures, and we present the formal definition of this cryptographic primitive as below.

An undeniable signature scheme consists of two algorithms, namely $Setup$ and $Sign$, and two protocols, namely $Confirm$ and $Disavow$. For every choice of the security parameter λ there is a message space $\mathcal{M}$ and a signature space Σ. Also we denote the signer by S, and the verifier by V.

Setup: *This PPT algorithm takes the security parameter λ, denoted by 1^λ as input, and outputs a key pair which is associated with the signer, $(pk, sk) \leftarrow Setup(1^\lambda)$.*

Sign: *This PPT algorithm generates a signature $\sigma \in \Sigma$ on a given message $m \in \mathcal{M}$, $\sigma \leftarrow Sign(m, sk)$.*

Confirm: *This protocol is executed between a signer and a verifier interactively. The common inputs are a message-signature pair $(m, \sigma) \in \mathcal{M} \times \Sigma$, and the signer's public key pk. The protocol outputs a bit b, and allows the signer to prove to a verifier that the signature σ is* ***valid*** *for the message m and the key pk. We have $b \leftarrow Confirm_{(S,V)}(m, \sigma, pk)$, where $b \in \{0, 1\}$.*

Disavow: *This protocol is executed between a signer and a verifier interactively. The common inputs are a message-signature pair $(m, \sigma) \in \mathcal{M} \times \Sigma$, and the signer's* *public key pk. The protocol outputs a bit b, and allows the signer to prove to a verifier that the signature σ is* ***invalid*** *for the message m and the key pk. We have $b \leftarrow Disavow_{(S,V)}(m, \sigma, pk)$, where $b \in \{0, 1\}$.*

Chapter 3

An Overview on Designated Confirmer Signatures

Designated confirmer signatures (DCS) are introduced by Chaum and van Antwerpen [21] as an extension of undeniable signature. More specifically, in a designated confirmer signature scheme, if the signer is unavailable, a semi-trusted third party called *the designated confirmer* can confirm the (in)validity of an alleged signature by running some interactive protocols with a verifier. However, such a verifier cannot transfer the signature's (in)validity to other parties by convincing them of the same fact. Furthermore, the designated confirmer can selectively convert a designated confirmer signature into a standard signature so that it can be publicly verifiable. A number of related work have been presented in the last two decades, like [64, 58, 18, 45, 38, 78, 85], though most of them are either insecure or inefficient. For example, [58] identifies attacks against the two concrete DCS scheme proposed in [64], Camenisch and Michels [18] shows the insecurity of [58], Wang et al. [78] points out security flaws in the schemes proposed in [45, 38], while the solutions given in [18, 85] are not efficient as they rely on general zero-knowledge protocols.

Apart from unforgeability which is a common security requirement for variants of digital signatures, the unique security property of a DCS scheme is called *invisibility* [18], which requires that any probabilistic polynomial adversary *cannot* feasibly determine the (in)validity of an alleged signature against adaptive attacking environment. In this chapter, we first introduce a traditional security model of DCS schemes; then discuss attacks against invisibility of two practical DCS schemes, and then fix the security flaw in one of the two schemes [89] and [83], i.e., Wei et al's scheme [83].

Related Works on DCS Schemes Since the introduction by Chaum and van Antwerpen [21], various generic DCS schemes have been produced from ordinary digital signatures and other cryptographic primitives such as public key encryptions, commitment schemes, and/or zero-knowledge protocols. We briefly review the most important attempts in chronological order:

- Chaum and van Antwerpen (1994) [21]: The first proposition of designated confirmer signatures to solve the weakness of undeniable signatures, with an example of DCS based on RSA scheme.

- Okamoto (1994) [64]: Okamoto give the first formal definition of designated confirmer signatures in the sense of rigorous concept, and proposed a practical construction by using digital signatures, public key encryptions, bit-commitment schemes and pseudo-random functions. Also it rigorously proves that the existence of public-key encryption is the necessary and sufficient assumption for constructing designated confirmer signatures.

- Michels and Stadler (1998) [58]: They pointed out a certain weakness of the DCS schemes by Okamoto [64] that the confirmer can forge a valid signature on behalf of the signer. Realizing this problem, they further proposed a new security model and introduced an efficient DCS scheme in that model by using signatures with the Fiat-Shamir paradigm and commitment schemes.

- Camenisch and Michels (2000) [18]: The authors identified an attack against the DCS schemes proposed in [21, 64, 58], where the validity of a DCS issued by a signer S can be linked to that of a DCS issued by another signer S'. As a result, those schemes are insecure if multiple signers share the same confirmer, and such multi-signer settings seem to be natural in e-commerce applications. Based on that observation, they proposed a new security model to cover this variant of attacks, and presented the "encryption of a signature" idea along with a security analysis of the resulting generic DCS schemes. However, this construction is provably secure but inefficient. Because in their confirmation/disavowal protocols, to prove the correctness of such an encryption, actually relies on general zero-knowledge proofs for NP statement.

- Goldwasser and Waisbard (2004) [45]: They introduced an interesting security notion called "unimpersonation" in their security model, meanwhile, the new model circumvent the requirement of "invisibility" and "non-transferability". By exploiting *strong witness hiding proofs of knowledge*, instead of zero-knowledge proof of knowledge, Goldwasser and Waisbard proposed several secure DCS schemes without appealing to random oracles in the weakened security model. Moreover, the disavowal protocol of their construction has still recourse to general concurrent ZK proofs of NP statements.

- Gentry et al. (2005) [38]: Gentry, Molnar and Ramzan presented a generic transformation from any secure ordinary signature scheme into a DCS scheme. Their basic idea is to add a middle layer in the "sign-and-encrypt" paradigm. In particular, they issue a DCS by generating an ordinary signature on the commitment of a message, while the randomness used for the commitment is encrypted under the confirmer's secret key separately. They also proved the result is secure in their new model, which is an enhancement of the one introduced in [18]. The authors give an interesting transformation because it gives rise to an efficient generic DCS scheme without having resource to either the random oracles or the general zero-knowledge proofs.

- Wang et al (2007) [78]: The authors first identified two flaws in the construction of [38], and proposed an improved DCS scheme based on the insecure version. Then they introduced a new way of designing efficient and generic DCS schemes without any public key encryptions. [1] One limitation is their construction still appeals to the random oracles.

In contrast, only a few *concrete* DCS schemes have been proposed. In 2008, Zhang et al. proposed an efficient DCS scheme based on bilinear pairings [89]. In the same year, Wei et al. presented a society-oriented DCS scheme [83], which is a new concept for sharing the signer and confirmer's capability among two groups of individuals respectively. The construction is based on two threshold cryptosystems, namely a threshold signature scheme[49] and a threshold encryption scheme[79]. Wei et al.'s scheme is so called as it allows a threshold of possible signers to collectively issue a DCS and a threshold of designated confirmers to collectively confirm the (in)validity of such an alleged DCS. However, It is discovered that both Zhang et al's DCS scheme and Wei et al's SDCS scheme fail to meet *invisibility*, and we will pay more attention to the two schemes and show the related attacks later in this chapter.

[1]Note this construction does not contradict with the result in [64], as the later proves that designated confirmer signatures exist if and only if public key encryptions exist. The reason is that, the scheme in [78] uses a confirmer commitment scheme as the building block. As stated in [58], probabilistic public key encryption schemes are actually a special case of confirmer commitments. This gives an intuition about the relation between these two primitives, i.e., it might be possible to derive a secure probabilistic public key encryption scheme from a secure confirmer commitment scheme. Though this still need to be further investigated.

3.1 Security Model

This section reviews the syntax (Section 1.1) and security requirements (Section 1.2) of DCS schemes. In general, we follow the definitions given by Camenisch and Michels [18], because their formalization is widely complied with by various schemes in the last decade. To improve the readability, some changes are made. For instance, we update the security model by introducing the existence of some efficient computable relation R (see in Section 3.1.2), to make the expressions more accurate.

3.1.1 Definitions (Syntax)

Basically, a designated confirmer signature scheme consists of three parties, a signer, a designated confirmer, and a verifier. Once the signer issues a DCS, the verifier can interactively validate it with the help from either the signer or the confirmer. In particular, for any alleged DCS, the confirmer will first check its validity, and execute the corresponding protocol, i.e., using confirm protocol (for a valid signature) or disavow protocol (for an invalid signature).

In the formal definition given below, $negl(\lambda)$ denotes any negligible function that grows slower than λ^{-v} for any positive integer v and for all sufficiently large integer λ. $\{\mathcal{A}(u)\}$ denotes the set of all possible output values of a probabilistic algorithm $\mathcal{A}$ with input u. In addition, we use "$X \leftarrow P(\cdot)$" and "$Y = A(\cdot)$"to denote that a protocol P outputs X, and an algorithm A outputs Y, respectively.

Definition 3.1. (Syntax) [18]: *A correct designated confirmer signature scheme involves three roles of parties, i.e., a signer S, a designated confirmer C, and a verifier V, and consists of the following components:*

- ***Key Generation*** (G_S, G_C): *Given the security parameter λ, denoted by 1^λ as input, probabilistic polynomial time (PPT) algorithm G_S outputs a pair of strings*

(sk_S, pk_S) as the signer's private key and public key. Similarly, PPT algorithm G_C that takes on input 1^λ, outputs a pair of strings (sk_C, pk_C) as the designated confirmer's private key and public key.

- ***Sign**: Given a message m and a signer's private key sk_S, algorithm Sign produces a (standard) signature σ for message m. Namely, $\sigma = Sign(m, sk_S)$.*

- ***Verify**: Given a public key pk_S, a message m, and an alleged signature σ, algorithm Verify outputs a bit b, where $b = 1$ indicates "Accept", and $b = 0$ indicates "Reject". We require that for any key pair (sk_S, pk_S), any message m, $Verify(m, Sign(m, sk_S), pk_S) = 1$.*

- ***DCSSign**: Given a message m, a signer's private key sk_S and the confirmer's public key pk_C, DCSSign is a probabilistic algorithm that generates a designated confirmer signature on the input message. We have $\sigma' = DCSSign(m, sk_S, pk_S, pk_C)$.*

- ***Extract**: Given $(m, \sigma', sk_C, pk_C, pk_S)$ as input, algorithm $Extract$ outputs a string σ such that $Verify(m, \sigma, pk_S) = b$, where $b = 1$ indicates "Accept", and $b = 0$ indicates "Reject".*

- ***Confirm**: As an interactive protocol, the designated confirmer C with private input sk_C can run $Confirm$ protocol with a verifier V to confirm that an alleged DCS σ' for a message m is extractable. The common input for the protocol is (m, σ', pk_S, pk_C). Eventually, the verifier outputs a bit b,where $b = 1$ indicates "Accept", and $b = 0$ indicates "$\perp$". We say σ' is valid w.r.t. message m, if the verifier's output $b = 1$. Otherwise, the validity of σ' is undetermined. The Confirm protocol should be both complete and sound.*

- ***Disavow**: As an interactive protocol, the designated confirmer C with private input sk_C can run Disavow protocol with a verifier V to convince that an alleged*

DCS σ' is unextractable. The common input to the protocol is (m, σ', pk_S, pk_C). Eventually, the verifier outputs a bit b,where $b = 1$ indicates "Accept", and $b = 0$ indicates "$\perp$". If the verifier's output $b = 1$, we say σ' is invalid *w.r.t. message m. Otherwise, the invalidity of σ' is undetermined. The Disavow protocol should be complete and sound.*

Remark 1. The algorithms and protocols above actually only allows the designated confirmer to do the verification, by confirming any alleged valid DCS or disavowing any alleged invalid one. A not-fully-extended definition is, as proposed in [45, 38, 78], to allow the signer be able to (partly) verify the signatures, namely, a $ConfirmedSign$ protocol is prepared to let the signer confirm a DCS immediately it is honestly generated. Comparing to the previous syntax, we also introduce a fully-extended definition in Chapter 5, where we allow the signer has the ability to disavow any invalid DCS.

3.1.2 Security Requirements

We follow the definitions in [18], and briefly illuminate the notions as below, namely game-based formalizations of *unforgeability* and *invisibility* are presented to improve the readability and the further discussion.

Definition 3.2. Completeness of $Confirm/Disavow$: *If the confirmer and the verifier are honest, then for all λ, all $(sk_S, pk_S) \in G_S(1^\lambda)$, all $(sk_C, pk_C) \in G_C(1^\lambda)$, all $m \in \{0,1\}^*$, and all $\sigma' \in \{DCSSign(m, sk_S, pk_S, pk_C)\}$, we require that*

$$Confirm_{(C,V)}(m, \sigma', pk_S, pk_C) \rightarrow \begin{cases} 1 & if\ \sigma' \in \{DCSSign(m, sk_S, pk_S, pk_C)\} \\ 0 & otherwise \end{cases}$$

$$and$$

$$Disavow_{(C,V)}(m, \sigma', pk_S, pk_C) \rightarrow \begin{cases} 1 & if\ \sigma' \notin \{DCSSign(m, sk_S, pk_S, pk_C)\} \\ 0 & otherwise \end{cases}$$

Definition 3.3. Soundness of $Confirm/Disavow$**:** *For any cheating confirmer* C^**, for all sufficiently large* λ*, all* $(sk_S, pk_S) \in G_S(1^\lambda)$*, all* $(sk_C, pk_C) \in G_C(1^\lambda)$*, all* $m \in \{0,1\}^*$*, and all* $\sigma' \in \{DCSSign(m, sk_S, pk_S, pk_C)\}$*, we require that*

$$\Pr[Confirm_{(C^*,V)}(m, \sigma', pk_S, pk_C) \rightarrow 1] < negl(\lambda)$$

if $\sigma' \notin \{DCSSign(m, sk_S, pk_S, pk_C)\}$ *and*

$$\Pr[Disavow_{(C^*,V)}(m, \sigma', pk_S, pk_C) \rightarrow 0] < negl(\lambda)$$

if $\sigma' \in \{DCSSign(m, sk_S, pk_S, pk_C)\}$*, The probability is taken over the coin tosses of* C *and* C^* *.*

Definition 3.4. Correctness of $Extract$**:** *for all sufficiently large* λ*, all* $(sk_S, pk_S) \in G_S(1^\lambda)$*, all* $(sk_C, pk_C) \in G_C(1^\lambda)$*, all* $m \in \{0,1\}^*$*, all* $\sigma' \in \{0,1\}^*$*, and all* $\sigma' \in \{DCSSign(m, sk_S, pk_S, pk_C)\}$*, it holds that*

$$Verify(m, Extract(m, \sigma', sk_C, pk_C, pk_S), pk_S) = 1$$

.

Definition 3.5. Security for the signer (Unforgeability): Unforgeability requires that no adaptive PPT adversary can forge a valid DCS on a fresh message, even it compromises the confirmer's secret key sk_C. Note this is also part of the security requirement that holds against the confirmer, and thus we allow the secret key of the confirmer as the input.

For any DCS scheme, we can specify an efficiently computable equivalence relation R (this concept first appears in Gentry et al's work [38]), and say (m, σ'') and (m, σ') are **equivalent** if and only if $R(m, \sigma', \sigma'') = 1$. Informally, such a binary relation R is used to classify two valid signature pairs respect to the same message. For example, if a DCS scheme is **strongly existentially unforgeable**, it requires the forger cannot even produce a valid DCS on any previously signed message. In that case, it may be appropriate to specify $R(m, \sigma', \sigma'') = 1$ if and only if $\sigma' = \sigma''$. However, R needs not be that restrictive. It depends on the specific DCS scheme. We update the original definition of unforgeability by introducing the existence of such a relation R, an present a new definition as follows:

Game-UF: *Key generation algorithms are run on input 1^λ, and output (pk_S, sk_S), (pk_C, sk_C) as the public/private key-pairs of the signer and the confirmer respectively. Given pk_S, pk_C, and sk_C, an PPT adversary $\mathcal{A}$ is allowed oracle access to the signer (i.e., it may ask designated confirmer signatures of polynomially many messages $\{m_i\}$ via $DCSSign$), and to the confirmer (i.e, $\mathcal{A}$ can access $Confirm_{(C,\mathcal{A})}$, $Disavow_{(C,\mathcal{A})}$, and Extract oracles). Let L_{sig} denote the list of all message-signature pairs (m_i, σ'_i) output by $DCSSign$ oracle and all (m_i, σ''_i) such that $R(m_i, \sigma'_i, \sigma''_i) = 1$. Finally, $\mathcal{A}$ outputs a message-signature pair (m, σ') where (m, σ') is a message-signature pair* *not in L_{sig}. Then, for any such $\mathcal{A}$, we require that $\mathcal{A}$'s output satisfies*

$$\Pr[Verify(m, Extract(m, \sigma', sk_C, pk_C, pk_S), pk_S) = 1] < negl(\lambda).$$

The probability is taken over the coin tosses of the signer S, the adversary $\mathcal{A}$, and the key generation algorithms G_S and G_C.

Definition 3.6. Security for the confirmer (Invisibility): Intuitively, this means that no adaptive PPT adversary can distinguish between two designated confirmer signatures (or between a valid DCS and an invalid DCS). Consider the following game against a distinguisher $\mathcal{D}$:

Game-INV: *Firstly, Key Generation algorithms are run for the signer and the confirmer on input 1^λ. $\mathcal{D}$ is given pk_S and pk_C, which are the public keys of the signer and the confirmer, with the addition of signer's secret key sk_S. As a training purpose, $\mathcal{D}$ is allowed to create signature-key pairs ($sk_\mathcal{D}$,$pk_\mathcal{D}$) (not necessarily via Key Generations) and to interact with the confirmer with respect to these keys. Furthermore, $\mathcal{D}$ can make arbitrary queries to the following oracles: $Confirm_{(C,\mathcal{D})}$, $Disavow_{(C,\mathcal{D})}$, and Extract. Then, the distinguisher has to present two messages m_0 and m_1. After a fair coin b is flipped by the challenger, the distinguisher is given a corresponding DCS $\sigma' = DCSSign(m_b, sk_S, pk_C)$, where $b \in \{0,1\}$. Now $\mathcal{D}$ is again allowed to access the above oracles except that it cannot enquire for (m_0, σ') or (m_1, σ') (and their equivalent DCSs) via $Confirm$, $Disavow$, or $Extract$ oracle. Finally, the distinguisher must output one bit information b' to guess the value of b. The distinguisher wins if and only if $b = b'$, and $\mathcal{D}$'s advantage is defined as $adv_\mathcal{D} = \Pr[\mathcal{D}\ wins]$. We require:*

$$adv_\mathcal{D} < negl(\lambda).$$

The above probability is taken over the coin tosses of the signer S, the confirmer

C, and key generation algorithms G_S and G_C.

Remark 2. This security property can be generalised in the multi-signer settings, i.e. in the scenario of many signers sharing the same confirmer. That is, the adversary cannot break invisibility w.r.t a specified signer, even if it knows secret keys of other signers. To update the definition, one could simply add the secret keys of the other $n-1$ signers, say sk_{S_i} where $sk_{S_i} \neq sk_S$ and $1 \leq i \leq n-1$, as the adversary's auxiliary input.

Definition 3.7. Security for the confirmer (Non-transferability of $Confirm$ and $Disavow$ protocols): The evidence generated in $Confirm$ or $Disavow$ protocol should be untransferable. Namely, although an adaptive PPT adversary $\mathcal{A}$ knows whether a given DCS is valid or not through the interactive verification, it does not gain any knowledge that can be used to convince a third party about the validity of that DCS. In particular, this notion is formalised in the following games considering a simulator $\mathcal{A}'$:

Game-NTR: *Firstly, the adversary $\mathcal{A}$ is given the public key pk_S and pk_C of the signer and the confirmer. It is allowed to make arbitrary oracle queries to $DCSSign$, $Confirm_{(C,V)}$, $Disavow_{(C,V)}$ and $Extract$. Again $\mathcal{A}$ is allowed to create signature-key pairs $(sk_{\mathcal{A}}, pk_{\mathcal{A}})$, and to run $DCSSgin$ and then interact with the confirmer with respect to these keys.*

In some stage, the adversary must present two strings, m and σ', for which it wishes to carry out the $Confirm$ (or $Disavow$) protocol with the confirmer. Next a fair coin b is flipped. If $b = 0$, the real confirmer and $\mathcal{A}$ run the $Confirm$ (or $Disavow$) protocol with common input (m, σ', pk_S, pk_C), while the confirmer's secret input will be sk_C. If $b = 1$, the simulator Sim is plugged in the place of the confirmer to run the $Confirm$ (or $Disavow$) protocol on (m, σ'). Sim is not given the confirmer's secret key, but is allowed to make a single call to an oracle which tells Sim whether the strings m and

σ' is a valid DCS w.r.t. pk_S and pk_C.

In parallel, the adversary is allowed to make arbitrary queries to the signer and the confirmer. And in all other interactions except the confirmation (or disavowal) on (m, σ'), the real signer or the real confirmer speaks with the adversary. Finally, $\mathcal{A}$ must output one bit information b' to guess the value of b. The adversary $\mathcal{A}$ wins if and only if $b = b'$, and $\mathcal{A}$'s advantage is defined as $adv_{\mathcal{A}} = \Pr[\mathcal{A}\ wins]$. We require for any adversary $\mathcal{A}$, there exists a simulator Sim such that for all sufficiently large λ, all $(sk_S, pk_S) \in G_S(1^\lambda)$, and all $(sk_C, pk_C) \in G_C(1^\lambda)$:

$$adv_{\mathcal{A}} < negl(\lambda).$$

The above probability is taken over the coin tosses of the signer S, the confirmer C, and key generation algorithms G_S and G_C.

Remark 3. A further discussion about the definitions and security notions of DCS schemes is laid out in Chapter 4. In particular, we introduce two additional security notions, namely "unimpersonation" and "transcript-simulatability", and explore the relations among those notions.

3.2 Invisibility of Zhang et al.'s DCS Scheme

Considering the construction for DCS schemes, Okamoto [64] proposed a straightforward way using standard cryptographic primitives, i.e., public key encryptions and digital signature schemes. The signer firstly issues a standard signature on a target message m, then encrypts the signature using the confirmer's public key, and finally, the resulting ciphertext is preserved as the designated confirmer signature on m. To prove the validity of such a DCS, the signer has to interact with the verifier (signature recipient), i.e., the signer should prove that what the verifier obtained is indeed an encryption of a standard signature on m. In fact, that NP statement requires general zero-knowledge proofs. Zhang et al.'s scheme [89], as the first concrete implementation of the above paradigm, outperforms the previous schemes [45, 38] on both signature size and computational cost. However, we discover that their scheme has a vulnerability with regard to invisibility.

3.2.1 Review of the Scheme

We briefly describe their scheme, and note that their scheme provides neither signer's confirm-ability nor signer's disavow-ability.

Setup: Choose a bilinear map $e : G \times G \to G_t$, where G is a multiplicative cyclic group of prime order p and a generator g. G_t is another multiplicative cyclic group such that $|G| = |G_t| = p$. The system parameters are (G,G_t,e,p,g).

Key Generation: Signer randomly selects $x, y \in_R \mathbb{Z}_q^*$, and computes $u = g^x$, $v = g^y$, then sets its public key as (u, v), and its private key as (x, y). Confirmer chooses a random number x_c from $\mathbb{Z}_q^*$ as its private key, and computes its public key as $\alpha = g^{x_c}$.

Sign: This is the same as the signing algorithm in ZCSM scheme [88]. Given a message $m \in \mathbb{Z}_q$, signer picks a random $r \in_R \mathbb{Z}_q^*$, and computes $\sigma = g^{(x+my+r)^{\frac{1}{2}}} \in G$. Here $(x + my + r)^{\frac{1}{2}}$ is computed modulo q. The algorithm will try with different random values for r until $x + my + r$ is a quadratic residue modulo q. The signature on message m is (σ, r).

Verification: Given public parameters, a message $m \in \mathbb{Z}_q$, and a signature (σ, r), anyone can verify that if the equation $e(\sigma, \sigma) = e(uv^m g^r, g)$ holds or not.

DCSSign: Given confirmer's public key α, a message $m \in \mathbb{Z}_q$, signer picks a random $r \in_R \mathbb{Z}_q^*$, and computes $\sigma' = \alpha^{(x+my+r)^{\frac{1}{2}}} \in G$. The designated confirmer signature on message m is (σ', r).

Confirm: confirmer first checks that (σ', r) has been signed by the signer using its secret key x_c, i.e., it checks if the equation $e(\sigma', \sigma') = e(uv^m g^r, \alpha)^{x_c}$ holds or not. If it holds, confirmer performs an interactive zero-knowledge proof with the verifier for knowledge: $\log_{e(uv^m g^r, \alpha)} e(\sigma', \sigma') = \log_{e(g,g)} e(\alpha, g)$. This is an interactive zero-knowledge proof system for the equality of two discrete logarithms[22].

Disavow: To disavow a purported signature (σ', r) on m, confirmer performs an interactive zero-knowledge proof with the verifier for proving the discrete logarithm $\log_{e(uv^m g^r, \alpha)} e(\sigma', \sigma')$ and $\log e(\alpha, g)$ are unequal.

Extract: For (m, σ', r), confirmer can extract the ordinary ZCSM signature on m using its secret key α: $\sigma = \sigma'^{x_c^{-1}}$.

3.2.2 Mounting the Attack

In this Section, we mount an attack against the invisibility of Zhang et al.'s scheme. In this attack, if a verifier has already obtained two valid designated confirmer signatures, it will be able to verify the validity of any alleged designated confirmer signatures by himself.

In $Confirm$ protocol, a confirmer first validates the signature by the equation $e(\sigma', \sigma') = e(uv^m g^r, \alpha)^{x_c}$, of which the right half can be re-written into three parts:

$$e(\sigma', \sigma') = e(u^{x_c}, \alpha) \cdot e(v^{x_c}, \alpha)^m \cdot e(\alpha^r, \alpha). \qquad (*)$$

For each distinct signature, $e(u^{x_c}, \alpha)$ and $e(v^{x_c}, \alpha)$ are constant, while $e(\alpha^r, \alpha)$ can be calculated since α and r are public. Our target is to compute $e(u^{x_c}, \alpha)$ and $e(v^{x_c}, \alpha)$, so that the verifier can check the equation (*) by himself and determines the signature's validity. The technical details are described as follows.

1. Suppose the verifier already holds two valid designated confirmer signatures, (σ'_1, r_1) on message m_1 and (σ'_2, r_2) on message m_2. Firstly, it computes $A_1 = \frac{e(\sigma'_1,\sigma'_1)}{e(\alpha^{r_1},\alpha)}$, and $A_2 = \frac{e(\sigma'_2,\sigma'_2)}{e(\alpha^{r_2},\alpha)}$.

2. According to equation (*), the equation $\frac{e(\sigma',\sigma')}{e(\alpha^{r},\alpha)} = e(u^{x_c}, \alpha) \cdot e(v^{x_c}, \alpha)^m$ holds, then the verifier computes $\frac{e(v^{x_c},\alpha)^{m_1}}{e(v^{x_c},\alpha)^{m_2}} = \frac{A_1}{A_2}$. So, he can obtain $e(v^{x_c}, \alpha) = (\frac{A_1}{A_2})^{\frac{1}{m_1-m_2}}$, where $\frac{1}{m_1-m_2}$ is the inverse of $(m_1 - m_2)$ modulo q.

3. The verifier can compute $e(u^{x_c}, \alpha) = \frac{A_1}{e(v^{x_c},\alpha)^{m_1}}$ after it gets the value of $e(v^{x_c}, \alpha)$.

4. Finally, as the verifier knows the values of $e(u^{x_c}, \alpha)$ and $e(v^{x_c}, \alpha)$, it can validate any DCS signatures without the confirmer by checking equation (*).

According to Goldwasser and Waisbard's security model [45] cited in [89], the defined security requirement for designated confirmer only covers "unimpersonation" but excludes "invisibility" and "non-transferability". Note that this attack breaks invisibility consequently. We also found the security proof for Lemma 1 and proof for Theorem 3 in [89] are incomplete.

For Lemma 1, the proof is incomplete because there were no analysis of oracle simulation by the challenger, since they only mentioned the forger $\mathcal{F}$ could access q_{DCS} times *DCSSign*, q_C times *Confirm* and q_D times *Disavow* oracle adaptively, and output a forged message-signature pair, which could be imposed by another forger $\mathcal{F}'$ that aims to break the unforgeability of underlying ZCSM signature scheme [88]. Nonetheless, we can simply make up the simulation as below:

if $\mathcal{F}$ requests a DCS from ConfirmedSign oracle, the challenger $\mathcal{F}'$ can send the same query to its own Signing oracle which will issue a ZCSM signature (σ, r), then $\mathcal{F}'$ responds to the $\mathcal{F}$ with a corresponding DCS $(\sigma', r) = (\sigma^{x_c}, r)$. Note that $\mathcal{F}'$ knows confirmer's secret key x_c so that it can check any alleged DCS's validity, because we assume the confirmer could be corrupted.

if $\mathcal{F}$ requests a Confirmation (Disavowal) via *Confirm* (*Disavow*) oracle, $\mathcal{F}'$can send a correct response by using x_c.

if $\mathcal{F}$ requests a extraction for a DCS, $\mathcal{F}'$can send a correct ZCSM signature by using x_c.

For Theorem 3, They did not give the full oracle simulations to the adversary before the challenge, i.e. the algorithm $\mathcal{B}$ which is a challenger to solve the Computational Diffie-Hellman Problem, only provides $ConfirmedSign$ oracle to the adversary. Meanwhile, $\mathcal{B}$ should be able to simulate the other oracles including $Confirm$, $Disavow$ and *Extract*, to convince the adversary. We find the attack does not comply with the not-complete proof, because according to "soundness" property of zero-knowledge proofs, since $\mathcal{B}$ is not able to convince the adversary for the corresponding zero-knowledge proofs without the secret, and it cannot run $Confirm$ or $Disavow$ protocol in fact.

3.3 Invisibility of Wei et al.'s DCS Scheme

Wei, Zhang, and Chen proposed a new and interesting concept of society-oriented designated confirmer signature scheme (SDCS) [83] recently. They extends the standard DCS into a "group-based" DCS via threshold cryptography. In their proposition, there is a "signer group" consists of n individuals, and a "confirmer group" consists of l individuals. To issue a society-oriented designated confirmer signature , at least t members of the signer group form a "signing group", and cooperate to produce a SDCS for the receiver. To validate the signature, at least k members of the confirmer group form a legitimate "confirming group", and cooperate to provide the validity proof. In particular, there is a “signing combiner” whose task is to choose and encrypt some random values, compute the commitments, and collect all partial results during the signing phase. Analogically, a “confirming combiner” collects partial witnesses produced by the k confirmers and outputs the final validity proof of the alleged signature.

However, we find their concrete scheme fails to meet invisibility (Note that, this attack also applies to a similar paper [82] by Wei et al.’s). Recall that, a secure DCS scheme should meet invisibility informally requires that, if given a DCS σ', no (adaptive) adversary can distinguish the signed source between message m_0 and message m_1 with non-negligible advantage better than 1/2.

3.3.1 Review of the Scheme

To depict the attack, we give a brief introduction about Wei et al.’s scheme without redundant maths.

System Parameters Generation: Given the security parameter λ, system parameters are produced as $SP = (P, G, g, h, n, t, l, k, N, M, H, u, v; p, q)$. G is a cyclic group with order P. g, h are two random generators of G. The signer group has n members, while at least t out of which are required to sign a message. The confirmer group has l members , while at least k out of which are required to confirm a signature’s validity. N is a typical RSA modulus, where it is the product of two 512-bit secure primes: $N = pq$, $p = 2p' + 1$ and $q = 2q' + 1$, and p', q' are also primes. M is the product of p' and q'. H is a collision-free hash function $\{0, 1\}^* \rightarrow \{0, 1\}^{1024}$. v is a random generator in the quadratic residue group QR_N and u is an element in Z_N^* whose Jacobi symbol with respect to N is -1.

Key Generations: Given system parameters SP, keys related to all players in the scheme are generated as follows. $n < e < min(p', q')$ is the public prime exponent for the singer group while d is the private signing key. $d_i = f(i)(n!)^{-1}$ mod M, $(i =$

$1, 2, ..., n)$ is the signing-key share ($SK_{S_i} = d_i$) for the $i - th$ signer, where a random polynomial $f(x) = \sum_{i=0}^{t-1} a_i x^i \in Z_P[x]$, $a_0 = d$, and $a_i \in_R Z_M^*(i = 1, 2, ..., t-1)$. $v_i = v^{d_i}$ mod $N(i = 1, 2, ..., n)$ is the public verification value ($PK_{S_i} = v_i$) associated with d_i. $(SK_{C_i}, PK_{C_i})(i = 1, 2, ..., l)$ is the private/public key pairs of the $i - th$ confirmer, where $SK_{C_i} \in_R Z_P^*$ and $PK_{C_i} = g^{SK_{C_i}}$.

SDCS Generation: For a message $m \in Z_P^*$ to be signed, the signing combiner interacts with the members of the signing group and issues a SDCS σ^* in the following steps,

1. the signing combiner firstly computes a Pedersen commitment [67] $\varphi = g^m h^r$ by a random value $r \in_R Z_P^*$;

2. Then it computes the ciphertext c of the randomness r using the public keys $PK_{C_1}, ..., PK_{C_i}$ of l members of the confirming group as $c = Enc_{PK_{C1},...PK_{C_l}}(r) = (R, \omega_1, ...\omega_l)$, where $R = g^r$, $\omega_j = F(PK_{C_j}^r)$, Enc is an encryption function in a (k, l) threshold encryption scheme [49]. Another random polynomial $F(x)$ is defined as $F(x) = \sum_{i=0}^{k-1} b_i x^i \in Z_P[x]$, where $b_0 = r$ and $b_1, ..., b_{k-1} \in_R Z_P^*$.

3. The signing combiner secretly sends a tuple (m, φ, c) to each member of the signing group with a proof $PK\{(\alpha) : \varphi = g^m h^\alpha \wedge R = g^\alpha \wedge \omega_1 = PK_{C_1}^\alpha \wedge ... \wedge \omega_l = PK_{C_L}^\alpha\}$.

4. Each member of the signing group validates the proof of (m, φ, c), and compute $m' = H(\varphi \parallel c)u^{(1-J(\frac{H(\varphi\|c)}{N}))/2}$ where $J(.)$ is the Jacobi symbol. A partial signature is also computed as $\sigma_i = m'^{2d_i}$. Then they send them secretly back to the signing combiner with the proofs $PK\{(\beta) : v_i = v^\beta \wedge \sigma_i = m'^{2\beta}\}$.

5. Given at least t valid partial signatures, the signing combiner computes $\sigma = \prod_{i=1}^{t} \sigma_i^{2n! \prod_{j \in \{1,...,t\}, j \neq i} \frac{j}{j-i}}$ mod N. The it publishes the SDCS on message m as

$\sigma^* = (\varphi, c, \sigma)$. Note this step and step 4 are actually the generation of a (t, n) threshold signature [79].

SDCS Confirmation/Disavowal: To validate a SDCS σ^* on a message m, the verifier interacts with the confirming group as follows,

1. The verifier checks σ is a valid signature on m' by checking the equation $\sigma^e = m'^4$. Then it sends the SDCS σ^* with the message m to k members of the confirming group that it trusts.

2. Each member of the confirming group, can compute $c = R^{SK_{C_j}}$ using their private key SK_{C_j}, and sends it to the confirming combiner with a proof $PK_{C_j}\{(\gamma) : c_j = R^\gamma \wedge PK_{C_j} = g^\gamma\}$.

3. The confirming combiner computes a value

$$r' = \sum_{j=1}^{k} \omega_j \prod_{t \in \{1,\ldots,k\}, t \neq j} \frac{c_t}{c_t - c_j}.$$

   It checks the equation $R = g^{r'}$. If it does not hold, c_j and $PK_{C_j}\{(\gamma) : c_j = R^\gamma \wedge PK_{C_j} = g^\gamma\}$ are sent to the verifier who will decrypt the ciphertext himself and be convinced that it has submitted an invalid ciphertext, the terminate the procedure. Otherwise, the confirming combiner performs a bi-proof that interacts with the verifier, to prove the equality $log_g R = log_h \frac{\varphi}{g^m}$ or the inequality $log_g R \neq log_h \frac{\varphi}{g^m}$.

4. The verifier will be convinced that (m, σ^*) is a valid message-SDCS pair if the equality of $log_g R$ and $log_h \frac{\varphi}{g^m}$ holds. Otherwise, it will know σ^* is not a valid SDCS on m for the inequality of those two discrete logarithms. Here we omitted the technical details of proving the equality or inequality of two discrete logarithms, which is proposed in Section 3.2 [57].

3.3.2 An Attack on Invisibility

The successful launch of the attack depends on a security flaw that the ciphertext c of randomness r can be re-used in different signatures.

For instance, a SDCS $\sigma^* = (\varphi, c, \sigma)$ is signed on a message m_b, where $b = \{0, 1\}$, $\varphi = g^m h^r$, c = $Enc(r, PK_{C_1}, ..., PK_{C_l})$ and σ as described in the previous protocols. An (adaptively chosen message) adversary $\mathcal{A}$ given σ^*, is to guess whether this signature is signed on m_0 or m_1, which are two messages selected by himself before the challenge. In addition, $\mathcal{A}$ is also given the public/private key-pairs of the signer group, and the public keys of the confirmer group according to the definition of security for the confirmers in [83]. The attack can be launched as below:

1. The adversary picks a new message $\overline{m}$ that $\overline{m} \neq m_0$, and $\overline{m} \neq m_1$.

2. It calculates $\overline{\varphi} = \varphi g^{\overline{m}} g^{-m_0}$, which again equals $g^{\overline{m}+m_b-m_0} h^r$.

3. $\mathcal{A}$ generates a SDCS on $\overline{m}$, i.e. $\overline{\sigma^*} = (\overline{\varphi}, c, \overline{\sigma})$. This procedure is reasonable because the adversary holds all private keys of members of the signing group, and it can simulate the SDCS generation phase by himself. More specifically, the generation of threshold signature $\overline{\sigma}$ does not require the knowledge of randomness r in [79].

4. After that, $\mathcal{A}$ inquires $\mathcal{O}_{\mathcal{V}}$ for message-signature pair $(\overline{m}, \overline{\sigma^*})$. Note the verification oracle $\mathcal{O}_{\mathcal{V}}$ is provided in the security model in[83], which on input a message-signature outputs whether or not it is correct w.r.t. the private keys of the signing group and the public keys of the confirming group.

5. Finally, $\mathcal{A}$ outputs $b = 0$ if the response from $\mathcal{O}_{\mathcal{V}}$ is "correct", i.e. $g^{\overline{m}+m_b-m_0} h^r = g^{\overline{m}} h^r$. Otherwise, it outputs $b = 1$. It is clear that $\mathcal{A}$ can distinguish the two messages with probability 1.

It is straightforward to check the correctness of the above attacks.

3.3.3 A Repair

Now we consider how to fix the above security flaw in Wei et al.'s SDCS scheme. The basic idea is that we should let the confirming combiner know the "context" of ciphertext c, i.e., for which message and with respect to which users it is created. To this end, we can use public encryption with "labels" by taking the context information as a label. For example, we may use the Paillier based CCA2-secure encryption with labels introduced by Camenisch and Shoup [19]. More specifically, to use this encryption scheme, the signer group can define label $L = m||PK_{S1}||\cdots||PK_{Sn}||PK_{C1}||\cdots||PK_{Cl}$ when they issues a SDCS for message m w.r.t. the confirmer group with public keys of $PK_{C1}\cdots PK_{Cl}$. Beside this modification, all the procedures are the same as in the Wei et al.'s original SDCS scheme, though all zero-knowledge proof should be given with the context of label L. So, this improvement is compatible with the original SDCS construction, maintains its efficiency, and also overcomes the above security flaw against invisibility.

Chapter 4

A Theoretical Analysis of Security Model

As mentioned in the previous chapter, the central security property of a designated confirmer signature scheme is *invisibility* [18], which requires that any probabilistic polynomial time (PPT) adversary cannot feasibly determine the (in)validity of an alleged signature. That is, the (in)validity of an alleged signature is invisible to a verifier so that the only way to check this is to interact with either the signer or the designated confirmer. However, in the literature researchers have also proposed two other related properties, namely *unimpersonation* [64, 45] and *transcript simulatability* [38, 78]. Intuitively, unimpersonation requires an attacker cannot impersonate either the signer or the confirmer to run the given interactive protocols with a verifier to validate a signature. Transcript-simulatability requires that the transcripts (i.e. evidence) generated in those interactive protocols should be simulatable. If such transcripts were forwarded, rather than those generated via directly interacting with the real prover, i.e the signer or the confirmer, they cannot convince the other party and show the (in)validity of the alleged signature. Eventually, a notion named *non-transferability* is introduced in [18], which can be seen as a simplified version of transcript-simulatability that give some restrictions about verification protocols, and it informally means that one cannot get more information out of verification protocols than whether a signature is valid or not. The relations between these four properties are not clear and have never been **formally discussed** after they were proposed.

In cryptography, it is necessary to study the relations between different cryptographic primitives or different definitions of the same or similar security properties. The most famous example is probably the equivalence of two definitions on the security of public key encryption against adaptive chosen ciphertext attack (CCA2) [69], i.e., the IND-CCA2 and the NM-CCA2 [4]. Namely, CCA2 can be equivalently formalised in the context of either indistinguishability (IND) or non-malleability (NM). The importance of such a question is twofold. On the one hand, if the equivalence of different notions is known, a designer is free to choose any of them to prove the security for any given cryptosystem, according to the features of the scheme analysed and/or his/her preference and familiarity. Namely, such a result increases the flexibility of security analysis and scheme designs. On the other hand, if we know that some similar notions are actually not the same, then we should try to construct cryptosystems which are secure w.r.t. stronger or even the strongest security notion, as such a scheme effectively satisfies all weaker properties as well. So, our productivity is improved.

In this chapter, we classify these different security notions under proper assumptions, associated with both intuitive discussions and formal security proofs. After an introduction to the theoretical background, we first discuss the relations between invisibility and unimpersonation with a formal proof presented. Then, as a rather purely theoretical interest, we examine the concept of transcript-simulatability, and try to find if this new notion in Gentry et al's model [38] is stronger than or equivalent to any previous notions, namely invisibility or non-transferability in Camenisch and Michels' model [18]. The result seems interesting: On one hand, if any DCS scheme satisfies both invisibility and non-transferability, it naturally satisfies transcript-simulatability. On the other hand, however, the result slightly changes, that is, if any DCS scheme satisfies transcript-simulatability, a weakened notion of non-transferability and invisibility is guaranteed.

4.1 Different Security Notions

Unforgeability, as introduced in the previous chapter, is an essential notion to guarantee the signer's security. However, considering the confirmer's security, we discover that it contains more requirements like invisibility and non-transferability, as stated in the previous chapter. In fact, there exist other dimensions to define the confirmer's security, and we introduce two of them in this chapter, namely *unimpersonation* and *transcript-simulatability*, which were both appeared in the previous literature of DCS schemes.

4.1.1 A Weak Security Notion: Unimpersonation

Of course, for a DCS scheme to be secure, it should be infeasible for an adversary to impersonate the confirmer. Okamoto's model [64] firstly captures *"unimpersonation"* in a formal way, and is later complied by Goldwasser and Waisbard's transformation [45]. However as pointed in Camenisch and Michels's elucidation (the second paragraph of subsection 2.2 in [18]), "his model defines a weaker notion of security of the confirmer: the adversary knowing the signer's secret key wins the game only if it is able to behave like the confirmer, i.e., to confirm and disavowal signatures, but does not win the game if it can distinguish between two confirmer signatures (or between a valid and an invalid confirmer signature)". We agree with their comments, and give the grounds in a latter discussion by showing that invisibility actually implies unimpersonation. We present the formal definition of "unimpersonation of the confirmer" (also see in Figure 4.1, where the public parameter π is shorthand for $(1^\lambda, pk_S, pk_C)$, and KG is a abbreviation of the key generation algorithm) developed from [45] with some changes (see the discussion in Remark 2).

Definition 4.1. Security for the confirmer (Unimpersonation) *Let $\mathcal{I}$ be a PPT impersonator. On given input the public keys pk_S and pk_C under the security parameters 1^λ, $\mathcal{I}$ enters the learning phase that it can request the executions of $\mathcal{O}$ oracle, including $DCSSign$, $Confirm_{(C,\mathcal{I})}$, $Disavow_{(C,\mathcal{I})}$ and $Extract$ for polynomially many times on the inputs of its choice. At the end of learning phase, $\mathcal{I}$ must output a pair (m, σ') of its choice and an additional bit coin, where $coin = 1$ indicates that it is a valid message-DCS pair, and $coin = 0$ indicates that it is an invalid one. In the impersonation phase, $\mathcal{I}$ executes the $Confirm_{(C,V)}$ protocol as the prover if $coin = 1$. Otherwise, it executes the $Disavow_{(C,V)}$ protocol as the prover. The impersonator $\mathcal{I}$ wins if and only if:*

$$Confirm_{(\mathcal{I},V)}(m, \sigma', pk_S, pk_C) \rightarrow 1 \; if \; coin = 1$$
$$or \; Disavow_{(\mathcal{I},V)}(m, \sigma', pk_S, pk_C) \rightarrow 1 \; if \; coin = 0$$

$\mathcal{I}$'s advantage in this game is defined to be $adv_{\mathcal{I}} = \Pr[\mathcal{I} \; wins]$. We say a DCS scheme is secure for the confirmer iff $adv_{\mathcal{I}}$ is a negligible function after executing the above game. The above probability is taken over all possible coins used by $\mathcal{I}$, S, C, V, and the key generation algorithms G_S and G_C. Also this requirement should hold when many signers share the same confirmer. Namely, when $\mathcal{I}$ knows polynomially many secret key sk_{S_j} such that $sk_{S_j} \neq sk_S$.

Remark 1. The given definition is different from the definitions in Okamoto's model[64]. In [64], the $Disavow$ protocol is integrated in the verification process with the $Confirm$ protocol, while our model separates the verification in two different protocols. Also our definition is weaker, since the definition in [64] allows the adversary to have the signer's secret key.

Remark 2. The given definition is also slightly different from the Goldwasser and Waisbard's definition [45]. In [45], the disavowal case is overlooked, namely the adversary should not succeed in executing $Disavow_{(C,V)}$ protocol when its chosen challenging DCS is invalid. Also we replace t the adversary's $ConfirmedSign$ oracle with a $DCSSign$ oracle.

Remark 3. The previous DCS models ([64] and [45]) require the challenge message-DCS pair **not** necessary to be fresh. In other words, the adversary can still use the same pair as the one involved in previous oracle queries.

Remark 4. Note that in the above definition, to win the game the adversary has to run the same $Confirm$ protocol specified in the given DCS scheme. So, from the viewpoint of a verifier, such an attacker is actually impersonating the role of the confirmer. That is the reason why the security definition is called "unimpersonation", rather than "security for designated confirmers" [45].

Remark 5. A simultaneous notion is "unimpersonation of the signer" in confirming (disavowing) a designated confirmer signature. Note this is not equal to forge a valid DCS. It is naturally involved when the scheme supports a full verification, namely, even the signer can confirm and disavow any designated confirmer signatures. The formal definition of "unimpersonation of the signer" is quite straightforward, as based on the definition 4.1, that one only needs to add the oracle access with $Confirm_{(S,\mathcal{A})}$and $Disavow_{(S,\mathcal{A})}$, and later $\mathcal{A}$ should impersonate as the signer in $Confirm_{(S,V)}$ or $Disavow_{(S,V)}$ protocol as a challenge. However, most of the previous schemes are not consistent with the signers disavowal ability, and thus we only refer to the confirmer's security here.

```
1. $(pk_S, sk_S, pk_C, sk_C) \leftarrow KG(1^\lambda)$
2. $(m, \sigma', coin) \leftarrow \mathcal{I}^{\mathcal{O}}(\pi)$
3. if $coin = 1$, $b \leftarrow Confirm_{(\mathcal{I},V)}(\pi, m, \sigma')$
   else, $b \leftarrow Disavow_{(\mathcal{I},V)}(\pi, m, \sigma')$
4. $\mathcal{I}$ wins iff $b = 1$.
```

Figure 4.1: Impersonation Game $Game_{UNIMP}$

Invisibility implies Unimpersonation. The soundness of the confirm and disavow protocols intuitively captures the requirement that "a prover cannot cheat" when interacting with a verifier in the $Confirm$ or $Disavow$ protocol i.e. it cannot convince a verifier that a signature is both valid and invalid. However, it does not prevent a third party from impersonating the prover.

Camenisch and Michels [18] claimed that as Okamoto's model [64] only covers unimpersonation for the confirmer's security, a scheme secure in Okamoto's model may suffer from adaptive signature-transformation attack that violates invisibility. Such a counter example is further given in Section 2.2 of [18]. Therefore, we can conclude that Okamoto's model [64] is weaker than Camenisch and Michels' model [18] (CM model), as it only captures "unimpersonation", and excludes invisibility. Hence, any DCS scheme which is secure in Okamoto's model may suffer from "adaptive signature-transformation attacks". Intuitively, such an adaptive signature-transformation adversary can transform a challenge DCS with respect to a given signing key into another DCS with respect to another signing key such that the resulting signature is valid if and only if the original signature is valid. In other words, this attack relates to invisibility. Note that in CM model, the adversary is allowed at anytime to create additional signature-key pairs, and to interact with the confirmer with respect to these keys. Detailed description of this attack can be acquired in [18], Section 2.3.

Though Camenisch and Michels have mentioned that unimpersonation is a weaker notion than the security formalised in their model, it is still worthwhile to give further exploration for the implicity behind. Invisibility intuitively requires no adaptive adversary can distinguish between a valid DCS and an invalid DCS if both are signed on the same message. We observe that if the adversary can impersonate the confirmer by successfully convincing any verifier of a DCS via any interactive protocol, it must trivially know the signature's validity. Hence, this means that an attacker can break unimpersonation it can also break invisibility. We present a formal proof for the following theorem. In addition, Figure 4.2 is to demonstrate the invisibility game in Def. 6 in Chapter 3, where $\sigma' \notin L_{\mathcal{O}}$ means (m_0, σ') or (m_1, σ') (and their equivalent DCSs) should not be queried in $\mathcal{O}$, including $Confirm$, $Disavow$, and $Extract$ oracle.

Theorem 4.1. Let $\mathcal{DCS}$ be a designated confirmer signature scheme, if $\mathcal{DCS}$ satisfies invisibility as defined in Def. 3.6 in Chapter 3, then the scheme satisfies unimpersonation as specified in Def. 4.1, that is, no PPT algorithm $\mathcal{A}$ can impersonate as the confirmer with a non-negligible probability, under the adaptive chosen message attacks.

Proof: Considering a PPT algorithm $\mathcal{A}$ executes the unimpersonation game, if $\mathcal{A}$ can break $\mathcal{DCS}$ by impersonating the confirmer in performing $Confirm_{(\mathcal{A},V)}$ or $Disavow_{(\mathcal{A},V)}$ protocol as a prover, with a non-negligible probability ε and within a polynomial number q of oracle queries, we shall construct another adversary $\mathcal{B}$ that breaks invisibility of $\mathcal{DCS}$ by winning the invisibility game with a non-negligible probability. In the security proof, $\mathcal{B}$ should simulate the environment for $\mathcal{A}$ about its permitted oracle queries.

At the beginning of the executions, $\mathcal{B}$ receives the public keys of the signer and the confirmer, pk_S and pk_C, which are generated via key generation algorithm. In addition, $\mathcal{B}$ receives the signer's secret key sk_S, and thus can simulate the $DCSSign$ oracle by himself. Then $\mathcal{B}$ forwards the signer and the confirmer's public keys to $\mathcal{A}$.

Next $\mathcal{B}$ runs $\mathcal{A}$ as a subroutine, and answers all $\mathcal{A}$'s oracle queries as follows. According to the definition 3.6 in Chapter 3, $\mathcal{B}$ is allowed to access $Confirm$, $Disavow$ and $Extract$ oracles. Thus for $Confirm_{(C,\mathcal{A})}$, $Disavow_{(C,\mathcal{A})}$ and $Extract$ oracle queries, $\mathcal{B}$ relays $\mathcal{A}$'s related queries and the corresponding responses between $\mathcal{A}$ and $\mathcal{B}$'s own challenger. For $DCSSign$ oracle queries, since $\mathcal{B}$ possesses sk_S, it simulates $DCSSign$ oracle for $\mathcal{A}$ by himself.

Before the simulation, $\mathcal{B}$ tries to guess which message will be selected by $\mathcal{A}$ in its $DCSSign$ oracle and later to perform $Confirm$ or $Disavow$ protocol in $\mathcal{A}$'s challenge with respect to that message. In particular, $\mathcal{B}$ should select a random index j such that $1 \leq j \leq q$ in advance. This fixed index j is used to guess a message m_j which will be asked in $DCSSign$ oracle queries and later used by $\mathcal{A}$ to break unimpersonation. Accordingly, once a message m_j is asked in $DCSSign$ oracle, $\mathcal{B}$ will set m_j as one challenging message for its own invisibility game and randomly select another challenging message m'. After submitting these two messages, $\mathcal{B}$ will get a DCS σ^* from its challenger. Without loss of generality, we assume that after flipping a fair coin b, $\mathcal{B}$'s challenger always signs on m_j if $b = 0$, and it signs on m' if $b = 1$. Now $\mathcal{B}$ forwards σ^* to $\mathcal{A}$ as the response to $\mathcal{A}$'s $DCSSign$ query on message m_j.

Eventually, $\mathcal{A}$ outputs a fresh pair $(m_{\mathcal{A}}, \sigma_{\mathcal{A}})$ for which it wants to carry out the $Confirm$ or $Disavow$ protocol as the prover.

Without loss of generality, we assume $\mathcal{B}$ is plugged into the verifier's place by honestly executing the protocol $Confirm_{(\mathcal{A},\mathcal{B})}$ or $Disavow_{(\mathcal{A},\mathcal{B})}$ on the pair $(m_{\mathcal{A}}, \sigma_{\mathcal{A}})$. Now $\mathcal{B}$ tries to guess whether (m_j, σ^*) or (m', σ^*) is a valid message-DCS pair by using the following strategy:

- Case 1: If $\mathcal{A}$ successfully performs $Confirm_{(\mathcal{A},\mathcal{B})}(m_{\mathcal{A}}, \sigma_{\mathcal{A}})$, and $(m_{\mathcal{A}}, \sigma_{\mathcal{A}})$ is an equivalent DCS of (m_j, σ^*) where $m_{\mathcal{A}} = m_j$, $\mathcal{B}$ outputs 0 as its guess to the value of b. This suggests that $\mathcal{B}$ made a correct guess, and thus it succeeds to find (m_j, σ^*) is valid, while (m', σ^*) is invalid.

```
1. $(pk_S, sk_S, pk_C, sk_C) \leftarrow KG(1^\lambda)$
2. $(m_0, m_1) \leftarrow \mathcal{A}^{\mathcal{O}}(\pi, sk_S)$
3. $b \overset{R}{\leftarrow} \{0, 1\}$
4. if $b = 0$, $\sigma' \leftarrow DCSSign(\pi, m_0, sk_S)$
   else, $\sigma' \leftarrow DCSSign(\pi, m_1, sk_S)$
4. $b' \leftarrow \mathcal{A}^{\mathcal{O}}(m_0, m_1, \sigma')$
5. Return 1 iff $b = b'$ and $\sigma' \notin L_{\mathcal{O}}$.
```

Figure 4.2: Invisibility Game $Game_{INV}$

- Case 2: If $\mathcal{A}$ successfully performs $Disavow_{(\mathcal{A},\mathcal{B})}(m_{\mathcal{A}}, \sigma_{\mathcal{A}})$, and $(m_{\mathcal{A}}, \sigma_{\mathcal{A}})$ is an equivalent DCS of (m_j, σ^*) where $m_{\mathcal{A}} = m_j$, $\mathcal{B}$ outputs 1 as its guess to the value of b. This suggests that $\mathcal{B}$ made a correct guess, and thus it succeeds to find (m_j, σ^*) is invalid, while (m', σ^*) is valid.

- Case 3: If $\mathcal{A}$ has never asked as many as j messages or did not use (m_j, σ^*) or its equivalent DCS to run $Confirm$ or $Disavow$ protocol by impersonating confirmer's role, which suggests $\mathcal{B}$ fails in its simulation, and it outputs a random bit as its guess to the value of b'. Indeed, in this case, $\mathcal{B}$ has no hope to relate the validity of its challenging signatures to $(m_{\mathcal{A}}, \sigma_{\mathcal{A}})$.

Apparently, $\mathcal{B}$ wins the invisibility game with a non-negligible advantage if either Case 1 or Case 2 happens. One may note the probability that, $(m_{\mathcal{A}}, \sigma_{\mathcal{A}})$ is an equivalent DCS of (m_j, σ^*) where $m_{\mathcal{A}} = m_j$, is $\frac{1}{q}$. The reason is that, apart from the DCS pairs acquired from $\mathcal{B}$'s $DCSSign$ oracle, $\mathcal{A}$ has negligible probability to solely construct any valid DCS pair unless unforgeability cannot be satisfied. And thus under the assumption that $\mathcal{A}$'s probability ε in successfully performing $Confirm_{(\mathcal{A},\mathcal{B})}(m_{\mathcal{A}}, \sigma_{\mathcal{A}})$ or $Disavow_{(\mathcal{A},\mathcal{B})}(m_{\mathcal{A}}, \sigma_{\mathcal{A}})$ is non-negligible, $\mathcal{B}$ wins the invisibility game with a non-negligible advantage $\frac{\varepsilon}{q}$. $\square$

```
1. $(pk_S, sk_S, pk_C, sk_C) \leftarrow KG(1^\lambda)$
2. $b \overset{R}{\leftarrow} \{0,1\}$
3. $(m, \sigma') \leftarrow \mathcal{A}^{\mathcal{O}}(\pi)$
4. if $b = 0$, $b_1 \leftarrow Confirm_{(C,\mathcal{A})}(m, \sigma')$,
   $b_2 \leftarrow Disavow_{(C,\mathcal{A})}(m, \sigma')$;
   else, $b_1 \leftarrow Confirm_{(Sim^{ValidityO},\mathcal{A})}(m, \sigma')$,
   $b_2 \leftarrow Disavow_{(Sim^{ValidityO},\mathcal{A})}(m, \sigma')$
5. $b' \leftarrow \mathcal{A}^{\mathcal{O}}$, and return 1 iff $b = b'$.
```

Figure 4.3: Non-transferability Game $Game_{NTR-0}$

```
1. $(pk_S, sk_S, pk_C, sk_C) \leftarrow KG(1^\lambda)$
2. $b \overset{R}{\leftarrow} \{0,1\}$
3. $(m, \sigma') \leftarrow \mathcal{A}_0^{O}(\pi)$
4. if $b = 0$, $b_1 \leftarrow Confirm_{(C,\mathcal{A}_1)}(m, \sigma')$,
   $b_2 \leftarrow Disavow_{(C,\mathcal{A}_1)}(m, \sigma')$,$\tau \leftarrow \mathcal{A}_1$;
   else, ,$\tau \leftarrow Sim^{ValidityO}(m, \sigma')$;
5. $b' \leftarrow \mathcal{A}_2^{O}(m, \sigma', \tau)$, and return 1 iff $b = b'$.
```

Figure 4.4: Non-transferability Game $Game_{NTR-1}$

```
1. $(pk_S, sk_S, pk_C, sk_C) \leftarrow KG(1^\lambda)$
2. $(m_0, m_1, s) \leftarrow \mathcal{A}_0^{\mathcal{O}}(sk_S, \pi)$
3. $b \overset{R}{\leftarrow} \{0,1\}$
4. $\sigma' \leftarrow DCSSign(\pi, m_b, sk_S)$
5. if $b = 0$, $\tau \leftarrow \mathcal{A}_1^{\mathcal{O}}(b, m_0, m_1, s, \sigma', \pi)$;
   else, $\tau \leftarrow Sim^{DCSSign}(b, m_0, m_1, s, \sigma', \pi)$
6. $b' = \mathcal{A}_2^{\mathcal{O}_{lim}}(\tau, m_0, m_1, \sigma', \pi)$
7. Return 1 iff $b = b'$ and $\sigma' \notin L_{ext}$.
```

Figure 4.5: Transcript-simulatability Game: $Game_{TS}$

4.1.2 Another Notion: Transcript-simulatability

At Asiacrypt 2005, Gentry et al. [38] introduced another security notion in DCS cryptosystems, namely "transcript-simulatability". This notion informally requires that the evidences of confirmation or disavowal of a DCS should be simulatable.

We present the formal definition of transcript-simulatability as below. The formulation follows the way of [38] and [78] except we use $DCSSign$ algorithm to replace $ConfirmedSign$ protocol, with related modified oracles .

Definition 4.2. Transcript-simulatability: *We say a DCS scheme is* transcript simulatable *if for any PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$ involved in the following game (see Figure 4.5), there exists a PPT algorithm Sim such that $\mathcal{A}$'s advantage w.r.t. Sim, i.e., correctly guessing the value of bit b, is negligible:*

$$adv_{\mathcal{A}} = |\Pr[Game_{TS}\ returns\ 1] - 1/2| \leq negl(\lambda).$$

The above probability is taken over all possible coins used by $\mathcal{A}_0$, $\mathcal{A}_1$, $\mathcal{A}_2$, Sim and key generation algorithms.

Game$_{TS}$: *π denotes the public parameters $(1^n, pk_C, pk_S)$. With access to all oracles in $\mathcal{O} = \{DCSSign, Confirm_{(C,\mathcal{A})}, Disavow_{(C,\mathcal{A})}, Extract\}$, algorithm $\mathcal{A}_0$ first outputs two messages m_0 and m_1. Then, a DCS σ' on m_b is output randomly by DCSSign algorithm, where b is a random bit generated by flipping a fair coin. After that, $\mathcal{A}_1$, Sim and $\mathcal{A}_2$ play the game in which Sim tries to make its output τ (when m_1 is signed) indistinguishable from $\mathcal{A}_1$'s output τ (when m_0 is signed); $\mathcal{A}_2$ with input $(\pi, m_0, m_1, \sigma', \tau)$ attempts to guess the value of bit b, i.e., distinguish whether m_0 or m_1 has been signed. In the game, $\mathcal{A}_1$ has access to all oracles in set $\mathcal{O}$, i.e., all oracles in $\mathcal{O}$ under the restriction $\sigma' \notin L_{ext}$ that both (m_0, σ') and (m_1, σ'), together with their "equivalent" DCSs, should not be queried to the $Extract$ oracle. Similarly, $\mathcal{A}_2$*

has access to oracles in $\mathcal{O}_{lim}$, i.e., all oracles in $\mathcal{O}$ with the restriction that $\mathcal{A}_2$ cannot make any query on (m_0, σ'), (m_1, σ'), or their "equivalent" DCSs. In contrast, Sim is given very limited oracle access set, i.e., it can make only q times of DCSSign queries as long as $\mathcal{A}_0$ makes at most q times of DCSSign queries. Finally, $\mathcal{A}_2$ outputs one bit b' as its guess to the value of b, i.e., whether m_0 or m_1 is signed. Let $adv_{\mathcal{A}}$ denote the advantage of the adversary $\mathcal{A}$.

Intuitively, algorithms $\mathcal{A}_1$, $\mathcal{A}_2$ and Sim represent verifier V_1, verifier V_2 and a simulation algorithm respectively. In the viewpoint of $\mathcal{A}_2$, to guess σ' is signed on which message, the transcript from a real verifier $\mathcal{A}_1$ is no more convincing or informative than the transcript from a simulation algorithm Sim.

Remark 6. One may note this security requirement is quite similar to "non-transferability" which is proposed in Camenisch and Michels' model (the definition can be found in [18], and also in Def.3.7 in Chapter 3). However, the definition of transcript-simulatability actually includes an implicit indistinguishability, namely the adversary cannot identify which is the valid message-signature pair, (m_0, σ') or (m_1, σ')? This motivate us to explore the relations between transcript-simulatability, non-transferability, and invisibility. A further detailed discussion for the relations between those three notions is carried out in the next section, and to make the comparison compatible, we make some changes to the DCS model. In particular, we assume only the confirmer can verify a DCS and interactively convince the verifier of the signature's validity. Hence, the "$ConfirmedSign$" protocol is replaced by the "$DCSSign$" algorithm where the latter is an algorithm that outputs a valid DCS. In fact, our change is just to disable the verifiability of the signer in GMR model [38] , and it does not affect the correctness of the result, as non-transferability actually means non-transferable of the confirmer's interactive verification. Because the signer and the confirmer's verifiablity are symmetric, we believe the comparison based only on the separate confirmer's verifiability is reasonable.

4.2 Is CM Model As Strong As GMR Model?

Two notions are introduced in CM model [18], i.e., invisibility and non-transferability. Recall that invisibility informally requires no adversary can see the (in)validity of an alleged signature, while non-transferability informally requires one cannot get more information out of the Confirm/Disavow protocol than whether a signature is valid or not. Note that, Figure 4.3 is to demonstrate the non-transferability game in Camenisch and Michels' definition [18], where Sim denotes the simulator in the non-transferability game, and $ValidityO$ denotes the single oracle that outputs a DCS's validity.

From the intuition, it seems the requirement of non-transferability may be covered by (or equivalent to) the requirement of transcript-simulatability. Meanwhile, by investigating the definition of invisibility and transcript-simulatability, we find the former can be derived from the later from the definitions, which means transcript-simulatability implies invisibility. Hence, one may raise an assumption: "Under proper conditions, transcript-simulatability is equivalent to invisibility plus non-transferability, that is, GMR model is as strong as CM model in some way. To address this issue, we tries to find a formal proof in two directions, i.e, one half about whether GMR model covers CM model, and the other half about whether CM model covers GMR model. However, we find it seems hard to deal with non-transferability in both directions, and we introduce another type of non-transferability, and we explain the related reasons as below.

4.2.1 A New Definition of Non-transferability

Normally, non-transferability guarantees that a verifier who learns whether a given DCS is valid or not by interacting with the confirmer in the confirm or disavow protocols, should not be able to prove this fact to a third party. More specifically, the verifier should be able to "fake" any evidence of the validity of a signature obtained by interacting with the confirmer. However, non-transferability in [18] and in Def 3.7. in Chapter 3, is actually defined based on an interaction-based indistinguishability, which means the adversary should not be able to tell whether it was interacting with a real confirmer or a simulation algorithm. We agree with the comments by Wikström that (see the discussion after Definition 8 in [85]), this requirement seems too strong, which relies on the existence of a straight-line zero knowledge simulator for an interactive proof without set-up assumptions. Hence we propose a new definition but preserves the main implications and applications of non-transferability as follows. For simplicity, we let NTR-0 denote the definition of non-transferability in [18] and Def 3.7. in Chapter 3. Consequently, we introduce the new definition of non-transferability.

Definition 4.3. Non-transferability of $Confirm$ **and** $Disavow$ **protocols** (NTR-1)**:** The evidence generated in $Confirm$ and $Disavow$ protocols should be untransferable. Namely, although an adaptive PPT adversary $\mathcal{A}$ knows whether a given DCS is valid or not through the interactive verification, it does not gain any knowledge that can be used to convince a third party about the validity of that DCS. Because there always exists an accompanying simulator of $\mathcal{A}$, which is able to produce the evidence that is indistinguishable from the true evidence. In particular, this notion is formalised in the following game considering a PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$ and a PPT simulator Sim:

Initially, the adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$ *is given the public key* pk_S *and* pk_C *of the signer and the confirmer.* $\mathcal{A}_0$ *is allowed to make arbitrary oracle queries to* $DCSSign$*,*

$Confirm_{(C,V)}$, $Disavow_{(C,V)}$ and $Extract$. Again $\mathcal{A}_0$ is allowed to create signature-key pairs $(sk_{\mathcal{A}}, pk_{\mathcal{A}})$, and to run $DCSSgin$ and then interact with the confirmer with respect to these keys. In some stage, $\mathcal{A}_0$ must present two strings, m and σ', for which it wishes to carry out $Confirm$ (or $Disavow$) protocol with the confirmer. Then a fair coin b is flipped.

If $b = 0$, the confirmer and $\mathcal{A}_1$ run the $Confirm$ (or $Disavow$) protocol with common input (m, σ', pk_S, pk_C), while the confirmer's secret input will be sk_C. Eventually, $\mathcal{A}_1$ stops with an output τ, where τ is the evidence shows that σ' is a valid (invalid) signature on m.

If $b = 1$, the PPT simulator Sim is involved in the game. Sim is not given the confirmer's secret key, but is allowed to make a single call to an oracle which tells Sim whether the strings m and σ' is a valid DCS w.r.t. pk_S and pk_C. Eventually, Sim stops with an output τ, where τ is the evidence shows that σ' is a valid (invalid) signature on m.

On receiving (m, σ', τ) with the public keys pk_S and pk_C, $\mathcal{A}_2$ outputs a bit b' to guess the value of b. In addition, A_2 is allowed to make arbitrary oracle queries to $DCSSign$, $Confirm_{(C,V)}$, $Disavow_{(C,V)}$ and $Extract$.

$\mathcal{A}$'s advantage relative to Sim in this game is defined as $adv_{\mathcal{A},Sim} = \Pr[b' = b] - \frac{1}{2}$. We require for any PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$, there exists a simulator Sim such that for all sufficiently large λ, all $(sk_S, pk_S) \in G_S(1^\lambda)$, and all $(sk_C, pk_C) \in G_C(1^\lambda)$, $adv_{\mathcal{A},Sim} \leq negl(\lambda)$.

Remark 7. Note that, Figure 4.3 and Figure 4.4 demonstrate the non-transferability games in the definitions of NTR-0 and NTR-1 respectively, where $ValidityO$ denotes the single oracle that outputs a DCS's validity.

Remark 8. It seems one may not easily to identify the relation between NTR-0 and NTR-1 for the following reasons. On one hand, it seems by reducing the requirement of distinguishing two interactions, to the requirement of distinguishing two pieces of evidences, NTR-1 seems weaker than NTR-0, as the simulator in NTR-1 could usually be a straight-line ZK simulator for an interactive proof without set-up assumptions, which is much harder to construct than a rewindable simulator in NTR-1. On the other hand, in NTR-1, the adversary $\mathcal{A}_1$ can append arbitrary information on the transcript from $Confirm_{(C,V)}(m, \sigma')$ if it wants, which requires the simulator to be more powerful regarding to a revisable transcript. And hence it remains an open problem to find the relations between these two security notions.

4.2.2 A Proof for One Side

For the first problem, i.e., whether GMR model covers CM model? We think the answer is *NOT SURE*, since NTR-0 in CM model seems too strong to achieve the computational security. However, if we replace NTR-0 with NTR-1 in CM model, we think the answer is *YES*. The reason is quite straightforward. By the informal above discussion, one could simply obtain the invisibility game by deleting algorithms A_1 and A_1' (i.e. Step 5), and deleting τ in the input of algorithm A_2. On the other hand, the challenge in the transcript-simulatability game essentially requires A_2 cannot distinguish between a true transcript (shows that σ' is signed on m_0) from a simulated one, even the later is on a false statement. Comparing against the similar challenge in the non-transferability game, such a requirement is clearly stronger, and thus we intuitively think transcript-simulatability also implies non-transferability. In fact, two results can be proved within the following theorem.

Theorem 4.2. Let $\mathcal{DCS}$ be a designated confirmer signature scheme which has transcript-simulatability, then $\mathcal{DCS}$ has invisibility and non-transferability as specified in Def 4.3.

Proof: We use the method of proof by contraposition, by proving the following two lemmas, i.e., Lemma 4.1 and Lemma 4.2. Let $\mathcal{A}_{INV}$, $\mathcal{A}_{NTR}$, $\mathcal{A}_{TS}$ be three adversaries which aim to win the invisibility game, non-transferability game, and transcript-simulatability game respectively. □

Lemma 4.1. If a PPT adversary $\mathcal{A}_{INV}$ breaks the scheme $\mathcal{DCS}$ on non-transferability by executing $Game_{INV}$ in polynomial time t_{INV} with a non-negligible advantage adv_{INV}, a PPT adversary $\mathcal{A}_{TS}$ can break the scheme $\mathcal{DCS}$ on transcript-simulatability by executing $Game_{TS}$ in polynomial time t_{TS} with a non-negligible advantage adv_{TS}.

Proof: We remark that because $Game_{INV}$ can be derived from $Game_{TS}$ by deleting algorithms $\mathcal{A}_1$ and $\mathcal{A}'_1$ (i.e. step 5 and 6), and deleting in the input τ of algorithm $\mathcal{A}_2$. Hence, an adversary in $Game_{TS}$ with more resource, i.e., additional transcripts τ, should have an advantage which is larger than a similar adversary in $Game_{INV}$ when guessing the random bit.

We build a PPT adversary $\mathcal{A}_{TS}$ by using $\mathcal{A}_{INV}$ as follows. The challenger of $\mathcal{A}_{TS}$ runs the key generation algorithm to generate the public parameters and the key pairs of the signer and the confirmer. Th adversary $\mathcal{A}_{TS}$ receives the public parameters including pk_S, pk_C from its challenger, and relays a copy to $\mathcal{A}_{INV}$.

In $\mathcal{A}_{INV}$'s training phase, all oracle queries and responses are relayed by $\mathcal{A}_0$, which is a subalgorithm of $\mathcal{A}_{TS}$ that has the full oracle access in $\mathcal{O}$. $\mathcal{A}_{INV}$ outputs two strings, say m_0 and m_1, which will be alternatively signed by its challenger.

$\mathcal{A}_{TS}$ uses m_0 and m_1 as its own challenging messages, and sends them to its challenger. At this point $\mathcal{A}_{TS}$'s challenger flips a coin to obtain a bit b and takes σ' as the DCS on m_b. After $\mathcal{A}_{INV}$ receives its challenge tuple (m_0, m_1, σ') from $\mathcal{A}_{TS}$, the executions follow $Game_{TS}$ in step 5. At the end of the procedure, either $\mathcal{A}_1$or $\mathcal{A}'_1$ outputs a transcript τ to convince $\mathcal{A}_2$ that σ' is signed on m_0.

$\mathcal{A}_{INV}$ is again allowed to enquire $\mathcal{O}$ via $\mathcal{A}_2$'s oracles. Eventually, $\mathcal{A}_{INV}$ outputs a bit b' as its guess on b, and $\mathcal{A}_{TS}$ uses b' as its own answer.

It is straightforwardly to find that if $\mathcal{A}_{INV}$ makes a successful distinction with a non-negligible advantage adv_{INV}, $\mathcal{A}_{TS}$'s advantage to break transcript-simulatability is non-negligible, and we have $adv_{TS} = adv_{INV}$. In addition, the running time of $\mathcal{A}_{TS}$ is equal to the running time of A_{INV}, i.e., $t_{INV} = t_{TS}$. □

Lemma 4.2. If there exists a PPT adversary $\mathcal{B}_{NTR} = (\mathcal{B}_0, \mathcal{B}_1, \mathcal{B}_2)$ breaks the scheme $\mathcal{DCS}$ on non-transferability by executing $Game_{NTR}$ in polynomial time t_{NTR} with a non-negligible advantage adv_{NTR}, then there exists a PPT adversary $\mathcal{A}_{TS}$ that breaks the scheme $\mathcal{DCS}$ on transcript-simulatability by executing $Game_{TS}$ in polynomial time t_{TS} with a non-negligible advantage adv_{TS}.

Proof: Essentially, we need to prove such a statement: if there exists an PPT adversary $\mathcal{B}_{NTR} = (\mathcal{B}_0, \mathcal{B}_1, \mathcal{B}_2)$, for arbitrary PPT simulator $\mathcal{B}'_1$, the output of $\mathcal{B}_1$ and the output of $\mathcal{B}'_1$ can be distinguished by $\mathcal{B}_2$, then there exists an PPT adversary $\mathcal{A}_{TS} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$, such that for arbitrary PPT simulator $\mathcal{A}'_1$, the output of $\mathcal{A}_1$ and the output of $\mathcal{A}'_1$ can be distinguished by $\mathcal{A}_2$.

we shall construct a transcript-simulatability adversary $\mathcal{A}_{TS} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$ by using the successful non-transferability adversary $\mathcal{B}_{NTR} = (\mathcal{B}_0, \mathcal{B}_1, \mathcal{B}_2)$ as a subroutine. In particular, $\mathcal{A}_{TS}$ should simulate all permitted oracles for $\mathcal{B}_{NTR}$ in $Game_{NTR}$ as follows.

Initially, the challenger of $\mathcal{A}_{TS}$ runs the key generation algorithm to generate the public parameters and the key pairs of the signer and the confirmer. Th adversary $\mathcal{A}_{TS}$ receives the public parameters including pk_S, pk_C from its challenger, and relays a copy to $\mathcal{B}_{NTR}$.

Because $\mathcal{A}_0$ is a subalgorithm of $\mathcal{A}_{TS}$ and has the full oracle access in $\mathcal{O}$, $\mathcal{A}_0$

relays all of $\mathcal{B}_0$'s $DCSSign$, $Confirm$, $Disavow$ and $Extract$ queries and responses. Meanwhile, $\mathcal{B}_0$ is allowed to create its own key pairs $(sk_{\mathcal{B}}, pk_{\mathcal{B}})$, and to run $DCSSgin$ and then interacts with $\mathcal{A}_0$ for the confirmation or disavowal with respect to these keys. Again, $\mathcal{A}_0$ still relays all $Confirm$ or $Disavow$ queries and responses. We allow $\mathcal{A}_0$ to maintain a list $L_0^{DCSSign}$ that $L_0^{DCSSign} = \{(m_i, \sigma_i') \mid (m_i, \sigma_i') \leftarrow DCSSign\}$ as $\mathcal{B}_0$'s all $DCSSign$ queries, where i does not exceed the polynomial number of $\mathcal{B}_0$'s oracle access in its training phase. Furthermore, $L_0^{DCSSign}$ will be added to the state information s and further sent to $\mathcal{A}_1$ or $\mathcal{A}_1'$.

At some point, $\mathcal{B}_0$ outputs two strings, $\overline{m}$ and $\overline{\sigma'}$, for which it wants to carry out the verification protocols. Let $\overline{\tau}$ denote the transcript of confirming or disavowing the DCS-pair $(\overline{m}, \overline{\sigma'})$ in the following simulation. $\mathcal{A}_0$ outputs two strings m_0 and m_1, where $m_0 = \overline{m}$, and m_1 is randomly selected from the message space. In addition, $\mathcal{A}_0$ adds $(m_0, \overline{\sigma'})$ to its states s.

Then the challenger of $\mathcal{A}_{TS}$ flips a fair coin b, and produces a DCS σ' on m_b. $\mathcal{A}_1$ or a PPT simulator $\mathcal{A}_1'$ executes as follows:

- Case 1 : If $b = 0$, $\mathcal{A}_1$ runs $Confirm_{(C,\mathcal{A}_1)}$ and $Disavow_{(C,\mathcal{A}_1)}$ protocols with C on the pair $(m_0, \overline{\sigma'})$, and records the transcript $\overline{\tau}$. $\mathcal{A}_1$ outputs $\overline{\tau}$ as the evidence showing σ' is signed on m_0, thought $\overline{\tau}$ is meaningless w.r.t. (m_0, σ').

- Case 2 : If $b = 1$, the executions are a little complex. Firstly, we assume any PPT simulator $\mathcal{A}_1'$ involved in the following executions is always willing to issue an indistinguishable output from the output in Case 1. The reason is that, we can define two types of simulator. The first-type simulator always issues a distinguishable output from the output in Case 1, while the second-type simulator always tries to issue an indistinguishable output from the output in Case 1. However, for the first-type simulator, $\mathcal{A}_2$ can always make a distinction and thus the statement is true. So without loss of generality, we simply assume any PPT simulator $\mathcal{A}_1'$ is the second-type simulator. $\mathcal{A}_1'$ constructs the simulated transcript $\overline{\tau}$ as follows. Since $\mathcal{A}_1'$ has access to $DCSSign$ oracle, it maintains all pairs $\{(m_j', \sigma_j')\}$ previously generated by $DCSSign$ as a list $L_1^{DCSSign}$, where $j \in \{1, .., q\}$. Also $\mathcal{A}_1'$ extracts $L_0^{DCSSign}$ from s.

- If $(\overline{m}, \overline{\sigma'}) \in L_0^{DCSSign}$, which means $(\overline{m}, \overline{\sigma'})$ is previously generated by $\mathcal{A}_0$'s $DCSSign$ oracle. $\mathcal{A}_1'$ marks "valid" on $(\overline{m}, \overline{\sigma'})$.
- If $(\overline{m}, \overline{\sigma'}) \in L_1^{DCSSign}$, which means the previously selected pair$(\overline{m}, \overline{\sigma'})$ is also a valid DCS generated by $\mathcal{A}_1'$'s $DCSSign$ oracle, and that is a rare case for a randomized algorithm. $\mathcal{A}_1'$ marks "valid" on $(\overline{m}, \overline{\sigma'})$.
- If $\overline{m} = m_i'$ and $R(\overline{m}, \sigma_i', \overline{\sigma}) = 1$, which means $\overline{\sigma}$ is an equivalent DCS of some signature σ_i' in $L_0^{DCSSign}$. $\mathcal{A}_1'$ marks "valid" on $(\overline{m}, \overline{\sigma'})$.
- If $\overline{m} = m_j'$ and $R(\overline{m}, \sigma_j', \overline{\sigma}) = 1$, which means $\overline{\sigma}$ is an equivalent DCS of some signature σ_j' in $L_1^{DCSSign}$. $\mathcal{A}_1'$ marks "valid" on $(\overline{m}, \overline{\sigma'})$.
- In all other cases, $\mathcal{A}_1'$ marks "invalid" on $(\overline{m}, \overline{\sigma'})$.

On receiving the validity information of $(\overline{m}, \overline{\sigma'})$, $\mathcal{A}_1'$ simulates a proper confirmation or disavowal transcript $\overline{\tau}$ on $(\overline{m}, \overline{\sigma'})$ as the evidence showing σ' is signed on m_0. Next, on receiving $\overline{\tau}$ from either $\mathcal{A}_1$ or $\mathcal{A}_1'$, $\mathcal{A}_2$ invokes $\mathcal{B}_2$ with the input $(\overline{m}, \overline{\sigma'}, \overline{\tau})$. In addition, $\mathcal{A}_2$ simulates $\mathcal{B}_2$'s oracle queries by using its own oracles. Eventually, $\mathcal{B}_2$ output a bit b', and $\mathcal{A}_2$ uses b' as its guess to the value of coin b.

We remark that, in Case 2, As long as $\mathcal{A}_1'$ is the second-type simulator, its simulation for $\overline{\tau}$ is indistinguishable from a real non-transferability simulator's execution. The reason is that, in the real non-transferability game, the simulator is equipped with a validity oracle once to know $(\overline{m}, \overline{\sigma'})$'s validity. In our settings, $\mathcal{A}_1'$ knows all valid DCSs that were correctly generated by $DCSSign$ oracle and recorded in $L_0^{DCSSign}$

and $L_1^{DCSSign}$. Thus, with a small probability that $(\overline{m}, \overline{\sigma'})$ is a *valid* message-DCS pair if, either$(\overline{m}, \overline{\sigma'})$ or its equivalent DCSs appears in $L_0^{DCSSign}$ or $L_1^{DCSSign}$, or $\mathcal{A}_{NTR}$ has the ability to "forge" such a DCS without the signer's secret key sk_S. And hence $\mathcal{A}_1'$ always ensures the validity of $(\overline{m}, \overline{\sigma'})$ before it simulates a proper confirmation or disavowal transcript.

Note if $\mathcal{A}_{NTR}$ outputs its guess in polynomial time t_{NTR}, $\mathcal{A}_{TS}$ outputs its guess in polynomial time t_{TS}, where $t_{NTR} = t_{TS}$.

Now we analyze $\mathcal{A}_{TS}$'s probability of breaking $\mathcal{DCS}$ on transcript-simulatability. Let event E_i denote executions follow in Case i and the simulation in Case i succeeds. If $\mathcal{A}_{TS}$ wins the game with an advantage adv_{TS}, we have:

$$adv_{TS} = \Pr[b_{TS} = b] - \frac{1}{2} = \Pr[b_{TS} = b \mid E_1 \vee E_2] \times \Pr[E_1 \vee E_2] - \frac{1}{2}$$

Analogously, in Case 2, the simulation succeeds with a probability of $1 - adv_{UF}$ where adv_{UF}^* denotes $\mathcal{A}_{NTR}$'s advantage to "forge" a valid message-DCS pair in the above procedure. Note A_{NTR} is not allowed to possess sk_C which differs from a typical adversary in the unforgeability game (see in Def 2.2.4), and thus adv_{UF}^* will be less equal than the advantage of such a typical forger. So the simulation in Case 2 succeeds with a overwhelming probability assuming adv_{UF}^* is negligible. Thus the probability of E_2 that indicates the procedure falls in Case 2 and the simulation succeeds is $\Pr[E_2] = \frac{1}{2} \times (1 - adv_{UF}^*) \approx \frac{1}{2} \times (1 - adv_{UF})$, and we have $\Pr[E_1 \vee E_2] = \Pr[E_1] + \Pr[E_2] = 1 - \frac{1}{2} \times adv_{UF}$.

In both cases, $\mathcal{A}_{TS}$ always succeeds in guessing b if $\mathcal{A}_{NTR}$ can make a successful guess on b', so we have $\Pr[b_{TS} = b \mid E_1 \vee E_2] = \frac{1}{2} + adv_{NTR}$.

Therefore, we have:

$$adv_{TS} \approx (\frac{1}{2} + adv_{NTR}) \times (1 - \frac{1}{2} \times adv_{UF}) - \frac{1}{2} = adv_{NTR} - \frac{1}{4} \times adv_{UF} - \frac{1}{2} \times adv_{NTR} \times adv_{UF}$$

One could derive the following inequation by eliminating adv_{UF}:

$\neg TS$	$\neg INV$	$\neg NTR$	NTR	$\neg TS \rightarrow \neg INV \vee \neg NTR$	$\neg TS \wedge NTR \rightarrow \neg INV$
T	T	T	F	T	T
T	T	F	T	T	T
T	F	T	F	T	T
T	F	F	T	F	F
F	T	T	F	T	T
F	T	F	T	T	T
F	F	T	F	T	T
F	F	F	T	T	T

Table 4.1: A Truth Table

$$adv_{TS} \approx adv_{NTR}$$

and hence the result follows. $\square$

4.2.3 A Proof for the Other Side

On the other side, what we really care about is, *"if a DCS scheme satisfies both invisibility and non-transferability, transcript-simulatability is implicitly satisfied*". And the contrapositive of the above statement is: *"if a DCS scheme does not satisfy transcript-simulatability, either non-transferability or invisibility does not hold*". Employing the symbolic logic, using the symbol "$\neg P$" to denote the negation of the proposition P, one could derive the following result.

- Step1: Our goal is to prove: "$\neg TS \rightarrow \neg INV \vee \neg NTR$".
- Step 2: To prove: "$\neg TS \rightarrow \neg INV \vee \neg NTR$", is equivalent to prove such a statement: "$(\neg TS \rightarrow \neg INV) \vee (\neg TS \rightarrow \neg NTR)$".
- Step 3: Alternatively, to prove: "$\neg TS \rightarrow \neg INV \vee \neg NTR$", is equivalent to prove such a statement: "$\neg TS \wedge NTR \rightarrow \neg INV$".

In fact, we find neither the left clause nor the right clause in Step 2 can be straightforwardly proved in such an OR-statement. The reason is that, the definition of transcript-simulatability could be treated as a combination of invisibility and non-transferability, since $\mathcal{A}_2$ in $Game_{TS}$ can win the game if it can either make a distinction on (m_0, m_1, σ') which in fact looks like an invisibility challenge, or it can make a distinction on the generator of the evidence which turns into a non-transferability challenge of NTR-1 in that case. So either a successful invisibility attacker or a successful attacker for non-transferability of NTR-1 can conduct to a successful transcript-simulatability attacker. However, we cannot say the reverse holds, since one cannot explicitly figure out that these two underlying challenges in a transcript-simulatability game are independent. However, if one raises a reasonable assumption, one may get a weaker but still very useful theoretical result. Indeed, what we achieved is by showing the truth of such an statement in Step 3: "*Under the assumption that non-transferability is satisfied, a successful invisibility adversary exists if a successful transcript-simulatability adversary exists*". And such a result is formally presented in the following theorem. Note the correctness of the statement in step 3 is guaranteed by applying the truth table in Table 4.1.

One should note that non-transferability in the theorem below is specified as NTR-0, since we are only able to construct a successful invisibility attacker linking a NTR-0 secure DCS scheme. And it remains an open problem if one replace NTR-0 with NTR-1 in theorem 4.3., that is, the problem that how to prove that transcript-simulatability implies invisibility plus non-transferability in Def 3.7., Chapter 3 (or the definition of [18]).

Theorem 4.3. If the scheme $\mathcal{DCS}$ satisfies non-transferability (NTR-0) as specified in Def 3.7., and if a PPT adversary $\mathcal{A}_{TS}$ breaks the scheme $\mathcal{DCS}$ on transcript-simulatability by executing $Game_{TS}$ in polynomial time t_{TS} with a non-negligible advantage adv_{TS}, there exists a PPT adversary $\mathcal{A}_{INV}$ that can break the scheme $\mathcal{DCS}$ on invisibility by executing $Game_{INV}$ in polynomial time t_{INV} with a non-

negligible advantage adv_{INV}.

Proof: The main idea of the proof is to construct an invisibility adversary $\mathcal{A}_{INV}$ by using a transcript-simulatability adversary $\mathcal{A}_{TS}$ as a subroutine, where the former simulates the environment for $\mathcal{A}_{TS} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$ and $\mathcal{A}'_1$, which is the concomitant simulator of $\mathcal{A}_1$ in the transcript-simulatability game. During the simulation, we let the algorithm $\mathcal{A}_1$ violates with some constrains in a little way (The details of $\mathcal{A}_1$'s innocuous behaviors will be given shortly).

For the assumption that the scheme $\mathcal{DCS}$ meets NTR-0, we require that, for every non-transferability adversary say $\mathcal{A}_{NTR-0}$, there exists a simulator say $\mathcal{S}_{NTR-0}$ that is able to simulate an interaction with $\mathcal{A}_{NTR-0}$, and such a simulated interaction is indistinguishable from a interaction performed by the real confirmer. Let adv_{NTR-0} denote the advantage of $\mathcal{A}_{NTR-0}$ to win the non-transferability game in the definition of NTR-0, and we have $adv_{NTR-0} \leq negl(\lambda)$.

Initially, the challenger of $\mathcal{A}_{INV}$ runs the key generation algorithm to generate the public parameters and the key pairs of the signer and the confirmer. Th adversary $\mathcal{A}_{INV}$ receives the public parameters including pk_S, pk_C from its challenger, and relays a copy to $\mathcal{A}_0$. In addition, $\mathcal{A}_{INV}$ will hold the signer's secret key sk_S as required.

$\mathcal{A}_{INV}$ simulates $\mathcal{A}_0$'s oracle $\mathcal{O}$ including $Confirm$, $Disavow$ and $Extract$ queries by relaying the requests and responses between $\mathcal{A}_{INV}$'s challenger and $\mathcal{A}_0$. In particular, $\mathcal{A}_{INV}$ can simulate $\mathcal{A}_0$'s $DCSSign$ oracle using the signer's secret key sk_S by himself. At some point, $\mathcal{A}_0$ outputs two messages, say m_0 and m_1.

$\mathcal{A}_{INV}$ uses m_0 and m_1 as its output at its challenge phase. After flipping a fair coin b_{INV}, $\mathcal{A}_{INV}$'s challenger generates a DCS σ' on message m_b. Next $\mathcal{A}_{INV}$ flips a hidden coin b_{TS}. If $b_{TS} = 0$, $\mathcal{A}_{INV}$ sends (m_0, m_1, σ') to $\mathcal{A}_1$ as input; otherwise, $\mathcal{A}_{INV}$ sends (m_0, m_1, σ') to $\mathcal{A}'_1$ as input. Now $\mathcal{A}_{INV}$ simulates $\mathcal{A}_1$ or $\mathcal{A}'_1$'s oracle services by using the following strategy:

[*Simulation for* $\mathcal{A}_1'$] Because $\mathcal{A}_1'$ is only allowed to access to $DCSSign$ oracle. By using sk_S, $\mathcal{A}_{INV}$ can simply answer all $\mathcal{A}_1'$'s DCSSign queries.

[*Simulation for* $\mathcal{A}_1$] For all queries NOT involving σ', $\mathcal{A}_{INV}$ can simply relay those queries to its own challenger. However, for any query including σ', it cannot repeat that according to the limitation of such an invisibility adversary after receiving the challenge. Without loss of generality, we assume $\mathcal{A}_1$ is always requesting the interactions about a confirmation on (m_0, σ') and a disavowal on (m_1, σ'). Since NTR-0 is satisfied due to the assumption, to confirm (m_0, σ') or disavow (m_1, σ'), we alternatively require $\mathcal{A}_{INV}$ invokes $\mathcal{S}_{NTR-0}$ while treating (m_0, σ') as a valid message-DCS pair and (m_1, σ') as an invalid one. Note this instant guess is consistent with $\mathcal{A}_{INV}$'s previous choice, i.e., if $b_{TS} = 0$, $\mathcal{A}_{INV}$ sends (m_0, m_1, σ') to $\mathcal{A}_1$ as input.

[$\mathcal{A}_1$'s innocuous behavior] We require $\mathcal{A}_1$ always performs the following innocuous behavior during the simulation: For each confirmation on (m_0, σ') and each disavowal on (m_1, σ'), $\mathcal{A}_1$ tries to identify if it interacts with a real confirmer or a simulator. Furthermore, if it "thinks" it interacts with a simulator, it rejects that verification and terminates; otherwise, it accepts that verification.

At the end of the simulation, $\mathcal{A}_1$ or $\mathcal{A}_1'$ outputs a tuple $(m_0, m_1, \sigma', s, \tau)$ and sends it to $\mathcal{A}_2$, where s indicates the state information and τ denotes a transcript shows that σ' is signed on m_0. Now $\mathcal{A}_{INV}$ still relays all $\mathcal{A}_2$'s oracle queries that will not include (m_0, σ') or (m_1, σ') or their equivalent DCSs.

If $\mathcal{A}_2$ outputs a bit b, $\mathcal{A}_{INV}$ outputs b as its guess for its game.

Now we analyze $\mathcal{A}_{INV}$'s probability of breaking $\mathcal{DCS}$ on invisibility by using $(\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_1', \mathcal{A}_2)$ as subroutines. Note $\mathcal{A}_{INV}$ successfully simulates the environment for $(\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_1', \mathcal{A}_2)$ except some failure events happen. We define the following $fail$ events during the $\mathcal{A}_{INV}$'s simulation.

- $fail_1$: $b_{TS} \neq b_{INV}$. This indicates $\mathcal{A}_{INV}$ falsely sent (m_0, m_1, σ') to $\mathcal{A}_1$ or $\mathcal{A}'_1$.
- $fail_2$: $\mathcal{A}_1$ rejects the confirmation transcript on (m_0, σ') or a disavowal transcript on (m_1, σ'), after its interaction with $\mathcal{A}_{INV}$. This suggests $\mathcal{A}_1$ is potentially able to play as an non-transferability adversary, and distinguish the interaction between a real confirmer and a simulator who knows the signature's validity.

Obviously, we have $Pr[fail_1] = \frac{1}{2}$ since b_{TS} and b_{INV} are two independent fair coins. For the second type of failures, we think that $\mathcal{A}_1$ will definitely request a verification on (m_0, σ') or (m_1, σ'), to output a piece of transcript which shall convince $\mathcal{A}_2$ that σ' is signed on m_0. If $\mathcal{A}_1$ is aware that during interactions, at least one interaction is controlled by the simulator, it would be potentially able to play as a non-transferability adversary, and distinguish such an interaction, i.e., to verify (m_0, σ') or (m_1, σ'), between a real confirmer and a simulator. Recall that adv_{NTR-0} denotes the advantage of a PPT adversary to win the non-transferability game in the definition of NTR-0. Because $\mathcal{A}_1$ tries to *subconsciously* identify the transcript from $\mathcal{A}_{INV}$ during each interaction, $\mathcal{A}_1$'s probability to accept such a simulated transcript is $\epsilon = \frac{1}{2} - adv_{NTR-0}$. And thus we have $\Pr[fail_2] = 1 - \epsilon^2$.

From the simulation specified above, if $fail_1$ happens, $\mathcal{A}_{INV}$'s probability to guess b_{INV} correctly is exactly $\frac{1}{2}$, and we refer to this event as "$\mathcal{A}_{INV}$ wins". Otherwise, $\mathcal{A}_{INV}$'s probability to guess b_{INV}correctly is equal to $\mathcal{A}_{TS}$'s probability to guess b_{TS} correctly, and we refer to the later event as "$\mathcal{A}_{TS}$ wins".

we have:

$$\Pr[\mathcal{A}_{INV}\ wins] = \frac{1}{2} \times \Pr[fail_1] + \Pr[\mathcal{A}_{TS}\ wins] \times \Pr[\neg fail_1]$$

$$\begin{aligned}\frac{1}{2} + adv_{INV} = \frac{1}{2} \times \frac{1}{2} &+ (\Pr[\mathcal{A}_{TS}\ wins \mid \neg fail_2] \times \Pr[\neg fail_2] \\ &+ \Pr[\mathcal{A}_{TS}\ wins \mid fail_2] \times \Pr[fail_2]) \times \frac{1}{2}\end{aligned}$$

Note $Pr[\mathcal{A}_{TS}\ wins \mid fail_2] = \frac{1}{2}$ means $\mathcal{A}_{TS}$'s probability to win is exactly $\frac{1}{2}$ if $fail_2$ happens. Thus we have:

$$\frac{1}{2}+adv_{INV} = \frac{1}{2}\times\frac{1}{2}+((\frac{1}{2}+adv_{TS})\times\epsilon^2+\frac{1}{2}\times(1-\epsilon^2))\times\frac{1}{2} = \frac{1}{2}+\frac{1}{2}\times adv_{TS}\times(\frac{1}{2}-adv_{NTR-0})^2$$

that is, $adv_{INV} = \frac{1}{2} \times adv_{TS} \times (\frac{1}{2} - adv_{NTR-0})^2$

According to the assumption that adv_{NTR-0} is a negligible probability, One could derive the following inequation by eliminating adv_{NTR-0}:

$$adv_{INV} \approx \frac{1}{8} \times adv_{TS}$$

Hence if adv_{TS} is non-negligible, adv_{INV} is non-negligible.

Considering the running time of $\mathcal{A}_{INV}$: Because t_{NTR} and t_{TS} is polynomial, the total running time of $\mathcal{A}_{INV}$ is t_{TS} plus $\mathcal{A}_1$'s additional executing time which is $2 \times t_{NTR-0}$, where t_{NTR-0} denotes the polynomial time of $\mathcal{A}_{NTR-0}$ executing the game of NTR-0, and thus $\mathcal{A}_{INV}$ still succeeds in polynomial time. □

Since almost all the current practical or generic DCS schemes are constructed by using (concurrent) ZK proofs or ZK proofs of knowledge as the underlying building blocks. Furthermore, non-transferability follows in a straightforward manner from zero-knowledge property of the proofs in the $Confirm$ or $Disavow$ protocol (Similar argument can be found in the proof of Theorem 1 in [18]). One could get the following more applicable corollary.

Corollary 4.1. Let $\mathcal{DCS}$ be a designated confirmer signature scheme with the underlying verification protocols which are based on zero knowledge proofs. If $\mathcal{DCS}$ satisfies invisibility, it satisfies transcript-simulatability.

(Proof omitted). One could applying the above proof of Theorem 4.3, by replacing the assumption of NTR-0 into a stronger one such that the zero knowledge property of the underlying $Confirm$ and $Disavow$ protocols is guaranteed. And the related

INV $\longrightarrow$ UNIMP	*(by Theorem 4.1)*
TS $\longrightarrow$ INV	*(by Lemma 4.1)*
TS $\longrightarrow$ NTR-1	*(by Lemma 4.2)*
TS $\longrightarrow$ INV $+$ NTR-1	*(by Theorem 4.2)*
INV $+$ NTR-0 $\longrightarrow$ TS	*(by Theorem 4.3)*
INV $+$ ZK $\longrightarrow$ TS	*(by Corollary 4.1)*

Figure 4.6: Relations among security notions in DCS Schemes

equation still holds: $adv_{INV} \approx \frac{1}{8} \times adv_{TS}$. $\square$

4.3 Summary

We show that at least CM model is stronger than GMR model, namely the combination of invisibility and non-transferability (NTR-0) implies transcript-simulatability. However, the converse only holds when assuming a new definition of non-transferability, i.e., NTR-1 is satisfied. A more practical result is, if the underlying $Confirm$ and $Disavow$ protocols are based on zero knowledge proofs, transcript-simulatability is equivalent to invisibility. We show a more detailed result listed in Figure 4.6, where the strings of "INV", "UNIMP", "TS", represent the security notions of invisibility, unimpersonation and transcript-simulatability, respectively.

For the future work, it would be very interesting to find a formal proof that figures out the relations between the notion of NTR-0 and the notion of NTR-1. Although due to the Theorem 4.2 and Theorem 4.3, NTR-1 seems intuitively no stronger than NTR-0.

Chapter 5

A Paring-based DCS Scheme with Unified Verification

5.1 Introduction

There is one limit in most of the existing DCS schemes: A signer is *not* given the ability to disavow invalid DCS signatures. Therefore, the current concept of DCS has not yet fully extended that of undeniable signatures, as the latter does grant the signer the ability of disavowal. Moreover, in many applications it seems more sensible to enable the signer having the same ability as the confirmer to confirm any valid DCS and deny any invalid DCS. In fact, this additional ability will not only alleviate the burden of the confirmer, but effectively prevents the signer from viciously claiming: "*This alleged DCS is not valid, but I am not able to show this*". Galbraith and Mao [32] first pointed out DCS schemes should allow a signer to be able to deny invalid signatures but they did not present any construction with this property. Motivated by this observation, in this thesis we propose the concept of DCS with *unified verification*, together with a formal security model and a concrete construction. Simply speaking, in DCS scheme with unified verification both the signer and the designated confirmer can run the same protocol to confirm valid signatures, and another same protocol to disavow invalid signatures. Based on the security models in [18, 32], we first present a new security model for DCS with unified verification to capture all desirable security requirements (Section 3). We also point out that the proposed model can be easily generalised to accommodate *DCS with full verification*, in which the signer and the confirmer do not necessarily run the same protocols to confirm or disavow signatures.

Then, we consider how to construct a DCS with unified verification. This is a challenge, as simply revising the existing constructions does not work. The reason is that almost all previous DCS schemes [45, 38, 58, 64, 78] follow the approach of encrypting the signer's signature under the confirmer's public key. So, without the confirmer's private key the signer is not able to show that a CCA2 ciphertext is not a proper encryption of his/her signature for a given message. In fact, it seems that even Wang et al.'s DCS [78] without using public encryption cannot be converted into a scheme supporting the signer's disavowal, since an alleged DCS may contain a non Diffie-Hellman tuple, for which the signer does not know a witness to prove this fact at all.

However, this does not mean that it is impossible to construct DCS schemes with unified verification or with full verification. Due to the amazing property of bilinear pairings, we constructed the first concrete DCS scheme which supports *unified verification* in the preliminary version of this work [81], though that scheme only achieves weak invisibility. By making a simple enhancement to our previous work [81], we get a new and secure DCS scheme with unified verification in this thesis (Section 4) and prove its security in the random oracle model (Section 5). Specifically, the new scheme is constructed by encrypting the BLS pairing based short signature [14] under the *signed* ElGamal encryption [72]. Note that directly exploiting plain ElGamal encryption [30] cannot guarantee the *invisibility*, due to ElGamal's malleability (See more discussion in Section 4). Moreover, compared to the existing DCS schemes [45, 38, 78, 48], the proposed solution has a conceptual simpler structure and a short signature size, though the computational overhead is a little higher due to pairing evaluation. Another interesting observation about our construction is that the underlying signed ElGamal encryption is actually not CPA secure due to the fact that the DDH (Decisional Diffie-Hellman) problem is easy in pairing setting, though signed ElGamal encryption is proved to be CCA2-secure in the random oracle model and in the generic group model by Schnorr and Jakobsson [72]. This is seemingly contradictory to the result by Okamoto [64]. The likely reason for this is that our definitions on the security of DCS are different from Okamoto's, but we are not very sure at this moment. So, here we would like to promote this issue as an open problem.

5.2 Bilinear Pairings and the BLS Signature

Basically, a pairing is a function that takes two points on an elliptic curve as input, and outputs an element of some multiplicative group. Weil pairing and Tate pairing are two known symmetric pairings, while some other pairings, e.g., Eta pairing and Ate Pairing [34], have been given more and more attentions.

Definition 5.1. Suppose that G and G_t be two multiplicative cyclic groups of prime order q, while g is a generators of G. A *bilinear pairing* on (G, G_t) is a map $e : G \times G \to G_t$, which satisfies the following properties:

Bilinearity: *For all* $u, v \in G$, *and for all* $a, b \in \mathbb{Z}_q$, $e(u^a, v^b) = e(u, v)^{ab}$.

Non-degeneracy: $e(g, g) \neq 1$, *where* 1 *is the multiplicative identity of group* G_t.

Computability: e *can be efficiently computed.*

We now review the BLS short signature scheme of Boneh, Lynn and Shacham

[14]. In general, the scheme has three algorithms, *Key Generation*, *Sign*, and *Verify*. In addition, it needs a full-domain hash function H: $\{0,1\}^* \to G$.

Key Generation: A user randomly selects $x \in \mathbb{Z}_q^*$ as its private key, and computes $y = g^x$ as the corresponding public key.

Sign: To sign a message $m \in \{0,1\}^*$, the signer with private key x computes $h = H(m) \in G$, and the signature $\sigma = h^x \in G$.

Verify: Given a public key y and a message-signature pair (m, σ), a verifier first computes $h = H(m)$, and then checks if $e(g, \sigma) = e(y, h)$ holds or not. If it holds, it accepts the validity of (m, σ).

Note that the BLS signature is only one single element of G, so it is very short (for example, 171 bits) for some elliptic curves. In [14], it is proved that the security of the BLS signature scheme follows the hardness of Computational Diffie-Hellman (CDH) problem in the random oracle model. In fact, the original BLS signature [14] is described in the setting of *asymmetric pairing* e, i.e., e is a bilinear map from $G_1 \times G_2$ to G_t, while G_1, G_2, and G_t are all multiplicative cyclic groups of prime order q. Here, for simplicity we just use *symmetric pairing* by letting $G = G_1 = G_2$. We note that to extend our DCS scheme for the asymmetric pairing is straightforward.

5.3 Concurrent Zero-knowledge From Honest-Verifier Zero-Knowledge

Zero-knowledge (ZK) protocols allow a prover to validate theorems to a verifier without giving away any other knowledge other than the theorems being true (i.e., existing witnesses) [43]. Traditional notion of ZK considers the security in a stand-alone (or sequential) execution of the protocol. Motivated by the use of such protocols in an asynchronous network like the Internet where many protocols are run concurrently at the same time, studying security properties of ZK protocols in such concurrent settings has attracted extensive research efforts in recent years [29]. Informally, a ZK protocol is called concurrent zero-knowledge (CZK) if the ZK related simulatability property holds in the concurrent settings, namely, when a malicious verifier concurrently interacts with a polynomial number of honest prover instances and schedules message exchanges as it wishes. We note, in DCS schemes, we require CZK protocols, because an adversary in DCS schemes may act as arbitrary cheating verifiers during the concurrent execution of protocols that confirm or deny all alleged DCS signatures.[1]

In this work, for presentation simplicity, we describe the Confirm and Disavow protocols with Σ-protocols (i.e., 3-round public-coin special honest verifier interactive zero-knowledge (SHVIZK) with special soundness) directly.

Definition 5.2.

A 3-round public-coin protocol $\langle P, V\rangle$ is said to be a Σ-protocol for a relation R if the following hold:

- Completeness. If P, V follow the protocol, the verifier always accepts.

- Special soundness. From any common input x of length n and any pair of accepting conversations on input x, (a, e, z) and (a, e', z') where $e \neq e'$, one can efficiently computes w such that $(x, w) \in R$. Here a, e, z stand for the first, the second and the third message respectively and e is assumed to be a string of length t (that is polynomially related to n) selected uniformly at random in $\{0, 1\}^t$.

[1]We note that the CZK issue was not realised in [48], where only stand-alone 4-round ZK is mentioned.

- Special honest verifier zero-knowledge (SHVZK). There exists a probabilistic polynomial-time (PPT) simulator S, which on input x and a random challenge string e, outputs an accepting conversation of the form (a, e, z), with the same probability distribution as the real conversation between the honest P, V on input x.

Σ-protocols have been proved to be a very powerful cryptographic tool and are widely used. Transformation methodologies from Σ-protocols to CZK protocols, in the common reference string (CRS) model, are known (e.g., [27, 36]), but usually incurs much additional computational and communication complexity. Moreover, for CZK transformation in the CRS model, *the CRS should be included as a part in the public-key of the confirmer*, which additionally increases the public-key length of the confirmer. The transformation methodology proposed in [27] is recalled in section 5.3.1, which is among the most efficient transformations.

As we aim for DCS in the RO model, in this work we develop a highly efficient transformation from Σ-protocols to *straight-line* CZK in the *unprogrammable* RO model, where straight-line CZK means that the CZK simulator works in a straight-line way (*without rewinding the underlying adversary*). Given access to a random oracle $\mathcal{O}$, we can transform a Σ-protocol into a non-interactive zero-knowledge (NIZK) protocol via the Fiat-Shamir heuristics. But, the NIZK got this way loses deniability [63, 66], which is however required for DCS schemes. The deniability loss is due to the programmability of RO in the security analysis [63, 66]. To overcome the deniability loss of simulation with programmable RO, the works of [63, 66] proposed the unprogrammable RO model, and showed that ZK with unprogrammable RO reserves the deniability property. We briefly discuss the main difference between programmable RO model and unprogrammable RO model. In the programmable RO model, the simulator can (1) see the queries parties make to the random oracle and (2) can choose the

Common input. An element $x \in L$ of length n, where L is an $\mathcal{NP}$-language that admits Σ-protocols.

P**'s** private input. A witness w for $x \in L$.

Random oracle. An *unprogrammable* random oracle denoted $\mathcal{O}$.

Round-1. The verifier V takes $e \in \{0,1\}^k$ uniformly at random, and sends $c = \mathcal{O}(e)$ to P.

Round-2. The prover P sends a (i.e., the first-round of the underlying Σ-protocol by running the underlying P_L) to V.

Round-3. V sends e to P.

Round-4. After receiving e from V, P first checks whether $c = \mathcal{O}(e)$. If not, P simply aborts; otherwise (i.e, $c = \mathcal{O}(e)$), P sends z (i.e., the last-round of the underlying Σ-protocol by running the underlying P_L) to V.

V**'s** decision. V checks, by running the underlying V_L, whether (a, c, z) is an accepting conversation of the underlying Σ-protocol for showing $x \in L$.

Table 5.1: Straight-line CZK protocol with unprogrammable RO

answers to these queries. The second is what we refer to as *programming* the random oracle. Suppose our goal is to simulate a transcript of the random oracle RO at some value s. Our intuition about the random oracle as a truly random function indicates that picking a truly random string should suffice, and indeed, even no computationally unbounded distinguishers, can distinguish a truly random string from RO's output of s, provided the distinguisher does not get access to RO. However, in the unprogrammable RO model, we give the distinguisher access to RO, then the only "good" simulation of the transcript is RO's output of s, and the simulation must query RO at s. In this setting, the simulator is *not* allowed to choose the answers to oracle queries. We remark that, in this work, unprogrammable RO is used only for achieving highly practical CZK, other parts of security analysis still rely on regular (programmable) random oracle.

Roughly speaking, before running the Σ-protocol (a, e, z), we require the verifier to first commit to its random challenge e by sending $c = H(e)$ on the top, where H is a hash function that is modeled as an unprogrammable RO in the analysis. The protocol is depicted in Figure-1. *Note that the additional computational complexity and communication complexity, incurred by this approach of transformation with unprogrammable RO, is minimal: only a hash value is incurred.*

5.3.1 CZK Transformation From Σ-Protocols

Zero-knowledge (ZK) protocols allow a prover to validate theorems to a verifier without giving away any other knowledge other than the theorems being true (i.e., existing witnesses). This notion was introduced by Goldwasser, Micali and Rackoff [43] and its generality was demonstrated by Goldreich, Micali and Wigderson [41]. Since its introduction ZK has found numerous and extremely useful applications, and by now has been playing the central role in modern cryptography.

Traditional notion of ZK considers the security in a stand-alone (or sequential) execution of the protocol. Motivated by the use of such protocols in an asynchronous network like the Internet where many protocols are run concurrently at the same time, studying security properties of ZK protocols in such concurrent settings has attracted extensive research efforts in recent years, initiated by Dwork, Naor and Sahai [29]. Informally, a ZK protocol is called concurrent zero-knowledge (CZK) if the ZK related simulatability property holds in the concurrent settings, namely, when a malicious verifier concurrently interacts with a polynomial number of honest prover instances and schedules message exchanges as it wishes.

We note, in DCS schemes, we require CZK protocols, because an adversary in DCS schemes may act as arbitrary cheating verifiers during the concurrent execution of protocols that confirm or deny all alleged DCS signatures. In this work, for presentation simplicity, we describe the Confirm and Disavow protocols with Σ-protocols (i.e., 3-round public-coin special honest verifier zero-knowledge with special soundness) directly. We then discuss transformation methodologies from Σ-protocols to CZK protocols in the common reference string (CRS) model or in the *unprogrammable* random oracle model.

5.3.1.1 Transformation from Σ-Protocol into CZK in the CRS Model

There are several general methodologies that transform Σ-protocols into CZK arguments in the CRS model (e.g., [27, 36]). To our knowledge, the approach proposed in [27] is the most efficient and has conceptual simple structure, which is suggested to use in this work.

For the transformation proposed in [27], the CRS consists of the public-key for a trapdoor commitment scheme. In this work, we use the DL-based trapdoor commitment, where the public-key is $h = g^x$, and the commitment to a value $v \in Z_q$ is $c = g^r h^v$, where r is randomly taken from Z_q and is served as the decommitment information. Note that for this concrete implementation, to commit to a value in Z_q, the committer needs to perform about 1.5 exponentiations, and the receiver needs to perform also about 1.5 exponentiations. The communication complexity, besides the transmission of the committed value v (that is sent in the decommitment stage), is about $2|q|$ (suppose the commitment c is of about $|q|$ bits).

To transform a Σ-protocol (a, e, z) into CZK, the key idea of [27] is to send $C(a)$, rather then the plain a, at the first-round of the transformed protocol, where C denotes the trapdoor commitment scheme; in the third-round, the prover opens the value a and computes the third-round message z.

Moreover, for the general transformation in the CRS model, *the CRS should be included as a part in the public-key of the confirmer*, which additionally increases the public-key length of the confirmer.

5.3.1.2 CZK from Σ-protocols with *unprogrammable* RO

Given access to a random oracle (RO) $\mathcal{O}$, we can transform a Σ-protocol into a non-interactive zero-knowledge (NIZK) protocol via the Fiat-Shamir heuristic. But, the NIZK got this way loses deniability [63, 66], which is however required for DCS schemes. The deniability loss is due to the programmability of RO in the security analysis [63, 66]. To overcome the deniability loss of simulation with programmable RO, the works of [63, 66] proposed the unprogrammable RO model where all parties have access to an unprogrammable (fixed) RO, where ZK with unprogrammable RO reserves the deniability property.

In this section, we give a general yet simple method of transforming Σ-protocols into *straight-line* CZK with *unprogrammable RO*, where straight-line CZK means that the CZK simulator works in a straight-line way (*without rewinding the underlying adversary*).

Given a Σ-protocol $\langle P_L, V_L\rangle(x)$ which consists of three rounds (a, e, z) for an $\mathcal{NP}$-language L, the transformed protocol, denoted $\langle P, V\rangle$ is presented in Figure-1.

Roughly speaking, before running the Σ-protocol (a, e, z), we require the verifier to first commit to its random challenge e by sending $c = h(e)$ on the top, where h is a hash function that is modeled as RO in the analysis.

Note that the additional computational complexity and communication complexity, incurred by this approach of transformation with unprogrammable RO, is minimal: only a hash value is incurred.

Theorem 5.1. The protocol depicted in Figure-1 is a *straight-line* CZK *proof* with *unprogrammable* RO for any language admitting Σ-protocol.

Proof. The completeness of the protocol $\langle P, V\rangle$ can be directly checked.

Perfect soundness. The perfect soundness of $\langle P, V\rangle$ is from the observations: the commitment c perfectly hides e in the RO model; Then, the perfect soundness of

$\langle P, V\rangle$ is inherited from the special soundness of the underlying Σ-protocol $\langle P_L, V_L\rangle$. That is, the transformed protocol $\langle P, V\rangle$ is a *proof* rather than an *argument* (i.e., computationally sound protocol).

***Straight-line* CZK with *unprogrammable* RO.** For any concurrent malicious verifier V^*, the simulator S runs V^* as a subroutine and works as follows, with oracle access to a *unprogrammable* RO $\mathcal{O}$:

- For any oracle query made by V^* on input e, S makes the same query to the unprogrammable RO $\mathcal{O}$. S returns back the answer, denoted c, from $\mathcal{O}$ to V^*, and records (c, r) into a list $\mathcal{L}_{\mathcal{O}}$.
- Whenever V^* starts a new concurrent session, on a common input $x \in L$, by sending c (as the first-round message) to S, S works as follows:
  - S firstly checks whether $c \in \mathcal{L}_{\mathcal{O}}$. If not, S simply aborts the simulation, and outputs “failure". This failure is called “*Case-1 failure*" for presentation simplicity.
  - If $c \in \mathcal{L}_{\mathcal{O}}$, S retrieves the record (c, e) in $\mathcal{L}_{\mathcal{O}}$ and works as follows: S runs the underlying SHVZK simulator S_L (guaranteed for the underlying Σ-protocol $\langle P_L, V_L\rangle$) on the input (x, e), denoted $S_L(x, e)$, to get a simulate transcript (a, e, z) of the underlying Σ-protocol $\langle P_L, V_L\rangle$. Then S sends a to V^* as the second-round message of the current session. If V^* returns back e to S in the third-round, S returns back z in the fourth-round and successfully completes the simulation of the current session; If V^* returns back $e' \neq e$ in the third-round, S simply aborts the simulation, and outputs “failure". This failure is called “*Case-2 failure*" for presentation simplicity.

It is easy to check that S outputs “failure" (either Case-1 failure or Case-2 failure) with negligible probability in the RO model. Specifically, for Case-1 failure, with overwhelming probability V^* cannot guess the correct value c without querying the RO $\mathcal{O}$ with e; For Case-2 failure, with overwhelming probability V^* cannot get two different values e, e' such that $c = \mathcal{O}(e) = \mathcal{O}(e')$.

Conditioned on S does not output "failure", the simulation of S is identical to the real view of V^*, which establishes the CZK property. Furthermore, S works in the unprogrammable RO model, as S never programs the RO $\mathcal{O}$ by itself. Specifically, S only accesses the unprogrammable RO $\mathcal{O}$ to see the queries made by the underlying V^*. Moreover, the simulation of S with restricted RO is *straight-line*, as S never rewinds the underlying V^*. $\square$

5.4 Security Model

We update the DCS model following the security model in section 2, Chapter 3. In addition, we shall briefly mention how this new model can be modified to accommodate DCS with full verification.

Definition 5.3. (Syntax). *A* correct designated confirmer signature scheme with unified verification *involves three roles of parties, i.e., a signer S, a designated confirmer C, and a verifier V, and consists of the following components:*

Key Generation (G_s, G_c): *Given the security parameter λ, denoted by 1^λ, as input, probabilistic polynomial time (PPT) algorithm G_s outputs a pair of strings (sk_S, pk_S) as the signer's private key and public key, respectively. Similarly, PPT algorithm G_c that takes on input 1^λ, outputs a pair of strings (sk_C, pk_C) as the designated confirmer's private key and public key, respectively.*

Sign: *Given a message m and a signer's private key sk_S, algorithm Sign produces a (standard) signature σ for message m. Namely, $\sigma = Sign(m, sk_S)$.*

Verify: *Given a public key pk_S, a message m, and a signature σ, algorithm Verify* *outputs Accept or Reject. For any key pair* (sk_S, pk_S)*, any message* m*, we have*

$$Verify(m, Sign(m, sk_S), pk_S) = \text{Accept}.$$

DCSSign: *Given a message* m*, a signer's private key* sk_S *and the confirmer's public key* pk_C*, algorithm DCSSign outputs* σ' *as a designated confirmer signature on message* m*. Namely,* $\sigma' = DCSSign(m, sk_S, pk_C)$*.*

Extract: *Given* $(m, \sigma', sk_C, pk_C, pk_S)$ *as input, algorithm Extract outputs a string* σ *such that Verify(*m*,* σ*,* pk_S*) = Accept or* $\perp$*. In the case Extract can successfully extract a valid standard signature* σ *from* σ'*, we say that* σ' *is* ***extractable*** *w.r.t. message* m*. Otherwise,* σ' *is* ***unextractable****.*

Confirm: *As an interactive protocol, either the signer* S *with private input* sk_S *or the designated confirmer* C *with private input* sk_C *can run Confirm protocol with a verifier* V *to confirm that an alleged DCS* σ' *for a message* m *is extractable. The common input for the protocol is (*m*,* σ'*,* pk_S*,* pk_C*). After the protocol is run, the verifier outputs* $b \in$ *{Accept,* $\perp$*}. We say* σ' *is* ***valid*** *w.r.t. message* m*, if the verifier's output is Accept. Otherwise, the validity of* σ' *is undetermined. The Confirm protocol should be complete and sound.*

a) **`Completeness`**: *For all honest* C*,* S*, and* V*, if Verify(*m*, Extract(*m*,* σ'*,* sk_C*,* pk_C*,* pk_S*),* pk_S*) =Accept, then* $Confirm_{(C,V)}$*(*m*,* σ'*,* pk_S*,*pk_C*) =Accept, and* $Confirm_{(S,V)}$*(*m*,* σ'*,* pk_S*,* pk_C*)=Accept.*

b) **`Soundness`**: *For any potentially cheating confirmer* C'*, any potentially cheating signer* S'*, and any honest verifier V, if Verify(*m*, Extract(*$m, \sigma', sk_C, pk_C, pk_S$*),* pk_S*)* $= \perp$*, then*

$$\Pr[Confirm_{(C',V)}(m, \sigma', pk_S, pk_C) = \mathit{Accept}] < negl(\lambda), \text{ and}$$
$$\Pr[Confirm_{(S',V)}(m, \sigma', pk_S, pk_C) = \mathit{Accept}] < negl(\lambda).$$

The probability is taken over all possible coins tossed by C', S', V, G_s, G_c, and Extract. This means, neither a cheating confirmer C' nor a cheating signer S' can convince an honest verifier V that an un-extractable designated confirmer signature σ' is valid. In other words, all valid DCS signatures are extractable.

Disavow: *As an interactive protocol, either the signer S with private input sk_S or the designated confirmer C with private input sk_C can run Disavow protocol with a verifier V to convince that an alleged DCS σ' is unextractable. The common input to the protocol is (m, σ', pk_S, pk_C), while the verifier output is $b \in$ {Accept, $\perp$}. If the verifier's output is Accept, we say σ' is* invalid *w.r.t. message m. Otherwise, the invalidity of σ' is undetermined. The Disavow protocol should be complete and sound.*

a) **`Completeness:`** *For all honest C, S, and V, if Verify(m, Extract(m, σ', sk_C, pk_C, pk_S), pk_S) $= \perp$, then $Disavow_{(C,V)}$(m, σ', pk_S, pk_C) =Accept, and $Disavow_{(S,V)}$(m, σ', pk_S, pk_C) =Accept.*

b) **`Soundness:`** *For any potentially cheating confirmer C', any potentially cheating signer S', and any honest verifier V, if Verify(m, Extract(m, σ', sk_C, pk_C, pk_S), pk_S) =Accept, then*

$$\Pr\left[Disavow_{(C',V)}(m, \sigma', pk_S, pk_C) = Accept\right] < negl(\lambda), \text{ and}$$
$$\Pr\left[Disavow_{(S',V)}(m, \sigma', pk_S, pk_C) = Accept\right] < negl(\lambda).$$

The probability is taken over all possible coins tossed by C', S', V, G_s, G_c, and Extract. This means, neither a cheating confirmer C' nor a cheating signer S' can convince an honest verifier V that an extractable designated confirmer signature σ' is invalid. In other words, all invalid DCS must be unextractable.

Remark 1. In contrast to the models given in [18, 45, 38, 78], there are three main differences in the above syntax definition. Firstly, we include the basic signature generation and verification algorithms to make the syntax more complete. Secondly, an algorithm *DCSSign* is now used to produce a DCS instead of an interactive protocol *ConfirmedSign* in [45, 38, 78] to allow the signer generating a valid DCS and confirming it when it is just generated. The reason for this is that the signer will use the same *Confirm* protocol to show the validity of a DCS as does by the confirmer. Finally, in our model the signer is also able to use the *Disavow* protocol to show the invalidity of an alleged DCS. This is definitely necessary, as our DCS model targets to support unified verification.

Remark 2. Due to the above changes in syntax, we accordingly update the security definitions by including all necessary oracle accesses. Security for the signer or unforgeability requires that no adaptive PPT adversary can forge a valid DCS on a fresh message on behalf of a specific signer, even it compromises the secret keys of the confirmer and other signers. This means that unforgeability should be satisfied in multi-signer settings, i.e., in the scenario of multiple signers sharing the same confirmer. In our definition of unforgeability given below, the forging algorithm is not given oracle accesses for which the confirmer is the prover, since it already holds the confirmer's private key sk_C. Due to a similar reason, the *Sign* oracle for underlying signatures is not provided as the attacker can simulate this oracle by asking *DCSSign* queries and then running *Extract* to get basic signatures for any messages.

Definition 5.4. Security for the signer (Unforgeability): *Let $\mathcal{F}$ be a PPT forging algorithm, which on input 1^n, pk_S, pk_C and sk_C, can request oracle access in $\mathcal{O}_{\mathcal{F}}$={$DCSSign$, $Confirm_{(S,\mathcal{F})}$, $Disavow_{(S,\mathcal{F})}$} for polynomially many times for adaptively chosen inputs of its choice; and then outputs a DCS message-signature pair (m, σ') in which message m is not previously asked in DCSSign queries. We say a DCS scheme is* secure for the signer or existentially unforgeable, *if for any PPT forging algorithm $\mathcal{F}$,*

$$\Pr[Verify(m, Extract(m, \sigma', sk_C, pk_C, pk_S), pk_S) = Accept] < negl(\lambda).$$

The probability is taken over all possible coins used by $\mathcal{F}$, S, and key generation algorithm G_s, G_c.

Intuitively, security for the confirmer or invisibility means that no adaptive PPT adversary $\mathcal{D}$ can distinguish between a valid DCS and an invalid DCS for a given message (or two designated confirmer signatures).

Definition 5.5. Security for the confirmer (Invisibility): *Firstly, Key Generation algorithms are run for the signer and the confirmer on input 1^λ. $\mathcal{D}$ is given pk_S and pk_C, which are the public keys of the signer and the confirmer. As a training purpose, $\mathcal{D}$ is allowed to create signature-key pairs $(sk_\mathcal{D}, pk_\mathcal{D})$ (not necessarily via Key Generations) and to interact with the confirmer with respect to these keys. Furthermore, $\mathcal{D}$ can make arbitrary oracle queries in $\mathcal{O}_\mathcal{D}$ = {Sign, DCSSign, $Confirm_{(S,\mathcal{D})}$, $Confirm_{(C,\mathcal{D})}$, $Disavow_{(S,\mathcal{D})}$, $Disavow_{(C,\mathcal{D})}$, $Extract\}$. Then, the distinguisher has to present one fresh message m. After a fair coin is flipped, the adversary is given a corresponding DCS $\sigma' = DCSSign(m, sk_S, pk_C)$, where $b = 0$, or a fake DCS signature chosen uniformly at random from the signature space where $b = 1$. Here the signature space has a finite size that depends only on the security parameter n. Now $\mathcal{D}$ is again allowed to access the above oracles except that it cannot enquire for σ' via any of these oracles. Finally, the distinguisher must output one bit information b' to guess the value of b. We say a DCS scheme with unified verification is invisible, if for any PPT distinguisher $\mathcal{D}$:*

$$|\Pr[b' = b] - 1/2| \leq negl(\lambda).$$

The above probability is taken over the coin tosses of the signer, the confirmer, key generation algorithms and the oracles.

Remark 3. Note we adopt the definition of invisibility by Galbraith and Mao [32], which is slightly stronger than the definition proposed by Camenisch and Michels [18]. What we defined here is actually to require that the adversary cannot decide the validity of a given DCS with respect to its chosen message, without the help of the signer or the confirmer. However, the security requirement in [18], requires that the adversary should be unable to relate a chosen message with a valid DCS from a face DCS. Galbraith and Mao have proved that these two types of invisibility are actually equivalent satisfying some particular properties in the standard model of computation. In addition, we disallow the adversary to have sk_S. Otherwise, it will be trivial for him to distinguish signatures via unified verification protocols.

Definition 5.6. Security for the confirmer (Non-transferability): This is an extended version of the non-transferability definition in Chapter 3. Intuitively, we require the evidence generated in $Confirm$ or $Disavow$ protocols should be untransferable. Namely, although an adaptive PPT adversary $\mathcal{A}$ knows whether a given DCS is valid or not through the interactive verification, it does not gain any knowledge that can be used to convince a third party about the validity of that DCS. In particular, this notion is formalised in the following games considering a PPT simulator $\mathcal{A}'$:

Game-NTR: *Firstly, the adversary $\mathcal{A}$ is given the public key pk_S and pk_C of the signer and the confirmer. It is allowed to make arbitrary oracle queries to $DCSSign$,$Confirm_{(S,V)}$, $Disavow_{(S,V)}$, $Confirm_{(C,V)}$, $Disavow_{(C,V)}$, and $Extract$. Again$\mathcal{A}$ is allowed to create signature-key pairs $(sk_{\mathcal{A}}, pk_{\mathcal{A}})$, and to run $DCSSgin$ and then interact with the confirmer with respect to these keys. Let a string P denote the prover which could be either the signer S or the confirmer C in any interactive protocols, where $P \in \{C, V\}$.*

In some stage, the adversary must present two strings, m and σ', for which it wishes to carry out the $Confirm_{(P,V)}$ (or $Disavow_{(P,V)}$) protocol with the prover. Next a fair coin b is flipped. If $b = 0$, the real prover and $\mathcal{A}$ run the $Confirm_{(P,V)}$ (or

$Disavow_{(P,V)}$) protocol with common input (m, σ', pk_S, pk_C), while the prover's secret input will be sk_P. If $b = 1$, the simulator Sim is plugged in the place of the real prover to run the$Confirm_{(P,V)}$ (or $Disavow_{(P,V)}$) protocol on (m, σ'). Sim is not given either the confirmer's secret key or the signer's secret key, but is allowed to make a single call to an oracle which tells Sim whether the strings m and σ' is a valid DCS w.r.t. pk_S and pk_C.

In parallel, the adversary is allowed to make arbitrary queries to the signer and the confirmer. And in all other interactions except the confirmation (or disavowal) on (m, σ'), the real signer or the real confirmer speaks with the adversary. Finally, $\mathcal{A}$ must output one bit information b' to guess the value of b. The adversary $\mathcal{A}$ wins if and only if $b = b'$, and $\mathcal{A}$'s advantage is defined as $adv_{\mathcal{A}} = \Pr[\mathcal{A}\ wins]$. We say a DCS scheme with unified verification is **non-transferable** *if for any adversary $\mathcal{A}$, there exists a simulator Sim such that for all sufficiently large λ, all $(sk_S, pk_S) \in G_S(1^\lambda)$, and all $(sk_C, pk_C) \in G_C(1^\lambda)$:*

$$adv_{\mathcal{A}} < negl(\lambda).$$

The above probability is taken over the coin tosses of the signer S, the confirmer C, and key generation algorithms G_S and G_C.

Applying the result in Chapter 4, we know another security notion, i.e., transcript simulatability proposed in GMR model [38], is implied by invisibility and NTR-0. So we omit a proof about transcript-simulatability in our security analysis. In fact, the main obstacle to prove transcript-simulatability in our scheme is that the signed ElGamal, our underlying encryption scheme, is even not CPA-secure. However, the generic DCS proposed in [38, 78] both rely on CCA2-secure public key encryption.

Definition 5.6. (Security). *We say a correct designated confirmer signature scheme* *is secure, if it satisfies security for the signer and for the confirmer. Namely, it is existentially unforgeable, non-transferable, and invisible.*

In fact, [38, 78] also studies *security for verifier or unfoolability*, which requires that any DCS confirmed by running *Confirm* protocol must be extractable, and that every alleged DCS confirmed by running *Disavow* protocol must be unextractable. As this property follows the soundness of *Confirm* and *Disavow* protocols [38, 78], we do not separately specify it here.

Finally, from the above description we can see that by introducing additional *Confirm* and *Disavow* protocols for the signer, the formal model for DCS with unified verification can be directly generalised to accommodate DCS with full verification, in which the signer can also confirm and disavow a signature, but not necessarily runs the same protocols as the confirmer.

5.5 The Proposed Scheme

Based on BLS signature scheme [14], which has been reviewed in Section 2.1, we now present a designated confirmer signature scheme with unified verification. Basically, a DCS in our scheme is just the signed ElGamal encryption [72] of a BLS signature. After the scheme description, we shall give more explanations on the construction.

We use a symmetric bilinear map $e : G \times G \to G_t$, where G is a multiplicative cyclic group of prime order q and g is a generator of G. In addition, two cryptographic hash functions H and H' are used. In particular, $H : \{0,1\}^* \to G$ is a full-domain hash function, and $H' : \{0,1\}^* \to \mathbb{Z}_q^*$ is a standard cryptographic hash function.

Key Generation: The signer picks $x_s \in_R \mathbb{Z}_q^*$ as its private key, and computes $y_s = g^{x_s}$ as its public key. Similarly, the confirmer sets its private/public key pair as $(x_c, y_c = g^{x_c})$, where $x_c \in_R \mathbb{Z}_q^*$.

Sign: Given a signer's private key x_s and a message m, output the signature σ= $h^{x_s} \in G$, where h= $H(m) \in G$.

Verify: Given (m, σ), check whether $e(g, \sigma)$ = $e(y_s, h)$ holds, where $h = H(m) \in G$.

DCSSign: After generating a basic signature σ= $H(m)^{x_s}$ for message m by using the signer's private key x_s, output DCS for message m as $\sigma' = (\sigma_1, \sigma_2, s, t)$ by computing:

$$\sigma_1 = y_c^r,\ \sigma_2 = \sigma g^r,\ \text{where } r \in_R \mathbb{Z}_q^*;\ \text{and}$$
$$s = H'(y_c^k, \sigma_1, \sigma_2), t = k + sr \bmod q,\ \text{where } k \in_R \mathbb{Z}_q^*.$$

It is easy to see that σ' is exactly the signed ElGamal encryption [72] under the private/public key pair $(x_c^{-1}, g = y_c^{x_c^{-1}})$, which is equivalent to the confirmer's key pair $(x_c, y_c = g^{x_c})$. Namely, (σ_1, σ_2) is the naive ElGamal ciphertext of basic signature $\sigma = H(m)^{x_s}$ under the key pair $(x_c^{-1}, g = y_c^{x_c^{-1}})$, while (s, t) is a Schnorr signature on message (σ_1, σ_2) under the temporary private/public key pair $(r, \sigma_1 = y_c^r)$.

Extract: Given a message m and an *alleged* DCS $\sigma' = (\sigma_1, \sigma_2, s, t)$, which satisfies $s \equiv H'(y_c^t \sigma_1^{-s}, \sigma_1, \sigma_2)$, the confirmer extracts the basic signature $\sigma = \sigma_2/\sigma_1^{x_c^{-1}}$ if $e(\sigma_2, y_c)/e(\sigma_1, g) = e(h, y_s)^{x_c}$, where $h = H(m)$. Otherwise, $\perp$ is output.

Confirm: Given common input (m, σ', y_s, y_c), where $\sigma' = (\sigma_1, \sigma_2, s, t)$ is an alleged DCS, the confirmer C with the private key x_c can check the validity of the DCS by verifying whether $e(\sigma_2, y_c)/e(\sigma_1, g) = e(h, y_s)^{x_c}$ holds or not, where $h = H(m)$. As $e(h, y_s)^{x_c} \equiv e(h, y_c)^{x_s}$, the signer S with the private key x_s can similarly know the validity of σ' by checking $e(\sigma_2, y_c)/e(\sigma_1, g) = e(h, y_c)^{x_s}$. If σ' is valid, either the confirmer C or the signer S can convince a verifier V of that fact by running the following interactive zero knowledge protocol:

$$PK\{(x_c \vee x_s) : [e(\sigma_2, y_c)/e(\sigma_1, g) = e(h, y_s)^{x_c} \wedge y_c = g^{x_c}]$$
$$\vee[e(\sigma_2, y_c)/e(\sigma_1, g) = e(h, y_c)^{x_s} \wedge y_s = g^{x_s}]\}.$$

Disavow: On input (m, σ', y_s, y_c), where $\sigma' = (\sigma_1, \sigma_2, s, t)$ is an alleged DCS, if $e(\sigma_2, y_c)/e(\sigma_1, g) \neq e(h, y_s)^{x_c}$ or $e(\sigma_2, y_c)/e(\sigma_1, g) \neq e(h, y_c)^{x_s}$, where $h = H(m)$, this means that σ' is an invalid DCS for message m. Then, either the confirmer C or the signer S can run the following interactive zero knowledge protocol with a verifier V to convince this fact:

$$PK\{(x_c \vee x_s) : [e(\sigma_2, y_c)/e(\sigma_1, g) \neq e(h, y_s)^{x_c} \wedge y_c = g^{x_c}]$$
$$\vee[e(\sigma_2, y_c)/e(\sigma_1, g) \neq e(h, y_c)^{x_s} \wedge y_s = g^{x_s}]\}.$$

Note that the above PKs are for "the (in)equality of two discrete logarithms" $\vee$ "the (in)equality of another two discrete logarithms", and each part can be proved easily by using the standard techniques [22, 16, 53]. The implementation details of these zero knowledge proofs are given in Appendix B.

Remark 4. First, note that the idea of building a DCS here is inspired by the Boneh et al's verifiably encrypted signature (VES) scheme [13], which encrypts a basic BLS signature using ElGamal encryption with the adjudicator's key $(x_c, y_c = g^{x_c})$. The adjudicator in VES plays a similar role as the confirmer in DCS. Here, we exploit the same idea but change the format of the ciphertext via effectively setting the confirmer's key pair $(x_c^{-1}, g = y_c^{x_c^{-1}})$. The result is very interesting, as we get a DCS scheme in which the validity of a hidden signature (i.e. DCS) is not publicly visible any more, compared to Boneh et al.'s hidden but publicly verifiable VES.

However, the above resulting scheme is actually not a secure DCS, as it fails to meet invisibility, due to the malleability of naive ElGamal encryption. That is, given a target DCS $(\sigma_1 = y_c^r, \sigma_2 = \sigma g^r)$ for a message m, an adaptive attacker can simply derive the validity of (σ_1, σ_2) by inquiring the validity of $(\sigma_1' = \sigma_1 y_c^{r'}, \sigma_2' = \sigma_2 g^{r'})$ w.r.t. the same message m by selecting a random number r'. Note that such an attack is allowed in the security definition of invisibility. To address this issue, signed ElGamal encryption [72] is exploited to add one Schnorr signature (s, t) showing that the creator of ciphertext (σ_1, σ_2) indeed knows the secret value of r which is used for encryption. Equivalently, this implies that in the proposed scheme the issuer of a DCS $(\sigma_1, \sigma_2, s, t)$ knows the corresponding basic signature σ, as $\sigma = \sigma_2/g^r$. Hence, the above attack does not work any more, since such a DCS has a fixed format and is not malleable.

Remark 5. Note that the proposed DCS scheme is not strongly unforgeable (but is existentially unforgeable) for an attacker who has comprised the confirmer's private key x_c as explained below. Since a valid DCS for a message m has the form of $\sigma' = (\sigma_1, \sigma_2, s, t) = (y_c^r, \sigma g^r, s, t)$ satisfying $s \equiv H'(y_c^t \sigma_1^{-s}, \sigma_1, \sigma_2)$, the attacker with x_c can first extract the basic signature by computing $\sigma = \sigma_2/{\sigma_1}^{x_c^{-1}}$. Then, the attacker can trivially forge another valid DCS $\bar{\sigma}' = (\bar{\sigma}_1, \bar{\sigma}_2, \bar{s}, \bar{t})$ for the same message m. Nevertheless, this does not violate our definition of unforgeability specified in Definition 3, as a successful forger is required to produce a valid DCS on a *new* message, not a previously signed message.

Remark 6. According to Remark 2, both the signer and the confirmer can check a designated confirmer signature's validity or invalidity of an alleged DCS by using their own private keys. Then, either of them can run the same *Confirm* or *Disavow* interactive zero knowledge protocol with a verifier to show whether σ' is valid or not. Hence, to check the validity of a signature the verifier can interact with either the signer or the designated confirmer. Due to this reason, we call our scheme a DCS with *unified verification*. In particular, in our scheme the signer is granted the ability to disavow any invalid designated confirmer signature. This new feature is interesting, as our scheme serves a better extension of undeniable signatures [21], in which there is a disavow protocol for the signer; nevertheless no current DCS schemes except [48], which inspired by our prototype in [81] of this scheme, offer Disavow protocol for the signer.

A Generalised Version As discussed in Section 1, our unified verification DCS can be simply generalised to a full verification version. The idea is to get rid of the "OR" relation in the interactive zero-knowledge (IZK) protocols. For $Confirm_{(S,V)}$, the signer initially checks the validity of a given DCS, then runs a zero-knowledge protocol $PK\{x_s : e(\sigma_2, y_c)/e(\sigma_1, g) = e(h, y_c)^{x_s} \wedge y_s = g^{x_s}\}$. For $Confirm_{(C,V)}$, the ZK protocol will be $PK\{x_c : e(\sigma_2, y_c)/e(\sigma_1, g) = e(h, y_s)^{x_c} \wedge y_c = g^{x_c}\}$. To disavow an invalid signature, either the signer or the verifier firstly checks the invalidity of a given DCS using their own secret key, and then runs $PK\{x_s : e(\sigma_2, y_c)/e(\sigma_1, g) \neq e(h, y_c)^{x_s} \wedge y_s = g^{x_s}\}$ or $PK\{x_c : e(\sigma_2, y_c)/e(\sigma_1, g) \neq e(h, y_s)^{x_c} \wedge y_c = g^{x_c}\}$, respectively. Hence, the above extension accommodates the generalised DCS model, where the signer and the confirmer can confirm (or disavow) signatures via different protocols.

5.5.1 HVIZKs on Confirm and Disavow Protocols

In this section, we show how to run honest verifier interactive zero-knowledge proof (HVIZK), actually Σ-protocols, to complete the *Confirm* and *Disavow* protocols. To this end, we directly adapt the protocols given in [53], and depict the implementation details as below.

In the *Confirm* protocol, the knowledge statement is $PK\{(x_c \vee x_s)) : [e(\sigma_2, y_c)/e(\sigma_1, g) = e(h, y_s)^{x_c} \wedge y_c = g^{x_c}] \vee [e(\sigma_2, y_c)/e(\sigma_1, g) = e(h, y_c)^{x_s} \wedge y_s = g^{x_s}]\}$. So, once the prover (either the signer or the confirmer) and a verifier pre-compute $A_1 = A_2 = e(\sigma_2, y_c)/e(\sigma_1, g)$, $B_1 = e(h, y_s)$, $B_2 = e(h, y_c)$, $C_1 = y_c$, $C_2 = y_s$, and $D_1 = D_2 = g$, they can run the *Confirm* protocol as shown in Figures 5.1 and Figure 5.2 to prove "$(A_1 = B_1^{x_c} \wedge C_1 = D_1^{x_c}) \vee A_2 = B_2^{x_s} \wedge C_2 = D_2^{x_s})$".

In the *Disavow* protocol, the knowledge statement is $PK\{(x_c \vee x_s)) : [e(\sigma_2, y_c)/e(\sigma_1, g) \neq e(h, y_s)^{x_c} \wedge y_c = g^{x_c}] \vee [e(\sigma_2, y_c)/e(\sigma_1, g) \neq e(h, y_c)^{x_s} \wedge y_s = g^{x_s}]\}$. So, once the prover (either the signer or the confirmer) and a verifier pre-compute $A_1 = A_2 = e(\sigma_2, y_c)/e(\sigma_1, g)$, $B_1 = e(h, y_s)$, $B_2 = e(h, y_c)$, $C_1 = y_c$, $C_2 = y_s$, and $D_1 = D_2 = g$, they can run the *Disavow* protocol as shown in Figures 5.3 and Figure 5.4 to prove "$(A_1 \neq B_1^{x_c} \wedge C_1 = D_1^{x_c}) \vee A_2 \neq B_2^{x_s} \wedge C_2 = D_2^{x_s})$".

As mentioned before, both the above *Confirm* and *Disavow* HVIZK protocols should be converted into CZK so that they can be executed with multiple verifiers concurrently.

Common Input:

$A_1, A_2, B_1, B_2, C_1, C_2, D_1, D_2$

<u>Signer S</u>		<u>Verifier V</u>
(with private input x_s)		
$a, b_1, c_1 \in_R \mathbb{Z}_q$,		
$z_1 = {B_1}^{b_1} \cdot {A_1}^{c_1}$,		
$z_2 = {D_1}^{b_1} \cdot {C_1}^{c_1}$,		
$z_3 = {B_2}^{a}$, $z_4 = {D_2}^{a}$;	(1) $\xrightarrow{(z_1, z_2, z_3, z_4)}$	$c \in_R \mathbb{Z}_q$
$c_2 = c - c_1 \bmod q$,	(2) $\xleftarrow{c}$	
$b_2 = a - c_2 x_s \bmod q$.	(3) $\xrightarrow{(b_1, b_2, c_1, c_2)}$	Output Accept iff
		$c_1 + c_2 \equiv c \bmod q$,
		$z_1 = {B_1}^{b_1} \cdot {A_1}^{c_1}$, $z_2 = {D_1}^{b_1} \cdot {C_1}^{c_1}$,
		$z_3 = {B_2}^{b_2} \cdot {A_2}^{c_2}$, $z_4 = {D_2}^{b_2} \cdot {C_2}^{c_2}$.

Figure 5.1: The $Confirm_{(S,V)}$ Protocol in the concrete DCS-UV scheme

Common Input:

$A_1, A_2, B_1, B_2, C_1, C_2, D_1, D_2$

Confirmer C		Verifier V
(with private input x_c)		
$a, b_2, c_2 \in_R \mathbb{Z}_q$, $z_1 = {B_1}^a$, $z_2 = {D_1}^a$, $z_3 = {B_2}^{b_2} \cdot {A_2}^{c_2}$,		
$z_4 = {D_2}^{b_2} \cdot {C_2}^{c_2}$;	(1) $\xrightarrow{(z_1, z_2, z_3, z_4)}$	$c \in_R \mathbb{Z}_q$
$c_1 = c - c_2 \bmod q$,	(2) $\xleftarrow{\quad c \quad}$	
$b_1 = a - c_1 x_c \bmod q$;	(3) $\xrightarrow{(b_1, b_2, c_1, c_2)}$	Output Accept iff $c_1 + c_2 \equiv c \bmod q$, $z_1 = {B_1}^{b_1} \cdot {A_1}^{c_1}$, $z_2 = {D_1}^{b_1} \cdot {C_1}^{c_1}$, $z_3 = {B_2}^{b_2} \cdot {A_2}^{c_2}$, $z_4 = {D_2}^{b_2} \cdot {C_2}^{c_2}$.

Figure 5.2: The $Confirm_{(C,V)}$ Protocol in the concrete DCS-UV scheme

Common Input:

$A_1, A_2, B_1, B_2,$

$C_1, C_2, D_1, D_2.$

Signer S		Verifier V
(with private input x_s)		
$a, b_1, b_2, c_1, e, e', \beta \in_R \mathbb{Z}_q/\{1\}$, $\beta' = ({D_2}^{x_s}/C_2)^a$, $z'_1 = {D_2}^e/{C_2}^{e'}$, $z'_2 = {B_2}^e/{A_2}^{e'}$, $z_1 = \beta^{c_1}{D_1}^{b_1}/{C_1}^{b_2}$,		
$z_2 = {B_1}^{b_1}/{A_1}^{b_2}$;	(1) $\xrightarrow{(\beta, \beta', z_1, z_2, z'_1, z'_2)}$	Iff $\beta \neq 1$ and $\beta' \neq 1$,
	(2) $\xleftarrow{\quad c \quad}$	$c \in_R \mathbb{Z}_q$
$c_2 = c - c_1 \bmod q$,		
$b'_1 = e - c_2 x_s a \bmod q$,	(3) $\xrightarrow{(b_1, b_2, b'_1, b'_2, c_1, c_2)}$	Output Accept iff
$b'_2 = e' - c_2 a \bmod q$;		$c_1 + c_2 \equiv c \bmod q$, $z_1 = \beta^{c_1}{D_1}^{b_1}/{C_1}^{b_2}$, $z_2 = {B_1}^{b_1}/{A_1}^{b_2}$, $z'_1 = \beta'^{c_2}{D_2}^{b'_1}/{C_2}^{b'_2}$, $z'_2 = {B_2}^{b'_1}/{A_2}^{b'_2}$.

Figure 5.3: The $Disavow_{(S,V)}$ Protocol in the concrete DCS-UV scheme

Common Input:

$A_1, A_2, B_1, B_2,$

C_1, C_2, D_1, D_2

Confirmer C		Verifier V
(with private input x_c)		
$a, b'_1, b'_2, c_2, e, e', \beta' \in_R \mathbb{Z}_q/\{1\}$,		
$\beta = (D_1{}^{x_c}/C_1)^a$,		
$z_1 = D_1{}^e/C_1{}^{e'}$, $z_2 = B_1{}^e/A_1{}^{e'}$,		
$z'_1 = \beta'^{c_2} D_2{}^{b'_1}/C_2{}^{b'_2}$,		
$z'_2 = B_2{}^{b'_1}/A_2{}^{b'_2}$;	(1) $\xrightarrow{(\beta, \beta', z_1, z_2, z'_1, z'_2)}$	Iff $\beta \neq 1$ and $\beta' \neq 1$,
	(2) $\xleftarrow{\quad c \quad}$	$c \in_R \mathbb{Z}_q$
$c_1 = c$ - c_2 mod q,		
$b_1 = e$ - $c_1 x_c a$ mod q,	(3) $\xrightarrow{(b_1, b_2, b'_1, b'_2, c_1, c_2)}$	Output Accept iff
$b_2 = e'$ - $c_1 a$ mod q;		$c_1 + c_2 \equiv c$ mod q,
		$z_1 = \beta^{c_1} D_1{}^{b_1}/C_1{}^{b_2}$, $z_2 = B_1{}^{b_1}/A_1{}^{b_2}$,
		$z'_1 = \beta'^{c_2} D_2{}^{b'_1}/C_2{}^{b'_2}$, $z'_2 = B_2{}^{b'_1}/A_2{}^{b'_2}$.

Figure 5.4: The $Disavow_{(C,V)}$ Protocol in the concrete DCS-UV scheme

5.6 Security Analysis

5.6.1 Complexity Assumptions

We introduce some complexity assumptions required in our proposal as below. Let $negl(n)$ denote any negligible function that grows slower than n^{-v} for any positive integer v and for all sufficiently large integer n. $x \in_R X$ denotes an random element x is picked from set X uniformly, and $x_1, x_2, ..., x_n \overset{R}{\leftarrow} X$ denotes $x_1, x_2, ..., x_n$ are random elements picked from set X uniformly. All the other alphabets and symbols follow the previous meanings.

Definition 5.6. Computational Diffie-Hellman Assumption (CDH). *Given $g, g^a, g^b \in G$, no probabilistic polynomial-time (PPT) algorithm can output $g^{ab} \in G$ with non-negligible probability, where $a, b \in_R \mathbb{Z}_q^*$.*

Now we propose a new assumption, called "Decisional-coefficient-Linear (D-co-L, in short) assumption", to serve our security analysis in Theorem 3. We shall provide more confidence towards the D-co-L assumption in the generic bilinear groups.

Definition 5.7. Decisional-coefficient-Linear Assumption (D-co-L): *With $g \in G$ and a pairing e described as above,given a tuple $(g, g^a, g^b, g^w, g^{by}, g^{wa+y}, g^z)$, where $a, b, w, y, z \xleftarrow{R} \mathbb{Z}_q^*$, no PPT algorithm A can distinguish between g^{wa+y} and a random element g^z in G. Formally, for any PPT algorithm A, for $(g, g^a, g^b, g^w, g^{by}, g^{wa+y}, g^z)$, where $a, b, y, w, z \xleftarrow{R} \mathbb{Z}_q^*$, we define the advantage of A:*

$$Adv_A^{D\text{-}co\text{-}L}(n) = \mid \Pr[A(t, g^a, g^b, g^w, g^{by}, g^{wa+y}) = 1] - \Pr[A(t, g^a, g^b, g^w, g^{by}, g^z) = 1] \mid$$

The probability is over the uniform random choice of the parameters to A, and over the coin tosses of A. We say the decisional-coefficient-linear assumption (T, ϵ)-holds, if there is no such A, which runs in time at most T and $Adv_A^{D-co-L}(n)$ is at least ϵ.

We prove a lower bound on the computational complexity of the D-co-L problem in the generic group model following the proof techniques in [8], [11], [10] and [47], to give more confidence towards the D-co-L assumption. In this model, the adversary can only perform equality tests, because group elements of G and G_t are encoded as unique random strings. Three oracles are assumed to perform operations between group elements, including computing the group action in G and G_t, as well as the bilinear pairing $e : G \times G \to G_t$. Two injective functions are used to model this opaque encodings, $\xi : \mathbb{Z}_q \to \{0,1\}^*$ for group G, where a group element $g^t \in G$ is represented as a string $\xi(t)$, and an analogous function $\xi' : \mathbb{Z}_q \rightarrow \{0,1\}^*$ for group G_t.

Theorem 5.2. Let A be an algorithm that solves D-co-L problem in the generic group model, making at most l queries to the oracles which compute the group action in G and G_t, and the oracle which computes the bilinear pairing e. Suppose $a, b, w, y, z \overset{R}{\leftarrow} \mathbb{Z}_q$, $d \overset{R}{\leftarrow} \{0,1\}$, and ξ, ξ' are two random encoding functions as defined above for G and G_t respectively. Let $t_d = wa + y$ and $t_{1-d} = z$. Then about A's advantage, we have:

$$\epsilon := \mid Pr[A(q, \xi(1), \xi(a), \xi(b), \xi(w), \xi(by), \xi(t_0), \xi(t_1), \xi'(1)) = d] - \frac{1}{2} \mid \leq 4(l+8)^2/q.$$

Proof: We construct a simulation algorithm S that simulates the generic group oracles for A without committing to values for a, b, w, y, t_0, t_1. S keeps track of the group elements by their discrete logarithms (group exponents) to the generators $g \in G$ and $e(g,g) \in G_t$. Since the variables a, b, w, y, t_0, t_1 are undetermined, these discrete logarithms are polynomials in $\mathbb{Z}_q[a, b, w, y, t_0, t_1]$ with coefficients in $\mathbb{Z}_q$, which we denote by ρ_i for exponents in G and ρ'_i for exponents in G_t. S then maps these group exponents to arbitrary distinct strings it gives to A, i.e., $\xi_i = \xi(\rho_i)$ for ρ_i in G and $\xi'_i = \xi'(\rho'_i)$ for ρ'_i in G_t. S maintains two lists of pairs, i.e., $L = \{(\rho_i, \xi_i) : i = 0, 1, ..., \tau\}$ and $L' = \{(\rho'_i, \xi'_i) : i = 0, 1, ..., \tau'\}$, under the condition that at step κ in the game, $\tau + \tau' = \kappa + 8$. These lists are initialised at step $\kappa = 0$: ξ_0, ξ_1, ξ_2, ξ_3, ξ_4, ξ_5, ξ_6 and ξ'_0 are set to arbitrary distinct strings in $\{0,1\}^*$, which binds to $\rho_0 = 1$, $\rho_1 = a$, $\rho_2 = b$, $\rho_3 = w$, $\rho_4 = by$, $\rho_5 = t_0$, $\rho_6 = t_1$ and $\rho'_0 = 1$ respectively, where two counters are initialised $\tau = 7$ and $\tau' = 1$.

Initially, S gives all the strings created above to A, and then simulates the oracles for A as below. Without loss of generality, we assume that A only queries S on legitimate strings that were previously revealed.

Group Actions. To compute the product/division of two operands in G represented as ξ_i and ξ_j, where $0 \leq i, j \leq \tau$, S computes $\rho_\tau = \rho_i \pm \rho_j$. If $\rho_\tau = \rho_k$ for some $k < \tau$, set $\xi_\tau = \xi_k$; otherwise, it set ξ_τ to a string in $\{0,1\}^*$ distinct from $\xi_0, ..., \xi_{\tau-1}$. S then increases τ by one, adds the new pair (ρ_τ, ξ_τ) to the list L and give ξ_τ to A. Group action queries in G_t are treated similarly, this time by working with the list L' and the counter τ'.

Pairings. To compute the product of two operands in G represented as ξ_i and ξ_j, where $0 \leq i, j \leq \tau$, S computes $\rho'_{\tau'} = \rho_i \rho_j$. If $\rho'_{\tau'} = \rho'_k$ for some $k < \tau'$, set $\xi'_{\tau'} = \xi'_k$; otherwise, it set $\xi'_{\tau'}$ to a string in $\{0,1\}^*$ distinct from $\xi'_0, ..., \xi'_{\tau'-1}$. S then increases τ' by one, adds the new pair $(\rho'_{\tau'}, \xi'_{\tau'})$ to the list L' and give $\xi'_{\tau'}$ to A.

Note that at any time in the game, the total degree of any polynomial to represent an element in G is at most 2, and the total degree of any polynomial to represent an element in G_t is at most 4.

After at most l queries, A terminates and returns a guess $\hat{d} \in \{0,1\}$. At this point, S chooses random $\hat{a}, \hat{b}, \hat{w}, \hat{y}, \hat{z} \xleftarrow{R} \mathbb{Z}_q$. Consider $t_d = \hat{w}\hat{a} + \hat{y}$ and $t_{1-d} = \hat{z}$ for both choices of $d \in \{0,1\}$. The simulation provided by S is perfect and reveals nothing to A about d unless the chosen random values for the variables a, b, w, by, z, t_0, t_1 leads to an equality relation between these intermediate values that is not an equality of polynomials. This happens if either of the following events happens::

1. $\rho_i(a, b, w, y, wa + y, z) - \rho_j(a, b, w, y, wa + y, z) = 0$ but $\rho_i \neq \rho_j$ for some $0 \leq i, j \leq \tau$.

2. $\rho'_i(a, b, w, y, wa + y, z) - \rho'_j(a, b, w, y, wa + y, z) = 0$ but $\rho'_i \neq \rho'_j$ for some $0 \leq i, j \leq \tau'$.

3. any relation similar to the above in which $wa + y$ and z have been exchanged;

Because the group operations in G and G_t are implemented by the addition/subtraction between polynomials in L and L' respectively, and the pairing operations are implemented by the multiplication of polynomials in L, it is unable for the adversary to trivially obtain the knowledge of a multiple of the polynomial $wa + y$ via these operations.

When any of the above events occurs, S's responses to A's queries deviate from the real oracles' responses. Furthermore, in this case d is independent from algorithm A's view and A's probability of making a correct guess is exactly $1/2$. Since $\rho_i - \rho_j$ for fixed i and j is of degree at most 2, it equals zero for a random assignment of the variables in $\mathbb{Z}_q$ with probability at most $2/q$. Similarly, for fixed i and j, $\rho'_i - \rho'_j$ becomes zero with probability $4/q$. The same probabilities can be found in the third case. Therefore, we have that A makes a correct guess with advantage bounded by $\epsilon \leq 2 \cdot (\binom{\tau}{2}\frac{2}{q} + \binom{\tau'}{2}\frac{4}{q})$. Since $\tau + \tau' \leq l + 8$, we have $\epsilon \leq 4(l+8)^2/q$. $\square$

5.6.2 Security Proofs

Under the standard CDH assumption, the BLS signature scheme [14] is provably secure in the random oracle model. The unforgeability of our DCS relies upon the security of BLS scheme, *without direct use of random oracles*. The new D-co-L assumption, proposed in this work, gives rise to the invisibility of our DCS scheme in the random oracle model.

Because our $Confirm$ and $Disavow$ protocols are based on special honest verifier zero knowledge proofs (SHVZK) and can be converted to CZK protocols according to the transformation methodologies in section 3, the notion of non-transferability in Def 5.6 follows in a straightforward manner from the concurrent zero-knowledge property of the proofs, and we omit a proof on non-transferability.

Theorem 5.3. If the BLS signature scheme is (t',q'_H,q'_S,ε')-secure against existential forgery, then the proposed DCS scheme is (t,q_H,q_S,ε)-secure against existential

forgery w.r.t. Definition 4, where $q'_H = q_H$, $q'_S = q_S$, $\varepsilon' = \varepsilon$, $t' \leq t + 6c \cdot (q_S + q_C + q_D)$, where c is a constant, denoting the time to compute one pairing evaluation, one exponentiation in G, and one exponentiation in G_t.

Proof. Given a forgery algorithm F for the proposed DCS scheme, we shall construct a forgery algorithm F' for the underlying BLS signature scheme. For presentation simplicity, we assume F *behaves well* in the random oracle model, i.e., F always requests the hash of a message m before requesting a designated confirmer signature.

The BLS forger F' is given the signer's public key y_s for which the private key is unknown to F' and has access to the *Sign* and hash oracles. As the challenger for F, F' simulates and runs interactions with F as follows.

Setup. F' generates a key-pair (x_c, y_c) randomly by running G_c, which serves as the confirmer's key pair. Then F' runs F, providing it as input the public keys y_s and y_c, and also the confirmer's private key x_c.

Hash Queries. When F requests a hash on m, F' makes a query for m to its own hash oracle, and receives some value $h \in G$, then it responds h to F.

DCSSign Queries. When F requests a DCS on some m (it would have already queried the hash oracle on m), F' queries its own *Sign* oracle on message m, obtaining $\sigma \in G$. Then F' selects two random numbers $r, k \in \mathbb{Z}_q^*$, generates a DCS $\sigma' = (\sigma_1, \sigma_2, s, t)$ according to Eq. (5) and returns it to F.

Confirm and Disavow Queries. Whenever F asks to run either $Confirm_{(S,F)}$ or $Disavow_{(S,F)}$ protocol w.r.t. a DCS message-signature pair (m, σ'), F' can first checks the validity of σ' by using the secret x_c and then convinces F by running $Confirm_{(C,F)}$, $Disavow_{(C,F)}$ respectively in the role of the confirmer C. Note that in the view of point of algorithm F, such interactions are indistinguishable from those running in the role of the signer S. Note that for the proof of unforgeability, we actually need the witness indistinguishability property of Confirm and Disavow protocols, which does hold for the HVIZK protocols presented in Appendix B (without the need of transforming them into CZK with unprogrammable RO).

Output. Finally, if F halts declaring failure, F' declares failure too. Otherwise, F provides a valid and nontrivial DCS $\sigma'^* = (\sigma_1^*, \sigma_2^*, s^*, t^*)$ to F' on a message m^*. Then, F' computes $\sigma^* = \sigma_2^*/(\sigma_1^*)^{x_c^{-1}}$, which is a valid BLS signature on m^* under signer's public key y_s. A nontrivial forgery means that F did not query the *DCSSign* oracle on m^*, for which F' did not query its *Sign* oracle on m^*. Hence, (m^*, σ^*) forms a nontrivial BLS forgery.

Now we analyze the success probability and the running time of F'. Algorithm F' succeeds whenever F does, so the success probability of F equals to that of F', i.e., $\varepsilon = \varepsilon'$. The running time of F' is the running time of F plus the time it takes to respond q_H hash queries and q_S *DCSSign* queries, to run q_C *Confirm* queries and q_D *Disavow* queries, together the time to transform the final forged DCS into a BLS signature. Hash queries impose no overhead. Each *DCSSign* query requires F' to perform three exponentiations in G. For each *Confirm* query, F' will evaluate four paring computations and five exponentiations in G_t, while each *Disavow* query requires five paring computations and six exponentiations in G_t. The final signature transformation needs one exponentiation in G. Denote the time for pairing computation by pr, the time for exponentiation in G by ex_G, and the time for exponentiation in G_t by ex. We get that the total running time t' for F' is at most $t+(3q_S+1)\cdot ex_G+5(q_C+q_D)\cdot pr+(5q_C+6q_D)\cdot ex$.

In a summary, if F can $(t, q_H, q_S, q_C, q_D, \varepsilon)$-forges a DCS in the proposed DCS scheme, then F' can $(t', q'_H, q'_S, \varepsilon')$-forges a BLS signature, where $q'_H = q_H$, $q'_S = q_S$, $\varepsilon' = \varepsilon$, and $t' \leq t + 6c \cdot (q_S + q_C + q_D)$, where $c = pr + ex_G + ex$ is a constant. $\square$

Theorem 5.4. Under the D-co-L assumption, the proposed DCS scheme is invisible in the RO model.

Proof: Suppose a challenger algorithm $\mathcal{C}$ is given the D-co-LA challenge, i.e., to distinguish two tuples, $(g, g^a, g^b, g^w, g^{by}, g^{wa+y})$ and $(g, g^a, g^b, g^w, g^{by}, g^z)$ where g is a generator in a multiplicative cyclic group G with prime order q. A pairing is constructed as $e : G \times G \to G_t$ where G_t is another multiplicative cyclic group with the same order q. Consider the invisibility game modeled in Definition 5, C needs to simulate a DCS environment for some PPT distinguisher $\mathcal{D}$, in which $\mathcal{D}$ tries to distinguish two pairs: (m, sig) and (m, sig_R), where $sig = DCSSign(m, sk_S, pk_C)$ and sig_R is chosen uniformly at random from the signature space. $\mathcal{D}$ can access the hash oracle, $Sign$, $DCSSign$, $Confirm$, $Disavow$ oracles before and after the challenge request phase. So, C can setup a DCS scheme instance and simulate the game for D as below.

First, $\mathcal{C}$ sets $pk_C = g^b$, and $pk_S = g^w$. Then, C simulates all the oracles for D as follows:

$Hash$ **query by** $\mathcal{D}$: Upon receiving D's queried message m, C picks u randomly and sets $H(m) = g^u$. Then, add (m, u) to a H list, which is initially empty.

$Sign$ **query by** $\mathcal{D}$: For a queried message m, C first checks if $(m, u) \in H$ for some u. If yes, outputs basic signature $\sigma = g^{wu}$. Otherwise, selects u randomly, adds (m, u) to the H list, and outputs $\sigma = g^{wu}$.

$DCSSign$ **query by** $\mathcal{D}$: For a queried message m, similarly C can get a unique tuple $(m, u) \in H$ for some u. Then, C computes basic signature $\sigma = g^{wu}$. By picking a random r, C computes $\sigma_1 = pk_C^r = g^{br}$ and $\sigma_2 = \sigma \cdot g^r = g^{wu+r}$. Since r as 'the signing key', $\mathcal{C}$ can simply produce a Schnorr signature (s, t) for message (σ_1, σ_2). Finally, C outputs $(\sigma_1, \sigma_2, s, t)$ to D.

$Extract$ **query by** $\mathcal{D}$: For given an alleged DCS $(\sigma_1, \sigma_2, s, t)$ for message m, C outputs $\perp$ if it cannot find m in the H list. Otherwise, retrieves (m, u) from the H list and computes $\overline{\sigma} = g^{wu}$. Then, if $e(\sigma_2, g^b) \neq e(\sigma_1, g) \cdot e(\overline{\sigma}, g^b)$, C outputs $\perp$.

Otherwise, C knows that the queried DCS signature-message pair is valid, so it outputs the basic signature $\overline{\sigma} = g^{wu}$.

$Confirm$/$Disavow$ **query by** $\mathcal{D}$: Similar with the $Extract$ oracle, $\mathcal{C}$ can check the queried DCS's validity easily. To convince D the validity of queried DCS, $\mathcal{C}$ just needs to straightforwardly run the underlying CZK simulator.

For the challenge message m' submitted by $\mathcal{D}$, C computes the DCS signature sig, in case the coin toss is head, as follows. Let $H(m') = g^a$, which implicitly sets the underlying BLS signature for m' as $H(m')^{sk_S} = g^{wa}$. Then, C sets $\sigma_1 = g^{by}$, where y is treated as the randomness used in the original DCSSign phase, and $\sigma_2 = g^{wa+y}$. For the Schnorr signature part, i.e., constructing (s, t), we assume $\mathcal{C}$ also controls $H'(\cdot)$ oracle. Thus it can simply selects s, t randomly, and let $g^{bk} = g^{bt}g^{-brs} \bmod q$. Note that here the public key for the internal Schnorr signature is g^{br}, the secret key is r and the base of logarithm is g^b. Finally, C outputs the challenge DCS for message m as $sig = (\sigma_1, \sigma_2, s, t)$, which is a valid DCS on message m.

In case the coin toss is tail, C outputs a fake DCS $sig_R = (\sigma_1, \sigma_2, s, t)$ for message m', where $\sigma_1 = g^{by}$, $\sigma_2 = g^z$, and (s, t) is a simulated Schnorr signature showing that $(\sigma_1 = g^{by}, \sigma_2 = g^z)$ is a well formed ElGamal encryption.

After that, C can continuously answer D's oracle queries as simulated above. Finally, if $\mathcal{D}$ can distinguish the two signatures sig and sig_R by outputting a correct guess bit b', w.r.t. m' with a non-negligible advantage, it is straightforward to see that C can output the same bit b' to solve the given challenge also with a non-negligible advantage. $\square$

	GW [45]	GMR [38]	WBWB [78]	HWS [48]	Our Scheme
Random Oracle	No	No	Yes	No	Yes
Underlying Signatures	[26, 44, 37]	Any	Any	Not known	BLS [14]
$Confirm_{(C,V)}$	320ex	25ex	15ex	17ex+14pr	12ex+8pr
$Disavow_{(C,V)}$	generic ZK	60 ex	16ex	23.5ex+14pr	14.5ex+8pr
Signature Size (bits)	81,920	8,192	1,984	513	682

Table 5.2: Comparison of The Concrete DCS-UV with Some Existing DCS Schemes

5.7 A Comparison

We give a brief comparison between our DCS proposal and other knowing efficient schemes. Here, we compare these DCS schemes according to four categories, i.e., whether the scheme relies on the random oracle model [5], which kinds of underlying basic signatures are used, how about the communication efficiency, and what the signature size is. For signature size, we estimate all schemes with equivalent 1024-bits RSA security. We use pr and ex to denote the time for computing a pairing and an exponentiation in G and G_t, respectively. Note that the communication costs are estimated on the running time of $Confirm_{(C,V)}$ for consistency, and already include the overheads introduced by the transformation from HVIZK to CZK. According to Table 1, our scheme has a smaller signature size over WBWB scheme [78], which is also provably secure in the random oracle model. Comparing to Huang et al's scheme [48], we have higher communication efficiency, since their HVIZK requires more computational costs when transformed into CZK version, and our scheme is conceptual simpler.

5.8 Summary

Based on BLS short signature [14] we presented a new efficient designated confirmer signature (DCS) scheme that additionally enables the signer to disavow any invalid signatures. We call such a scheme as a DCS with full verification. As DCS has been considered for the extension of undeniable signatures, we believe this new feature is attracting in potential applications of DCS, like fair exchange [1] of digital commitments between two users over the Internet. Moreover, our scheme achieves the unified verification, as both the signer and confirmer just use the same $Confirm$ or $Disavow$ protocol to convince a verifier that an alleged DCS is valid or invalid, respectively. Based on security models given in [18, 32], we have proposed a new security model to accommodate a DCS with unified verification, and showed the security of the proposed scheme in the random oracle model under a newly introduced computational assumption, which is independent of interest. In addition, we have proposed a very efficient way that transforms Σ-protocols into concurrent zero knowledge protocols. As the future work, it would be very interesting to build new efficient DCS schemes with unified or full verification.

Chapter 6

A Generic DCS Construction with Full Verification

6.1 Introduction

At Aisacrypt '05, Gentry, Molnar, and Ramzan[38] proposed a generic DCS scheme (the GMR scheme for short) that involves the use of a signature on a commitment and a separate encryption of the randomness used for commitment. By adding this "layer of indirection", the underlying protocols in the GMR scheme have efficient instantiations that can avoid to use general zero-knowledge proofs for NP statements, and furthermore the performance of these protocols is not tied to the selection of underlying signature scheme, as the signature itself is publicly verifiable.

However, as pointed by Wang et al [78], the GMR scheme is flawed and does not meet the security requirement of invisibility. In particular, they discovered a reasonable attack based on the observation that in the GMR scheme, the ciphertext c of the randomness r could be re-used in different signatures. The technique of such an attack can be found in the section 4.2 of [78], and we omitted the details.

After we built the concrete designated confirmer signatures scheme in chapter 5, it seems quite natural to explore a generic construction that still retains the feature of *full verification*, i.e., either the signer or the confirmer can interactively verify arbitrary signatures, provide a convincing proof when confirming a valid signature or when disavowing an invalid signature. The main task of this chapter is to contribute to a generic DCS scheme with full verification.

Our Contributions Inspired by the interesting GMR scheme [38] and the improved GMR scheme in [78], we propose the first generic DCS scheme that supports full verification. The main idea of our construction is, to issue a DCS, the targeted message is initially sealed in a commitment φ. Then the randomness r used to open the commitment is doubly encrypted, that is, two ciphertexts say c_1 and c_2 will be generated as part of the DCS which are the encryptions on the randomness with regard to the signer or the confirmer's public key. The final output DCS is a combination of φ, c_1, c_2 and σ, where σ' is an ordinary signature on φ by using the signing key. To confirm or disavow such a DCS, either the signer or the confirmer can simply decrypt one of the ciphertexts using its private key to get the witness, i.e., the randomness. By checking the correctness of the commitment, the prover can later provides a ZK proof of knowledge for the equality of the randomness existed in the commitment and in the ciphertext.

Since our construction is a straight inheritance of the GMR transformation, our scheme enjoys the similar benefits of the former, i.e., the proposed generic scheme gives rise to an efficient and generic DCS construction without appealing to both random oracles and general zero-knowledge proofs. To avoid the re-used randomness by any adaptive adversary, we draw on the solution introduced by [78], i.e., let the underlying IND-CCA2 secure encryption scheme support the use of labels. In particular, we should let the confirmer be aware of the "context" of the ciphertext c meaning that

c is created with respect to which message m and which verification key. This is the reason that we introduce such a kind of specific encryption schemes in section 2.2.

Organizations After the background information of DCS schemes in this section, we present two cryptographic primitives in section 2, that is, the commitment scheme which is used as a "layer of indirection", to achieve efficient instantiations, as well as the public key encryption scheme that supports the use of labels to enhance the security. In section 3, we propose a generic transformation to convert any digital signatures into designated confirmer signatures. And we give the related security analysis of our scheme in section 4. We show how to efficiently instantiate the proposed scheme by choosing specific building blocks in section 5. Section 6 concludes our work of this chapter.

6.2 Cryptographic Primitives

6.2.1 Commitment Schemes

Commitment schemes play an important role in cryptography and their use is of particular importance within cryptographic protocols. A commitment can be viewed as the "digital" analog of a safe or a sealed envelope. During the so-called commit phase, a player (the sender) wants to commit on a value (or a bitstring) to a receiver such that the latter cannot deduce information about the committed value (*hiding* property). A second phase is revealing the commitment by disclosing some extra information allowing the receiver to learn and check the committed value. The value chosen during the commit phase must be the only one that the sender can compute and that validates during the revealing phase (*binding* property). In fact, the binding property ensures that between the two phases, the sender is not able to change its mind so that it should be impossible for him to open the commitment on a different value from the committed one.

An obvious application of commitment schemes are sealed-bid auctions. Each participant with a key puts his bid into his lockable box, and submits the box to the auctioneer. On receiving all bids, the auctioneer also requests the keys from those participants, unlock the boxes publicly and announces the winner. The important aspects of commitment schemes, the hiding property and the binding property, are reflected in this example: the actual bid should be kept secret until the bidding phase is over, and also no bidder should be able to change his bid after seeing a previously disclosed opponent's bid.

Perdesen Commitment Scheme An example of an information-theoretically hiding commitment scheme is the Perdesen commitment scheme [67], which is binding under the discrete logarithm (**DL**) assumption. We introduce the scheme with minor changes.

Let p and q be two large primes such that q divides $p-1$. G_q is a unique subgroup of $\mathbb{Z}_p^*$ of order q , and g is a generator of G_q. Let h be an element of G_q such that no body knows $\log_g h$.

The committer commits himself to a value $m \in \mathbb{Z}_q$ by choosing $t \in \mathbb{Z}_q$ at random, and computing

$$E(m,t) = g^m h^t$$

Such a commitment can be simply opened by revealing the value of m and t. The scheme has been proved to be statistically hiding and computationally binding under DL assumption. In particular, the commitment $E(m,t)$ reveals no information about m, and the committer cannot open a commitment to m' such that $m' \neq m$ unless it can compute $\log_g h$.

6.2.2 Secure Encryption Scheme with Labels

An interesting attack on the invisibility of DCSs, is to link the validity of the challenge signature to a reconstructed new signature, where the new signature has re-used the same randomness or the same key values existed in producing the challenge signature. Note such kind of attacks has been identified in chapter 3 and in [78].

To enhance the security of the scheme and to resist this attack, we should let the confirmer know the "context" of the ciphertext c meaning that c is created with respect to which message and which verification key. In particular, we introduce an adaptation of Paillier-based encryption scheme [65] that proposed by Camenisch and Shoup [19]. This so-called "CS-Paillier cryptosystem" encryption scheme supporting the use of labels, is an ideal solution for offering the resistance to the above attack.

We first introduce the notation system that is used to describe the scheme below. For a real number a, $\lfloor a \rfloor$ denotes the largest integer $b \leq a$, and $\lceil a \rceil$ the smallest integer $b \geq a$. For positive real numbers a and b, $[a]$ denotes the set $\{0, ..., \lfloor a \rfloor - 1\}$.

The Scheme Description The IND-CCA2 security of this scheme relies on the decisional composite residuosity assumption (**DCRA**) in $\mathbb{Z}^*_{n^2}$, where $n = pq$ is the product of two Sophie-Germain primes p and q (i.e., there exist two primes p_0 and q_0 such that $p = 2p_0 + 1$ and $q = 2q_0 + 1$). Informally, the DCRA states that it is intractable to distinguish random elements from $\mathbb{Z}^*_{n^2}$ and random elements from the subgroup consisting of all n-th powers of elements in $\mathbb{Z}^*_{n^2}$. We give the brief review of this encryption scheme as below.

The user generates a composite modulus $n = pq$ as above. The user's public key includes a collision-resistant hash function H, $h = 1 + n$, a random $g' \in \mathbb{Z}^*_{n^2}$, and values $g = g'^{2n}$, $y_1 = g^{x_1}$, $y_2 = g^{x_2}$, and $y_3 = g^{x_3}$, where x_1, x_2, $x_3 \in_R [n^2/4]$ constitute the private key. Define a function $abs(\cdot)$: $\mathbb{Z}_{n^2} \to \mathbb{Z}_{n^2}$ as $abs(a) = a$ if

$0 \leq a \leq n^2/2$, or $abs(a) = n^2 - a \bmod n^2$ if $n^2/2 < a \leq n^2$.

To encrypt a value $r \in [n]$ with a label $L \in \{0,1\}^*$, the sender picks $t \in_R [n/4]$ and computes a triple (u, e, v) by $u = g^t$, $e = y_1^t h^r$, and $v = abs((y_2 y_3^{H(u,e,L)})^t)$. The resulting ciphertext (u, e, v) with label L can be decrypted as follows. First, the user checks whether$abs(v) \equiv v$ and $u^{2(x_2+H(u,e,L)\cdot x_3)} \equiv v^2$. If any check fails, output $\perp$. Otherwise, the user computes $\hat{r} = (e/u^{x_1})^{2k}$ for $k = 2^{-1} \bmod n$. If $\hat{r}$ is of form h^r for some $r \in [n]$ (i.e., $\hat{r} - 1$ is divisible by n), then output $r = (\hat{r} - 1)/n \in [n]$. Otherwise, output $\perp$.

6.3 A Generic Construction of DCS with Full Verification

Intuition Behind One may find that, if using the paradigm of "sign-and-encrypt" method, it is not so easy to let the signer prove a DCS is invalid. Because without the decryption key, the signer has to prove the encrypted value is not a valid basic signature on the required message. Usually, to prove such a statement, the signer has to present a general zero-knowledge proofs, while such kinds of proofs always involve a reduction step to an NP-complete language (e.g., the language representing graphs that are three colorable), and cannot really be used in practice. Our main idea is, the targeted message is initially sealed in a commitment. The randomness used to open the commitment is then repeatedly encrypted, that is, two ciphertexts will be generated as part of the DCS which are the encryptions on the randomness with regard to the signer and the confirmer's public key. To confirm or disavow any DCS, either the signer or the confirmer can simply decrypts one of the ciphertexts using its private key to get the witness, i.e., the randomness. By checking the correctness of the commitment, the prover can later provide a ZK proof of knowledge for the equality of the randomness existed in the commitment and in the ciphertext.

The Scheme Description

To setup the scheme, the following building blocks of cryptographic primitives are adopted: a statistically hiding and computationally binding commitment scheme $Comm = (Com, CheckReveal)$; an IND-CCA2 secure public key encryption scheme which supports the use of "labels": $PKE = (PKE\_Gen, Enc, Dec)$; and an EUF-CMA digital signature scheme $DS = (DS\_Gen, Sig, Ver)$. The following algorithms/protocols describe the details of our scheme.

$KGen$: The signer S generates a signing key-pair $(sk_S, vk_S) \leftarrow DS\_Gen(1^\lambda)$ for any EUF-CMA digital signature scheme DS; Both the signer and the confirmer generate public and private keys for any IND-CCA2 secure public key encryption scheme supporting the use of labels: $(x_S, y_S) \leftarrow PKE\_Gen(1^\lambda)$, $(x_C, y_C) \leftarrow PKE\_Gen(1^\lambda)$.

$Sign$: To issue an ordinary signature on a message m, the signer S computes a statistically hiding and computationally binding commitment $\psi = Com(m, r)$ with the randomness r, and creates $\sigma = Sig(sk_S, \psi)$; The basic signature is $\sigma^* = (\sigma, r)$.

$Verify$: On input an basic DCS signature $\sigma^* = (\sigma, r)$ for a message m, this algorithm returns the output of $Ver(\psi, \sigma, vk_S)$, where $\psi = Com(m, r)$.

$DCSSign$: To issue a DCS on a message m, the signer S computes a statistically hiding and computationally binding commitment $\psi = Com(m, r)$ with the randomness r, and creates $\sigma = Sig(sk_S, \psi)$; In addition, S also computes two encryptions, i.e., $c_1 = Enc(y_S, r)$ and $c_2 = Enc(y_C, r)$, and prepares a zero-knowledge proof π_0 shows that c_1 and c_2 are properly prepared. In particular, π_0 should be a *non-interactive* zero knowledge (NIZK) protocol shows that both c_1 and c_2 encrypt the same randomness that is used to compute the commitment ψ. The output DCS is $\sigma' = (\psi, \sigma, c_1, c_2, \pi_0)$.

$Confirm_{(S,V)}$: On receiving a message-DCS pair $(m, \sigma') = (m, (\psi, \sigma, c_1, c_2, \pi_0))$, signer S first checks whether σ is signed on ψ. S aborts if this check fails. Also S

aborts if π_0 is invalid. Otherwise, S decrypts c_1 to get a value r, and then checks if $\psi \equiv Com(m, r)$. If any step of this procedure fails, S executes $Disavow_{(S,V)}$ protocol. Otherwise, using its private key x_S S runs the interactive protocol π_1 with the verifier. In particular, π_1 is a ZK proof of knowledge of a value r such that $c_1 = Enc(y_S, r)$ and $\psi = Com(m, r)$.

$Disavow_{(S,V)}$: On receiving a purported message-DCS pair $(m, \sigma') = (m, (\psi, \sigma, c_1, c_2, \pi_0))$, signer S checks if c_1 is a valid encryption of some r which can be decrypted by using its private key x_S. If not, it performs a ZK proof of knowledge showing that c_1 is not well-formed. Otherwise, S computes $r' = Dec(c_1, x_S)$. If $\psi \neq Com(m, r')$, S provides a ZK proof of knowledge that there is a value r' such that $\psi \neq Com(m, r')$ and $c_1 = Enc(y_S, r')$.

$Confirm_{(C,V)}$: On receiving a message-DCS pair $(m, \sigma') = (m, (\psi, \sigma, c_1, c_2, \pi_0))$, confirmer C first checks whether σ is signed on ψ. C aborts if this check fails. Also C aborts if π_0 is invalid. Otherwise, C decrypts c_2 to get a value r by using its private key x_C, and then checks the equation $\psi \equiv Com(m, r)$. If any step of this procedure fails, C executes $Disavow_{(C,V)}$ protocol. Otherwise, using its private key x_C C runs the interactive protocol π_2 with the verifier. In particular, π_2 is a ZK proof of knowledge of a value r such that $c_2 = Enc(y_C, r)$ and $\psi = Com(m, r)$.

$Disavow_{(C,V)}$: On receiving a purported message-DCS pair $(m, \sigma') = (m, (\psi, \sigma, c_1, c_2, \pi_0))$, confirmer C checks if both c_2 is a valid encryption of some r. If not, it performs a ZK proof of knowledge such that c_2 is not well-formed. Otherwise, C computes $r' = Dec(c_2, x_C)$. If $\psi \neq Com(m, r')$, C provides a ZK proof of knowledge showing that there is a value r' such that $\psi \neq Com(m, r')$ and $c_2 = Enc(y_C, r')$.

$Extract$: On input m and $\sigma' = (\psi, \sigma, c_1, c_2, \pi_0)$, the confirmer C first checks whether σ is signed on ψ, then decrypts c_2 to get a value r by using its private key x_C, and checks if $\psi \equiv Com(m, r)$. If any of the procedure fails, C outputs $\perp$. Otherwise,

C outputs the basic signature $\sigma^* = (\sigma, r)$.

Remark 1: In multi-signer settings, to enhance the invisibility of the GMR scheme, we should let the prover (the confirmer or one of the signers) know the "context" of the ciphertexts c_1 and c_2, meaning that they are created with respect to which message m and which verification keys. Namely, we can define a label $L = m||y_S||y_C$ so that the prover is aware of the context of the ciphertext. That is also the reason we use the labels in our construction, though we do not explicitly disclose the labels in the above proposition. Detailed descriptions of how to use public key encryption scheme supporting the use of labels, are demonstrated in the next section.

Remark 2: Gentry et al. pointed out that, all the statements involving zero-knowledge proofs can be expressed as NP statements (and have a short witness). Therefore, it's feasible, in theory, to instantiate the above scheme in polynomial time for any suitably secure encryption scheme, commitment scheme, and signature scheme.

Remark 3: In our DCSSign algorithm, π_0 shall be a NIZK proof that could be validated publicly. However, sometimes to conform to reality, one may also retain the interaction between the signature issuer, i.e., signer S, and the signature recipient, by replacing π_0 with a interactive ZK protocol π_0' that proves the equality of the encrypted message in c_1 and c_2.

Remark 4: We do not include the checks on σ' and π_0 in $Disavow$ protocols, since σ' is a publicly verifiable signature on ψ, and π_0 is a NIZK proof in general. Consequently, the verifier can verify these two elements by himself before performing the $Disavow$ protocol.

Remark 5: It is notable that the DCS σ' signs on the commitment ψ rather than the rare message m. The statistically hiding property of the underlying commitment scheme ensures the inability of discovering the relations between the messages and the DCSs. However, Our construction allows that the DCS σ' may convince any verifier that the signer indeed signed *some* message m, as the ordinary signature pair (ψ, σ) is publicly verifiable. Because (ψ, σ) is always a valid signature pair if the DCS σ' is valid, the distribution on the signature space corresponding to the random variable σ' for any fixed key and varying messages is *not* computationally indistinguishable from a uniform distribution. Based on this observation, our construction does not meet the requirement of the addition property in Galbraith and Mao's proposal [33] (see in section 2, Property A). Accordingly, our construction satisfies (weak) invisibility, while is not covered by the security of invisibility defined by Galbraith and Mao in [33], and a formal analysis will be provided in the next section. We emphasis that invisibility in [33] is a little stronger than the "weak invisibility" in Def 3.6, section 2, Chapter 3 (originally introduced by Camenisch and Michels [18]. A detailed discussion and proofs for identifying the relations of these two definitions can be seen in section 3 in [33]). In fact, for many real life applications, we think (weak) invisibility is enough to meet the security requirements.

6.4 Security Analysis

Let $DS = (DS\_Gen, Sig, Ver)$ be any signature scheme secure against chosen message attack, and let $PKE = (PKE\_Gen, Enc, Dec)$ be any IND-CCA2 secure encryption scheme and $Comm = (Com, DeCom)$ be any statistically hiding and computationally binding commitment scheme with perfect zero-knowledge proofs of knowledge for committed values secure against cheating verifiers. We use $\mathcal{DCS}$ to denote a DCS scheme with full verification following the construction in section 1. According to the security model given in section 4, Chapter 5, we demonstrate the security of the above generic construction in the following two theorems. We remark that, theorem 2 is about the invisibility analysis in the standard model, and we follow an alternative definition given in Def 3.6, section 2, Chapter 3 (and also by Camenisch and Michels [18]). In fact, Galbraith and Mao [32] have proved that these two types of invisibility are actually equivalent in the standard model of computation.

Theorem 6.1. If the underlying signature scheme $DS = (DS\_Gen, Sig, Ver)$ is existentially unforgeable against chosen message attacks and the commitment scheme is computationally binding, the scheme $\mathcal{DCS}$ is existentially unforgeable against chosen message attacks.

Proof: The main idea is, after the training phase, the adversary A has a non-negligible probability of successfully outputting a fresh DCS (m, σ') for which $Verify(m, Extract(m, \sigma', x_C, vk_S), vk_S) = Accept$. From A, we can construct an algorithm B that is able to either construct an existential forgery of the underlying signature scheme, or violate the binding property of the commitment scheme. We denote C as an instance of the underlying signature scheme. In the simulation, C will create related key-pairs and answer all B 's signing queries except the challenging query. Now we describe the simulations as follows.

Initially, C generates the signing-verification key-pair (sk_S, vk_S) via the algorithm $KGen$. B runs the algorithm PKE_Gen and generates the signer and the confirmer's encryption key-pairs, i.e., (x_S, y_S) and (x_C, y_C) respectively. B sends x_S and x_C to A.

B simulates all needed DCS oracles for A as follows:

$DCSSign$ query: For any queried message m, B picks a randomness r, computes a commitment $\psi = Com(m, r)$ with the randomness r. Then B uses its own oracle access to C, to obtain a signature on ψ, i.e., $\sigma = Sig(sk_S, (\psi, vk_S))$. B also generates appropriate ciphertexts c_1 and c_2 that encrypts r, and produces a ZK proof of knowledge π_0 shows c_1 and c_2 encrypts the same randomness r. The output DCS is $\sigma' = (\psi, \sigma, c_1, c_2, \pi_0)$. Meanwhile, B maintains a DCS-List when generating any new DCS.

For $Confirm_{(S,V)}$, $Disavow_{(S,V)}$, $Confirm_{(C,V)}$, $Disavow_{(C,V)}$ and $Extract$ queries, A can simulate and achieve correct results using x_S or x_C.

Suppose that A outputs a pair $(\hat{m}, \hat{\sigma'})$ where $\hat{\sigma'} = (\hat{\psi}, \hat{\sigma}, \hat{c_1}, \hat{c_2}, \hat{\pi_0})$, and $\hat{m}$ is a fresh message. B uses x_C to extract the underlying basic signature of $(\hat{m}, \hat{\sigma'})$, say $(\hat{\sigma}, \hat{r})$. B checks its DCS-List , if σ is found for some m, and $\sigma = \hat{\sigma}$, this suggests B must have responded to A's $DCSSign$ query on m by generating a randomness r, for which $\psi = Com(\hat{m}, \hat{r}) = Com(m, r)$. Since $\hat{m} \neq m$, this violates the binding property of the commitment scheme. Otherwise, B outputs σ as an existential forgery on message (ψ, vk_S). □

Theorem 6.2. If the underlying encryption scheme PKE is IND-CCA2 secure, then the scheme $\mathcal{DCS}$ is invisible w.r.t. the definition of Def 3.6 in section 2, Chapter 3.

Proof: The main idea is, in the invisibility game, when receiving a challenge DCS $\sigma' = (\psi, \sigma, c_1, c_2, \pi_0)$ on two messages m_0 and m_1, the adversary is to distinguish the validities of the DCSs adaptively. Since the commitment scheme is computationally binding, which means ψ is a valid commitment on either m_0 or m_1. But due to the perfect hiding property, the DCS adversary cannot succeed via ψ.

In the following simulation process, we use A as a subroutine to construct a PPT adversary B which will successfully break the underlying encryption scheme. We denote C as an instance of the underlying encryption scheme. In the simulation, C will create related key-pairs and answer all B's decryption queries except the challenging query.

Initially, two key-pairs (x_S, y_S) and (x_C, y_C) generated via PKE_Gen by C , are set as the public encryption keys of the signer and the confirmer. B runs $KGen$ for generating a signing-verification key-pair (sk_S, vk_S). We denote ϵ_A and ϵ_B as A and B's advantage for breaking the scheme $\mathcal{DCS}$ and the underlying encryption scheme

PKE respectively.

B simulates all DCS oracles for A as follows:

$DCSSign$ query: For any queried message m, B picks a randomness r, computes a commitment $\psi = Com(m, r)$ with the randomness r, and creates $\sigma = Sig(sk_S, \psi)$; B also computes two encryptions, i.e., $c_1 = Enc(y_S, r)$, $c_2 = Enc(y_C, r)$, and produces a ZK proof of knowledge π_0 shows c_1 and c_2 encrypts the same randomness r.. Eventually B returns the DCS $\sigma' = (\psi, \sigma, c_1, c_2, \pi_0)$ to A. In addition, B also maintains a (m, r)-List . Whenever a new DCS σ' created and returned to A, the corresponding message-randomness pair (m, r) is added to (m, r)-List .

$Confirm_{(S,V)}$ query: For any queried message-DCS pair (m, σ'), where $\sigma' = (\psi, \sigma, c_1, c_2, \pi_0)$, B looks up the (m, r)-List with input m. If a saved $(m, \hat{r})$ is found, B checks the DCS's validity with the value $\hat{r}$. Namely, B checks the equation $\psi = Com(m, \hat{r})$. B terminates if the check fails. Otherwise, B performs a zero-knowledge proof of knowledge of the value $\hat{r}$ such that $\psi = Com(m, \hat{r})$, $c_1 = Enc(y_S, \hat{r})$.

$Disavow_{(S,V)}$ query: For any queried message-DCS pair (m, σ'), where $\sigma' = (\psi, \sigma, c_1, c_2, \pi_0)$, B first enquires C's decryption oracle with input c_1. Once B receives the decrypted value $\hat{r}$, it checks if both c_1 and c_2 are valid encryptions of some randomness. If not, it performs a ZK proof of knowledge such that either c_1 or c_2 is not well-formed. Otherwise, if $\psi \neq Com(m, \hat{r})$, B provides a ZK proof of knowledge that there is a value $\hat{r}$ such that $\psi \neq Com(m, \hat{r})$ and $c_1 = Enc(y_S, \hat{r})$.

$Confirm_{(C,V)}$ query: For any queried message-DCS pair (m, σ'), where $\sigma' = (\psi, \sigma, c_1, , c_2, \pi_0)$, B looks up the (m, r)-List with input m. If a saved $(m, \hat{r})$ is found, B checks the DCS's validity with the value $\hat{r}$. Namely, B checks the equation $\psi = Com(m, \hat{r})$, if it fails, B terminates. Otherwise, B performs a zero-knowledge proof of knowledge of the value $\hat{r}$ such that $\psi = Com(m, \hat{r})$, and $c_2 = Enc(y_C, \hat{r})$.

$Disavow_{(S,V)}$ query: For any queried message-DCS pair (m, σ'), where $\sigma' =$

$(\psi, \sigma, c_1, c_2, \pi_0)$, B first enquires C's decryption oracle with input c_2. Once B receives the decrypted value $\hat{r}$, it checks if both c_1 and c_2 are valid encryption of some randomness. If not, it performs a ZK proof of knowledge such that either c_1 or c_2 is not well-formed. Otherwise, if $\psi \neq Com(m, \hat{r})$, B provides a ZK proof of knowledge that there is a value $\hat{r}$ such that $\psi \neq Com(m, \hat{r})$ and $c_2 = Enc(y_C, \hat{r})$.

$Extract$ query: For any queried message-DCS pair (m, σ'), where $\sigma' = (\psi, \sigma, c_1, c_2)$, B looks up the (m, r)-List with input m. If a saved $(m, \hat{r})$ is found, B checks the DCS's validity with the value $\hat{r}$. Otherwise, B terminates. Namely, B checks three equation $\psi = Com(m, \hat{r})$, if it fails, B terminates. Otherwise, B outputs $(\sigma, \hat{r})$.

In the challenge phase, After q adaptive oracle queries by A, it presents two messages m_0 and m_1, and sends them to B.

B picks two randomnesses respectively, i.e., r_0 for committing m_0, and r_1 for committing m_1. B computes a "possible" commitment by flipping a fair coin b', i.e., $\psi_{b'} = Com(m_{b'}, r_{b'})$, and further computes $\sigma_{b'} = Sig(sk_S, \psi_{b'})$. B uses r_0 and r_1 as the equal length messages in its "find stage", and sends them to C. By flipping a fair coin b, C returns a tuple (c_{1-b}, c_{2-b}), where $c_{1-b} = Enc(y_S, r_b)$ and $c_{2-b} = Enc(y_C, r_b)$, together with a ZK proof of knowledge π_{0-b} shows c_{1-b} and c_{2-b} encrypts the same message. B returns the challenge as $\sigma_b = (\psi_{b'}, \sigma_{b'}, c_{1-b}, c_{2-b}, \pi_{0-b})$.

At the end of the simulation, A outputs its guess on b', denote as b'', B straightly uses b'' as its own guess on b and outputs b''.

In fact, ϵ_A and ϵ_B are A and B's advantage for guessing each coin value respectively. The "failure" cases are when $b' \neq b$, that is the challenge DCS $\sigma_b = (\psi_{b'}, \sigma_{b'}, c_{1-b}, c_{2-b})$ is invalid, which means B's responses to A's queries deviate from the real oracles' responses. In that case, B's advantage is no better than a random guessing. Note in B's view, flipping the coin b' is independent from flipping the coin b. Thus the "failure" probability is exactly $1/2$.

Hence we have $\frac{1}{2}+\epsilon_B = \frac{1}{2}\times(\frac{1}{2}+\epsilon_A)+\frac{1}{2}\times\frac{1}{2}$, i.e., $2\cdot\epsilon_B = \epsilon_A$. If ϵ_B is non-negligible, ϵ_A is non-negligible as desired. $\square$

6.5 Implementation and Evaluation of the DCS Scheme

We show how to efficiently instantiate the above scheme. To fulfill the building blocks, we select the scheme by Camenisch and Shoup which was discussed in sub-section 2.2, as the underlying encryption scheme PKE. The commitment scheme will be a Pedersen-type commitment scheme as described in sub-section 2.1. We choose *BLS* short signature [14] as the underlying digital signature scheme. The reason we do not choose the very interesting Boneh-Boyen signature [8, 9] (BB signature) which also has short signature size, is that the underlying computational problem seems a little ornate and contrived. In fact, Boneh and Boyen give a reductionist security argument showing that a chosen message attacker cannot forge a signature provided that the so-called Strong Diffie-Hellman (SDH) problem is hard. This problem is parametrised by an integer l (which is a bound on the number of signature queries the attacker is allowed to make) and is denoted l-SDH. In particular, recently Jao and Yoshida[50] showed by using the techniques in [24], one can forge signatures in roughly $p^{2/5}$ operations (with roughly $p^{1/5}$signature queries) under certain conditions., where p represents the order of the underlying cyclic groups.

6.5.1 The instantiation

Considering a symmetric bilinear map $e : G \times G \to G_t$, where G is a multiplicative cyclic group of prime order q and g is a generator of G. $H : \{0,1\}^* \to G$ is a full-domain hash function. H' is a collision-resistant hash function. Let $n = p'q'$ be the product of two Sophie-Germain primes p' and q' (i.e., there exist two primes p_0 and q_0

such that $p' = 2p_0 + 1$ and $q' = 2q_0 + 1$). Similar to the subsection 2.2, we define a function $abs(\cdot)$: $\mathbb{Z}_{n^2} \to \mathbb{Z}_{n^2}$ as $abs(a)$ = a if $0 \leq a \leq n^2/2$, or $abs(a) = n^2 - a$ mod n^2 if $n^2/2 < a \leq n^2$. To obtain a verifiable encryption scheme from the CS-Paillier cryptosystem, we assume there is an additional composite modulus $n_2 = p_2q_2$, where $p_2 = 2p_2' + 1$ and $q_2 = 2q_2' + 1$ are two safe primes, along with elements $g_2, h_2 \in \mathbb{Z}^*_{n_2}$ of order $p_2'q_2'$.

In addition, we select a third group Γ of prime order ρ, with two generators δ and γ , and the discrete logarithm problem is assumed to be hard in Γ. In our scheme, a message m shall be committed by $Com(m, r) = \delta^m\gamma^r$, where $r \in_R [\rho]$. We require $n_2 \neq n$, $\rho =\mid \Gamma \mid < n \cdot 2^{-k-k'-3}$, and $2^k < min\{p', q', p_2', q_2'\}$ for two further security parameters k and k'. Actually, $\{0, 1\}^k$ defines the "challenge space" of the verifier, while k' controls the quality of the ZK property. In addition, it is required that the prover does not know the factorization of n_2. For simplicity, we suppose that $(n_2, g_2, h_2, \Gamma, \delta, \gamma)$ are generated by a trusted party and viewed as a common reference string. Note implementation details of the ZK protocols in $DCSSign$, $Confirm_{(S,V)}$, $Disavow_{(S,V)}$, $Confirm_{(C,V)}$, and $Disavow_{(C,V)}$ can be seen in Appendix A.

$KGen$: The signer S picks a random value $x \in_R \mathbb{Z}^*_q$ as its private key of the signing key-pair, and computes g^x as its public key of the signing key-pair. The signer generates its encryption/decryption keys as follows: it picks x_{S_1}, x_{S_2}, $x_{S_3} \in_R [n^2/4]$, where$x_S = (n, x_{S_1}, x_{S_2}, x_{S_3})$ constitute the decryption key. The signer computes $h = 1 + n$, picks a random value $g_S' \in \mathbb{Z}^*_{n^2}$, and computes $g_S = g_S'^{2n}$, $y_{S_1} = g^{x_{S_1}}$, $y_{S_2} = g^{x_{S_2}}$, and $y_{S_3} = g^{x_{S_3}}$ where $y_S = (h, n, g_S, y_{S_1}, y_{S_2}, y_{S_3})$ constitute the encryption key. The confirmer computes its encryption/decryption key similarly, and denotes as $x_C = (h, n, x_{C_1}, x_{C_2}, x_{C_3})$, and $y_C = (h, n, g_C, y_{C_1}, y_{C_2}, y_{C_3})$.

$Sign$: To issue an ordinary signature on a message m, the signer S computes a statistically hiding and computationally binding commitment$\psi = \delta^m\gamma^r$ with the randomness r, and creates $\sigma = H(\psi)^x$; The basic signature is $\sigma^* = (\sigma, r)$.

$Verify$: On input an basic DCS signature $\sigma^* = (\sigma, r)$ for a message m, this algorithm returns $Accept$ if $e(\sigma, g) = e(H(\psi), g^x)$, where $\psi = \delta^m \gamma^r$.

$DCSSign$: To issue a DCS on a message m, the signer S computes a commitment $\psi = \delta^m \gamma^r$ with the randomness r, and creates $\sigma = H(\psi)^x$; In addition, S also computes two encryptions, i.e., $c_1 = (u_1, e_1, v_1) = Enc(y_S, m)$ and $c_2 = (u_2, e_2, v_2) = Enc(y_C, m)$. In particular, to compute c_1, the signer should choose a random value $tem \in_R [n/4]$, and computes $u_1 = g_S^{tem}$, $e_1 = y_{S_1}^{tem} h^r$, $v_1 = abs((y_{S_2} y_{S_3}^{H(u_1, e_1, L_1)})^{tem})$ with a label $L_1 = m \parallel y_S$. Similarly, to compute c_2, the signer should choose another randomness $tem' \in_R [n/4]$, and computes $u_2 = g_C^{tem'}$, $e_2 = y_{C_1}^{tem'} h^r$, $v_2 = abs((y_{C_2} y_{C_3}^{H(u_2, e_2, L_2)})^{tem'})$ with the label $L_2 = m \parallel y_C$. S also provides a zero knowledge proof π_0 shows that both c_1 and c_2 encrypt the same randomness that is used to compute the commitment ψ. Finally, the output designated confirmer signature is $\sigma' = (\psi, \sigma, c_1, c_2, \pi_0)$ on the message m,

$$\pi_0 = PK\{(tem, tem', r, s) : e_1 = y_{S_1}^{tem} h^r \wedge e_2 = y_{C_1}^{tem'} h^r\}$$

Note in the CS-Paillier encryption scheme, the randomness r is hidden in the value e which is actually a "commitment", to prove the equality of the randomness encrypted by c_1 and c_2, is equivalent to prove that two commitments, e_1 and e_2, hide the same secret r. we use the NIZK protocol proposed by Boudot [15] which in fact is derived from proofs of equality of two discrete logarithms combined with a proof of knowledge of a discrete logarithm modulo n.

$Confirm_{(S,V)}$: On receiving a message-DCS pair $(m, \sigma') = (m, (\psi, \sigma, c_1, c_2, \pi_0))$, signer S first checks if $e(\sigma, g) \equiv e(H(\psi), g^x)$. S aborts if this check fails. Otherwise, S decrypts c_1 to get a value r', and then checks if $\psi \equiv \delta^m \gamma^{r'}$. Specifically, to decrypt c_1, the signer first checks if $abs(v_1) \equiv v_1$ and $u_1^{2(x_{S_2} + H(u_1, e_1, L_1) \cdot x_{S_3})} \equiv v_1^2$; If this does not hold, then output $\perp$ and halt, otherwise the signer computes $\hat{r} = (e_1/u_1^{x_{S_1}})^{2k}$ where $k = 2^{-1} \bmod n$. If $\hat{r}$ is of form h^d for some $d \in [n]$ (i.e., $\hat{r} - 1$ is divisible by n), then output $r' = (\hat{r} - 1)/n \in [n]$; otherwise, output $\perp$. If any of this procedure fails, S executes $Disavow_{(S,V)}$ protocol. Otherwise, S runs a ZK proof of knowledge of values $(x_{S_1}, x_{S_2}, x_{S_3}, r, s)$ with the verifier, where $s \in [n_2/4]$:

$$\begin{aligned}\pi_1 = PK\{(x_{S_1}, x_{S_2}, x_{S_3}, r, s) : y_{S_1} = g^{x_{S_1}} \wedge y_{S_2} = g^{x_{S_2}} \wedge y_{S_3} = g^{x_{S_3}} \\ \wedge e_1^2 = u_1^{2x_{S_1}} h^{2r} \wedge v_1^2 = u_1^{2x_{S_2}} u_1^{2H(u_1, e_1, L_1)x_{S_3}} \\ \wedge \psi\delta^{-m} = \gamma^r \wedge l = g_2^r h_2^s \wedge -n/2 < r < n/2\}\end{aligned}$$

$Disavow_{(S,V)}$: On receiving a message-DCS pair $(m, \sigma') = (m, (\psi, \sigma, c_1, c_2, \pi_0))$, signer S checks if c_1 is a valid encryption of some r with respect to label L_1. If not, it performs a ZK proof of knowledge such that c_1 is not well-formed. Otherwise, S computes $r' = Dec(c_1, x_S)$. If $\psi \neq \delta^m \gamma^{r'}$, S provides a ZK proof of knowledge r' such that $\psi \neq \delta^m \gamma^r$ and $c_1 = Enc(y_S, r')$. In general, S will provides a ZK proof for the following statement:

[c_1 is invalid w.r.t. $L_1 = m \parallel y_S$] OR [$\exists$ r' s.t. $r' = Dec(c_1, x_S)$ AND $\psi \neq \delta^m \gamma^{r'}$]

$Confirm_{(C,V)}$: On receiving a message-DCS pair $(m, \sigma') = (m, (\psi, \sigma, c_1, c_2, \pi_0))$, the confirmer C first checks whether σ is signed on ψ by checking if $e(\sigma, g) = e(H(\psi), g^x)$. C aborts if this check fails. Otherwise, C decrypts c_2 to get a value r by using x_C, and then checks if $\psi \equiv \delta^m \gamma^r$. If any of this procedure fails, C executes $Disavow_{(C,V)}$ protocol. Otherwise, C runs the interactive protocols π_2 with the verifier. Specially, π_2 is a ZK proof of knowledge of values $(x_{C_1}, x_{C_2}, x_{C_3}, r, s)$, where $s \in [n_2/4]$.

$$\begin{aligned}\pi_2 = PK\{(x_{C_1}, x_{C_2}, x_{C_3}, r, s) : y_{C_1} = g^{x_{C_1}} \wedge y_{C_2} = g^{x_{C2}} \wedge y_{C_3} = g^{x_{S_C}} \\ \wedge e_2^2 = u_2^{2x_{C1}} h^{2r} \wedge v_2^2 = u_2^{2x_{C_2}} u_2^{2H(u_2,e_2,L_2)x_{C_3}} \\ \wedge \psi\delta^{-m} = \gamma^r \wedge l = g_2^r h_2^s \wedge -n/2 < r < n/2\}\end{aligned}$$

$Disavow_{(C,V)}$:On receiving a message-DCS pair $(m, \sigma') = (m, (\psi, \sigma, c_1, c_2, \pi_0))$, confirmer C checks if c_2 is a valid encryption of some r with respect to label L_2. If not, it performs a ZK proof of knowledge such c_2 is not well-formed. Otherwise, C computes $r' = Dec(c_2, x_C)$. If $\psi \neq \delta^m\gamma^r$, C provides a ZK proof of knowledge that there is a value r' such that $\psi \neq \delta^m\gamma^r$ and $c_2 = Enc(y_C, r')$. In general, C will provides a ZK proof for the following statement:

[c_2 is invalid w.r.t. $L_2 = m \parallel y_C$] OR [$\exists\, r'$ s.t. $r' = Dec(c_2, x_C)$ AND $\psi \neq \delta^m\gamma^{r'}$]

$Extract$: On input m and $\sigma' = (\psi, \sigma, c_1, c_2)$, the confirmer C first checks whether σ is signed on ψ, then decrypts c_2 to get a value r by using x_C, and checks if $\psi = \delta^m\gamma^r$. If any of the procedure fails, C outputs $\perp$. Otherwise, C outputs the basic signature $\sigma^* = (\sigma, r)$.

we remark that the above DCS scheme is actually proposed for a message digest $m \in [\rho]$. That means, to sign a message M with arbitrary length, we should first compress M to a short digest m using a collision-resistant hash function.

6.5.2 Implementation Details

We describe how to implement π_0, π_1, π_2, and $Disaow$ protocols as follows.

π_0 in $DCSSign$ Algorithm. The details of this ZK protocol, which should be transformed into a CZK protocol in our DCS scheme, are illuminated as follows.

$$\pi_0 = PK\{(tem, tem', r, s) : e_1 = y_{S_1}^{tem} h^r \wedge e_2 = y_{C_1}^{tem'} h^r\}$$

1. The signer S randomly picks $\eta_1, \eta_2 \in_R [-n2^{k+k'-2}, n2^{k+k'-2}]$, and $\omega \in_R [-\rho 2^{k+k'}, \rho 2^{k+k'}]$, and computes $W_1 = y_{S_1}^{\eta_1} h^{\omega}$, $W_2 = y_{C_1}^{\eta_2} h^{\omega}$.

2. S computes $c = \mathcal{H}(W_1 \parallel W_2)$ where $\mathcal{H}$ is a hash function that outputs $2k$-bit strings.

3. S computes $D = \omega + c \cdot r$, $D_1 = \eta_1 + c \cdot tem$, $D_2 = \eta_2 + c \cdot tem'$, and sends (c, D, D_1, D_2) to the verifier. Note D, D_1, D_2 are all computed in $\mathbb{Z}$.

4. The verifier checks whether $c = \mathcal{H}(y_{S_1}^{D_1} h^D e_1^{-c} \parallel y_{C_1}^{D_2} h^D e_2^{-c})$.

π_1 **in** $Confirm_{(S,V)}$ **Protocol.** At the end of $Confirm_{(S,V)}$ protocol, S shall run a ZK proof of knowledge of values $(x_{S_1}, x_{S_2}, x_{S_3}, r, s)$ with the verifier, where $s \in [n_2/4]$:

$$\begin{aligned}\pi_1 = PK\{(x_{S_1}, x_{S_2}, x_{S_3}, r, s) : y_{S_1} = g^{x_{S_1}} \wedge y_{S_2} = g^{x_{S_2}} \wedge y_{S_3} = g^{x_{S_3}} \\ \wedge e_1^2 = u_1^{2x_{S_1}} h^{2r} \wedge v_1^2 = u_1^{2x_{S_2}} u_1^{2H(u_1,e_1,L_1)x_{S_3}} \\ \wedge \psi\delta^{-m} = \gamma^r \wedge \mathfrak{l} = \mathfrak{g}^r \mathfrak{h}^s \wedge -n/2 < r < n/2\}\end{aligned}$$

Let $\alpha = \psi\delta^{-m}$. For three integers a, b and c with $c > 0$, $a = b$ rem c denotes the balanced remainder of b modulo c. Namely, $a = b + kc \in [-c/2, c/2)$ for some integer k. The details of this ZK protocol, which should be transformed into a CZK protocol in DCS setting, are described as follows.

S first selects a random $s \in_R [n_2/4]$, and computes $l = g_2^r h_2^s$. Then S randomly selects $x'_{S_1}, x'_{S_2}, x'_{S_3} \in_R [-n2^{k+k'-2}, n2^{k+k'-2}]$, $r' \in_R [-\rho 2^{k+k'}, \rho 2^{k+k'}]$, and $s' \in_R [-n2^{k+k'-2}, n2^{k+k'-2}]$, and computes $(y'_{S_1}, y'_{S_2}, y'_{S_3}, e'_1, v'_1, \alpha', l')$ as follows:

$$y'_{S_1} = g^{x'_{S_1}}, y'_{S_2} = g^{x'_{S_2}}, y'_{S_3} = g^{x'_{S_3}}$$

$$e'_1 = u_1^{x'_{S_1}} h^{r'}, v'_1 = u_1^{x'_{S_2}} u_1^{H(u_1,e_1,L_1)x'_{S_3}}$$

$$\alpha' = \gamma^{r'}, l' = g_2^{r'} h_2^{s'}$$

After that, S sends $(l, y'_{S_1}, y'_{S_2}, y'_{S_3}, e'_1, v'_1, \alpha', l')$ to the verifier.

The verifier selects a random challenge $c \in_R \{0,1\}^k$ and sends c to the signer S.

S responds with $(\tilde{x}_{S_1}, \tilde{x}_{S_2}, \tilde{x}_{S_3}, \tilde{r}, \tilde{s})$ by computing

$$\tilde{x}_{S_1} = x'_{S_1} - cx_{S_1}, \tilde{x}_{S_2} = x'_{S_2} - cx_{S_2}, \tilde{x}_{S_3} = x'_{S_3} - cx_{S_3}$$

$$\tilde{r} = r' - cr, \tilde{s} = s' - cs$$

The verifier outputs $Accept$ or $\perp$, according to whether all the following equations/conditions hold or not respectively:

$$y'_{S_1} = y^c_{S_1} g^{\tilde{x}_{S_1}}, y'_{S_2} = y^c_{S_2} g^{\tilde{x}_{S_2}}, y'_{S_3} = y^c_{S_3} g^{\tilde{x}_{S_3}}, e'^2_1 = e^{2c}_1 u_1^{2\tilde{x}_{S_1}} h^{2\tilde{r}},$$

$$v'^2_1 = v^{2c}_1 u_1^{2\tilde{x}_{S_2}} u_1^{2H(u_1,e_1,L_1)\tilde{x}_{S_3}}, \alpha' = \alpha^c \gamma^{\tilde{r}}, l' = l^c g_2^{\tilde{r}} h_2^{\tilde{s}}, -n/4 < \tilde{r} < n/4.$$

The $Disavow_{(S,V)}$ **Protocol.** The signer needs to provide an ZK proof showing that: [c_1 is invalid w.r.t. $L_1 = m \parallel y_S$] OR [$\exists\ r'$ s.t. $r' = Dec(c_1, x_S)$ AND $\psi \neq \delta^m \gamma^{r'}$] holds. According to the analysis given in section 7.3 [19] , this means that the confirmer needs to prove that at least one of the following equations does not hold: $u_1^{2(x_{S_2}+H(u_1,e_1,L_1)x_{S_3})} = 1$, $(e_1/u_1^{x_{S_1}})^{2n} = 1$, $\alpha = \gamma^{[\log_{h^2}(e_1/u_1^{x_{S_1}})^2\, rem\, n]}$

where label $L_1 = m \parallel y_S$ and $\alpha = \psi\delta^{-m}$. We can directly adopt the ZK protocol specified in section 7.3 of the full paper [19], which is the same as our $Disavow_{(P,V)}$ protocol and is used to prove verifiable decryption of a discrete logarithm. Note one should replace the notations, i.e., change the variables $x_1, x_2, x_3, u, e, v, L, \delta$ into $x_{S_1}, x_{S_2}, x_{S_3}, u_1, e_1, v_1, L_1, \alpha$ respectively.

π_2 in $Confirm_{(C,V)}$ Protocol. Analogically, one could derive the detailed steps of running π_2 by replacing the related notations, i.e., changing

$x_{S_1}, x_{S_2}, x_{S_3}, u_1, e_1, v_1, L_1, x'_{S_1}, x'_{S_2}, x'_{S_3}, y'_{S_1}, y'_{S_2}, y'_{S_3}, e'_1, v'_1, \tilde{x}_{S_1}, \tilde{x}_{S_2}, \tilde{x}_{S_3}$
in the procedure of running π_1 into
$x_{C_1}, x_{C_2}, x_{C_3}, u_2, e_2, v_2, L_2, x'_{C_1}, x'_{C_2}, x'_{C_3}, y'_{C_1}, y'_{C_2}, y'_{C_3}, e'_2, v'_2, \tilde{x}_{C_1}, \tilde{x}_{C_2}, \tilde{x}_{C_3},$
respectively.

The $Disavow_{(C,V)}$ Protocol. The confirmer needs to provide an ZK proof showing that: [c_2 is invalid w.r.t. $L_2 = m \parallel y_C$] OR [$\exists\ r'$ s.t. $r' = Dec(c_2, x_C)$ AND $\psi \neq \delta^m \gamma^{r'}$] holds. Let $\alpha = \psi\delta^{-m}$. Similar to $Disavow_{(C,V)}$protocol above, one could directly use the ZK protocol in section 7.3 of [19], by replacing the notations of $x_1, x_2, x_3, u, e, v, L, \delta$ with $x_{C_1}, x_{C_2}, x_{C_3}, u_2, e_2, v_2, L_2, \alpha$, respectively.

6.5.3 Efficiency Analysis

We give a brief comparison between our instantiation in subsection 5.1 and two schemes, i.e., the GMR scheme [38], and the WBWB scheme introduced by Wang et al. in section 6 in [78]. Similar to the comparison made in [38], we also compare these DCS schemes in three categories, i.e., whether the scheme relies on the random oracle model, which kinds of the underlying ordinary signatures are adopted, and how about the computational efficiency. We also list the estimated numbers of exponentiations needed in each interactive protocol. Recall λ is a security parameter, let ex_G and ex_{GT} denote the time for computing an exponentiation in cyclic group G and G_t respectively. In fact, no exponentiation in group G_t is required to execute in our practical implementation. Not surprisingly, from Table 6.1, our scheme has the similar efficiency with the original GMR scheme, since our construction is a straight inheritance from the GMR scheme. And without using random oracles, we achieve $Confirm$ and

	GMR [38]	WBWB [78]	Our Scheme
Random Oracle	No	Yes	No
Underlying Signatures	Any	Any	Any
$Confirm_{(C,V)}$	25 ex_G	15 ex_G	25 ex_G
$Disavow_{(C,V)}$	60 ex_G	16 ex_G	60 ex_G

Table 6.1: Comparison of The Generic DCS-FV with Two Similar Schemes

$Disavow$ protocols with an acceptable efficiency.

6.6 Summary

We have shown that by extending the constructions of the improved GMR scheme in [78], a new generic DCS scheme could be derived which supports the signer's disavowability. Since both the signer and the confirmer can do confirmation and disavowal on any alleged signature, our proposed scheme is so-called "DCS with full verification". Also with the security analysis in section 4, we can use any digital signature scheme that is existentially unforgeable against chosen message attacks (EUF-CMA) and any commitment scheme that is computationally binding, together with a public encryption scheme that is IND-CCA2 secure, to build a specific secure instantiation of our DCS cryptosystem.

we explicitly specify how to use labels in the DCS scheme by implementing an instantiation which uses Pedersen commitments and CS-Paillier cryptosystem. For our $Confirm_{(P,V)}$ protocol ($P = \{S, C\}$), it requires proving the verifiable encryption of a discrete logarithm as described in subsection 5.2 of [19], and our $Disavow_{(P,V)}$ protocol requires proving the verifiable decryption of a discrete logarithm as described in subsection 7.3 of [19]. Meanwhile, to make the underlying zero-knowledge proofs efficient, we should use the Gennaro's approach [36] or CDM techniques [25] to transform SHVZK protocols to CZK protocols.

Going further, we may look for commitment schemes and efficient protocols based on different assumptions. For example, can we find some new technique to obtain an even more efficient instantiation based on bilinear mappings?

Chapter 7

Conclusions and Future Works

In this thesis, we have mainly investigated how to design and analyse designated confirmer signature schemes, as well as clarifying the relations of different security notions. By investigating the existing DCS schemes, we discovered that the security notion called “invisibility” is an essential property of DCS schemes, and some adaptive attacks that break invisibility, have been identified in different schemes. We explored previous DCS models, and found several security notions including invisibility are not very clear which bring confusions to future studies of DCS schemes. To address this issue, we made further analysis on related security notions in a more thorough way. More specifically, we reconciled the DCS model by comparing four properties, i.e., unimpersonation, invisibility, non-transferability and transcript-simulatability, and we provided formal proofs about the implications/equivalences between these properties. A important result is, we found that transcript-simulatability in GMR model [38] is covered by the combination of invisibility and non-transferability in CM model [18]; on the other hand, transcript-simulatability actually implies invisibility, and a different type of non-transferability.

Apart from the contributions of DCS modeling aspect, this thesis also improves the design of DCS schemes by proposing new constructions. As far as we know, none of the previously published DCS cryptosystems support the signer's disavow-ability, that is, the signer cannot convince a verifier that an alleged designated confirmer signature is invalid. From this observation, we proposed a new efficient pairing based DCS scheme that both the signer and the designated confirmer can run the same protocols to confirm a valid DCS or disavow an invalid signature, which we called *DCS with unified verification*. To achieve this, we introduced a new computational assumption, called Decisional Co-efficient Linear (D-co-L) assumption, whose intractability in pairing settings was analyzed in generic group model. The proposed scheme is composed by encrypting Boneh, Lynn and Shacham's pairing based short signatures [14] with signed ElGamal encryption [72]. The resulting solution is efficient in both aspects of computation and communication. Since the proposed scheme can be generalised by allowing the signer to run different protocols for confirming and disavowing signatures, which leads to a more practical version called *DCS with full verification,* for which we do not necessarily require the signer and the confirmer run the same confirm/disavow protocol.

We also introduced a new technique that gives the possibility of constructing generic DCS schemes with full verification. Our proposal is inspired by GMR scheme [38] and the improved GMR scheme in [78], where both use a commitment scheme as a "layer of indirection". The generic scheme is a straight inheritance of the GMR transformation, and it enjoys the similar benefits of the former, i.e., the proposed scheme gives rise to an efficient generic DCS construction without appealing to both random oracles and general zero-knowledge proofs. Furthermore, to avoid the re-used randomness by any adaptive adversary, we draw on the solution introduced by [78], i.e., let the underlying IND-CCA2 secure encryption scheme support the use of "labels". By implementing an instantiation which uses Pedersen commitments and CS-Paillier cryptosystem, we explicitly specify how to use labels in the DCS scheme.

Besides the above contributions, this work gives rise to some new open problems. Firstly, it is worthwhile to clarify the relations between the original version of non-transferability defined in Chapter 3 (also in [18]) and our proposed new definition as specified in Chapter 4, because if either an implication or an equivalence relation between the two types of non-transferability is found, the relations among transcript-simulatability, invisibility and non-transferability will be more complete and accurate. Secondly, we believe by finding some new techniques, one could construct a more efficient concrete DCS scheme with full verification (or even unified verification) in the standard model, comparing to the proposed scheme in Chapter 5. Furthermore, we could investigate the strong witness hiding proofs of knowledge approach of Goldwasser and Waisbard [45] with an eye towards weakening the assumptions required for an efficient instantiation.

Bibliography

[1] ASOKAN, N., SHOUP, V., AND WAIDNER, M. Optimistic fair exchange of digital signatures. *IEEE J.Sel. A. Commun. 18*, 4 (Sept. 2006), 593–610.

[2] BARRETO, P. S., GALBRAITH, S. D., HÉIGEARTAIGH, C. O., AND SCOTT, M. Efficient pairing computation on supersingular abelian varieties. *Des. Codes Cryptography 42*, 3 (Mar. 2007), 239–271.

[3] BELLARE, M. Practice-oriented provable security. In *Lectures on data security: modern cryptology in theory and practise* (1998), I. Damgård, Ed., vol. 1561 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin Germany, pp. 1–15.

[4] BELLARE, M., DESAI, A., POINTCHEVAL, D., AND ROGAWAY, P. Relations among notions of security for public-key encryption schemes. In *Advances in Cryptology – CRYPTO ' 98* (1998), H. Krawczyk, Ed., vol. 1462 of *Lecture Notes in Computer Science*, International Association for Cryptologic Research, Springer-Verlag, Berlin Germany, pp. 26–45.

[5] BELLARE, M., AND ROGAWAY, P. Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computer and communications security* (New York, NY, USA, 1993), CCS '93, ACM, pp. 62–73.

[6] BELLARE, M., AND ROGAWAY, P. Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computer and communications security* (New York, NY, USA, 1993), CCS '93, ACM, pp. 62–73.

[7] BELLARE, M., AND ROGAWAY, P. Optimal asymmetric encryption. In *Advances in Cryptology—EUROCRYPT 94* (9–12 May 1994), A. D. Santis, Ed., vol. 950 of *Lecture Notes in Computer Science*, Springer-Verlag, 1995, pp. 92–111.

[8] BONEH, D., AND BOYEN, X. Short signatures without random oracles. In *EUROCRYPT* (2004), C. Cachin and J. Camenisch, Eds., vol. 3027 of *Lecture Notes in Computer Science*, Springer, pp. 56–73.

[9] BONEH, D., AND BOYEN, X. Short signatures without random oracles and the sdh assumption in bilinear groups. *J. Cryptol. 21*, 2 (Feb. 2008), 149–177.

[10] BONEH, D., BOYEN, X., AND GOH, E.-J. Hierarchical identity based encryption with constant size ciphertext. In *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings* (2005), R. Cramer, Ed., vol. 3494 of *Lecture Notes in Computer Science*, Springer, pp. 440–456.

[11] BONEH, D., BOYEN, X., AND SHACHAM, H. Short group signatures. In *Advances in Cryptology — CRYPTO 2004: 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15–19, 2004. Proceedings* (pub-SV:adr, 2004), M. Franklin, Ed., vol. 3152 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 41–??

[12] BONEH, D., AND FRANKLIN, M. K. Identity-based encryption from the weil pairing. In *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology* (London, UK, UK, 2001), CRYPTO '01, Springer-Verlag, pp. 213–229.

[13] BONEH, D., GENTRY, C., LYNN, B., AND SHACHAM, H. Aggregate and verifiably encrypted signatures from bilinear maps. In *Proceedings of the 22nd international conference on Theory and applications of cryptographic techniques* (Berlin, Heidelberg, 2003), EUROCRYPT'03, Springer-Verlag, pp. 416–432.

[14] BONEH, D., LYNN, B., AND SHACHAM, H. Short signatures from the weil pairing. *J. Cryptol. 17*, 4 (Sept. 2004), 297–319.

[15] BOUDOT, F. Efficient proofs that a committed number lies in an interval. In *Proceedings of the 19th international conference on Theory and application of cryptographic techniques* (Berlin, Heidelberg, 2000), EUROCRYPT'00, Springer-Verlag, pp. 431–444.

[16] BOUDOT, F., AND TRAORÉ, J. Efficient publicly verifiable secret sharing schemes with fast or delayed recovery. In *Proceedings of the Second International Conference on Information and Communication Security* (London, UK, UK, 1999), ICICS '99, Springer-Verlag, pp. 87–102.

[17] BROWN, M., HANKERSON, D., LÓPEZ, J., AND MENEZES, A. Software implementation of the nist elliptic curves over prime fields. In *Proceedings of the 2001 Conference on Topics in Cryptology: The Cryptographer's Track at RSA* (London, UK, UK, 2001), CT-RSA 2001, Springer-Verlag, pp. 250–265.

[18] CAMENISCH, J., AND MICHELS, M. Confirmer signature schemes secure against adaptive adversaries. In *Proceedings of the 19th international confer-*

ence on Theory and application of cryptographic techniques (Berlin, Heidelberg, 2000), EUROCRYPT'00, Springer-Verlag, pp. 243–258.

[19] CAMENISCH, J., AND SHOUP, V. Practical verifiable encryption and decryption of discrete logarithms. *Lecture Notes in Computer Science 2729* (2003), 126–144.

[20] CANETTI, R., GOLDREICH, O., AND HALEVI, S. The random oracle methodology, revisited. *J. ACM 51*, 4 (July 2004), 557–594.

[21] CHAUM, D. Designated confirmer signatures. In *Advances in Cryptology—EUROCRYPT 94* (9–12 May 1994), A. D. Santis, Ed., vol. 950 of *Lecture Notes in Computer Science*, Springer-Verlag, 1995, pp. 86–91.

[22] CHAUM, D., AND PEDERSEN, T. P. Wallet databases with observers. In *Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology* (London, UK, UK, 1993), CRYPTO '92, Springer-Verlag, pp. 89–105.

[23] CHAUM, D., AND VAN ANTWERPEN, H. Undeniable signatures. In *Advances in Cryptology (CRYPTO '89)* (Berlin - Heidelberg - New York, Aug. 1990), Springer, pp. 212–217.

[24] CHEON, J. H. Security analysis of the strong diffie-hellman problem. In *Proceedings of the 24th annual international conference on The Theory and Applications of Cryptographic Techniques* (Berlin, Heidelberg, 2006), EUROCRYPT'06, Springer-Verlag, pp. 1–11.

[25] CRAMER, R., DAMGÅRD, I., AND MACKENZIE, P. D. Efficient zero-knowledge proofs of knowledge without intractability assumptions. In *Proceedings of the Third International Workshop on Practice and Theory in Public Key*

Cryptography: Public Key Cryptography (London, UK, UK, 2000), PKC '00, Springer-Verlag, pp. 354–372.

[26] CRAMER, R., AND SHOUP, V. Signature schemes based on the strong rsa assumption. *ACM Trans. Inf. Syst. Secur. 3*, 3 (Aug. 2000), 161–185.

[27] DAMGÅRD, I. Efficient concurrent zero-knowledge in the auxiliary string model. In *Proceedings of the 19th international conference on Theory and application of cryptographic techniques* (Berlin, Heidelberg, 2000), EUROCRYPT'00, Springer-Verlag, pp. 418–430.

[28] DIFFIE, W., AND HELLMAN, M. New directions in cryptography. *IEEE Trans. Inf. Theor. 22*, 6 (Sept. 2006), 644–654.

[29] DWORK, C., NAOR, M., AND SAHAI, A. Concurrent zero knowledge. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC-98)* (New York, May 23–26 1998), ACM Press, pp. 409–418.

[30] ELGAMAL, T. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inf. Theor. 31*, 4 (Sept. 2006), 469–472.

[31] FIAT, A., AND SHAMIR, A. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology – CRYPTO ' 86* (Santa Barbara, CA, USA, 1987), A. M. Odlyzko, Ed., vol. 263 of *Lecture Notes in Computer Science*, International Association for Cryptologic Research, Springer-Verlag, Berlin Germany, pp. 186–194.

[32] GALBRAITH, S. D., AND MAO, W. Invisibility and anonymity of undeniable and confirmer signatures. In *Proceedings of the 2003 RSA conference on The cryptographers' track* (Berlin, Heidelberg, 2003), CT-RSA'03, Springer-Verlag, pp. 80–97.

[33] GALBRAITH, S. D., AND MAO, W. Invisibility and anonymity of undeniable and confirmer signatures. In *Proceedings of the 2003 RSA conference on The cryptographers' track* (Berlin, Heidelberg, 2003), CT-RSA'03, Springer-Verlag, pp. 80–97.

[34] GALBRAITH, S. D., PATERSON, K. G., AND SMART, N. P. Pairings for cryptographers. *Discrete Appl. Math. 156*, 16 (Sept. 2008), 3113–3121.

[35] GALBRAITH, S. D., AND SMART, N. P. A cryptographic application of weil descent. In *IMA Int. Conf* (1999), M. Walker, Ed., vol. 1746 of *Lecture Notes in Computer Science*, Springer, pp. 191–200.

[36] GENNARO, R. Multi-trapdoor commitments and their applications to proofs of knowledge secure under concurrent man-in-the-middle attacks. In *Advances in Cryptology — CRYPTO 2004: 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15–19, 2004. Proceedings* (pub-SV:adr, 2004), M. Franklin, Ed., vol. 3152 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 220–236.

[37] GENNARO, R., HALEVI, S., AND RABIN, T. Secure hash-and-sign signatures without the random oracle. In *Proceedings of the 17th international conference on Theory and application of cryptographic techniques* (Berlin, Heidelberg, 1999), EUROCRYPT'99, Springer-Verlag, pp. 123–139.

[38] GENTRY, C., MOLNAR, D., AND RAMZAN, Z. Efficient designated confirmer signatures without random oracles or general zero-knowledge proofs. In *Proceedings of the 11th international conference on Theory and Application of Cryptology and Information Security* (Berlin, Heidelberg, 2005), ASIACRYPT'05, Springer-Verlag, pp. 662–681.

[39] GOLDREICH, O. *Foundations of Cryptography, Volume I Basic Tools*. Cambridge Univeristy Press, 2001.

[40] GOLDREICH, O., GOLDWASSER, S., AND MICALI, S. How to construct random functions. *J. ACM 33*, 4 (Aug. 1986), 792–807.

[41] GOLDREICH, O., MICALI, S., AND WIGDERSON, A. Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems. *J. ACM 38*, 3 (July 1991), 690–728.

[42] GOLDWASSER, S., AND MICALI, S. Probabilistic encryption. *Journal of Computer and System Sciences 28*, 2 (1984), 270–299.

[43] GOLDWASSER, S., MICALI, S., AND RACKOFF, C. The knowledge complexity of interactive proof systems. *SIAM J. Comput. 18*, 1 (Feb. 1989), 186–208.

[44] GOLDWASSER, S., MICALI, S., AND RIVEST, R. L. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput. 17*, 2 (Apr. 1988), 281–308.

[45] GOLDWASSER, S., AND WAISBARD, E. Transformation of digital signature schemes into designated confirmer signature schemes. In *TCC* (2004), M. Naor, Ed., vol. 2951 of *Lecture Notes in Computer Science*, Springer, pp. 77–100.

[46] HITCHCOCK, Y., DAWSON, E., CLARK, A., AND MONTAGUE, P. Implementing an efficient elliptic curve cryptosystem over $gf(p)$ on a smart card. In *Proc. of 10th Computational Techniques and Applications Conference CTAC-2001* (Apr. 2003), K. Burrage and R. B. Sidje, Eds., vol. 44, pp. C354–C377. [Online] `http://anziamj.austms.org.au/V44/CTAC2001/Hitc` [April 1, 2003].

[47] HUANG, Q., AND WONG, D. S. New constructions of convertible undeniable signature schemes without random oracles. Cryptology ePrint Archive, Report 2009/517, 2009. `http://eprint.iacr.org/`.

[48] HUANG, Q., WONG, D. S., AND SUSILO, W. A new construction of designated confirmer signature and its application to optimistic fair exchange. In *Proceedings of the 4th international conference on Pairing-based cryptography* (Berlin, Heidelberg, 2010), Pairing'10, Springer-Verlag, pp. 41–61.

[49] HWANG, T. Cryptosystem for group oriented cryptography. In *Advances in Cryptology—EUROCRYPT 90* (21–24 May 1990), I. B. Damgård, Ed., vol. 473 of *Lecture Notes in Computer Science*, Springer-Verlag, 1991, pp. 352–360.

[50] JAO, D., AND YOSHIDA, K. Boneh-boyen signatures and the strong diffie-hellman problem. In *Proceedings of the 3rd International Conference Palo Alto on Pairing-Based Cryptography* (Berlin, Heidelberg, 2009), Pairing '09, Springer-Verlag, pp. 1–16.

[51] JOUX, A. A one round protocol for tripartite diffie-hellman. In *Proceedings of the 4th International Symposium on Algorithmic Number Theory* (London, UK, UK, 2000), ANTS-IV, Springer-Verlag, pp. 385–394.

[52] KOBLITZ, N. Elliptic curve cryptosystems. *Mathematics of Computation 48*, 177 (Jan. 1987), 203–209.

[53] KUROSAWA, K., AND HENG, S.-H. 3-move undeniable signature scheme. In *Proceedings of the 24th annual international conference on Theory and Applications of Cryptographic Techniques* (Berlin, Heidelberg, 2005), EUROCRYPT'05, Springer-Verlag, pp. 181–197.

[54] LAY, G.-J., AND ZIMMER, H. G. Constructing elliptic curves with given group order over large finite fields. In *Proceedings of the First International Symposium on Algorithmic Number Theory* (London, UK, UK, 1994), ANTS-I, Springer-Verlag, pp. 250–263.

[55] MAO, W. *Modern Cryptography: Theory and Practice*. Prentice Hall PTR, 2003.

[56] MENEZES, A., VANSTONE, S., AND OKAMOTO, T. Reducing elliptic curve logarithms to logarithms in a finite field. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing* (New York, NY, USA, 1991), STOC '91, ACM, pp. 80–89.

[57] MICHELS, M., AND STADLER, M. Efficient convertible undeniable signature schemes. *Proceedings of 4th International Workshop on Selected Areas in Cryptography, SAC 1997* (1997), 231–244.

[58] MICHELS, M., AND STADLER, M. Generic constructions for secure and efficient confirmer signature schemes. In *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding* (1998), K. Nyberg, Ed., vol. 1403 of *Lecture Notes in Computer Science*, Springer, pp. 406–421.

[59] MILLER, V. S. Use of elliptic curves in cryptography. In *Advances in Cryptology—CRYPTO '85* (18–22 Aug. 1985), H. C. Williams, Ed., vol. 218 of *Lecture Notes in Computer Science*, Springer-Verlag, 1986, pp. 417–426.

[60] M'RAÏHI, D., NACCACHE, D., POINTCHEVAL, D., AND VAUDENAY, S. Computational alternatives to random number generators. In *Proceedings of the Selected Areas in Cryptography* (London, UK, UK, 1999), SAC '98, Springer-Verlag, pp. 72–80.

[61] NAOR, M., AND YUNG, M. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing* (New York, NY, USA, 1990), STOC '90, ACM, pp. 427–437.

[62] NECHAEV. Complexity of a determinate algorithm for the discrete logarithm. *MATHNASUSSR: Mathematical Notes of the Academy of Sciences of the USSR 55* (1994).

[63] NIELSEN, J. B. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology* (London, UK, UK, 2002), CRYPTO '02, Springer-Verlag, pp. 111–126.

[64] OKAMOTO, T. Designated confirmer signatures and public-key encryption are equivalent. In *Proceedings of the 14th Annual International Cryptology Conference on Advances in Cryptology* (London, UK, UK, 1994), CRYPTO '94, Springer-Verlag, pp. 61–74.

[65] PAILLIER, P. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of the 17th international conference on Theory and application of cryptographic techniques* (Berlin, Heidelberg, 1999), EUROCRYPT'99, Springer-Verlag, pp. 223–238.

[66] PASS, R. On deniability in the common reference string and random oracle model. In *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings* (2003), D. Boneh, Ed., vol. 2729 of *Lecture Notes in Computer Science*, Springer, pp. 316–337.

[67] PEDERSEN, T. P. Non-interactive and information-theoretic secure verifiable secret sharing. In *Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology* (London, UK, UK, 1992), CRYPTO '91, Springer-Verlag, pp. 129–140.

[68] POINTCHEVAL, D. Contemporary cryptology provable security for public keyschemes. *Advanced Course on Contemporary Cryptology* (2005), 133–189.

[69] RACKOFF, C., AND SIMON, D. R. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology* (London, UK, UK, 1992), CRYPTO '91, Springer-Verlag, pp. 433–444.

[70] RIVEST, R. The md5 message-digest algorithm. *RFC 1321* (1992).

[71] RIVEST, R. L., SHAMIR, A., AND ADLEMAN, L. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM 21*, 2 (Feb. 1978), 120–126.

[72] SCHNORR, C.-P., AND JAKOBSSON, M. Security of signed elgamal encryption. In *Proceedings of the 6th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology* (London, UK, UK, 2000), ASIACRYPT '00, Springer-Verlag, pp. 73–89.

[73] SEMAEV, I. A. Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curve in characteristic p. *Mathematics of Computation 67*, 221 (Jan. 1998), 353–356.

[74] SHOUP, V. Lower bounds for discrete logarithms and related problems. In *Proceedings of the 16th annual international conference on Theory and applica-*

tion of cryptographic techniques (Berlin, Heidelberg, 1997), EUROCRYPT'97, Springer-Verlag, pp. 256–266.

[75] SMART, N. P. The discrete logarithm problem on elliptic curves of trace one. *J. Cryptology 12*, 3 (1999), 193–196.

[76] TATE, J. Duality theorems in galois cohomology over number fields. *Proceedings of the International Congress of Mathematicians (Stockholm, 1962)* (1962), 288–295.

[77] U.S. DEPARTMENT OF COMMERCE AND NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. *Secure Hash Standard - SHS: Federal Information Processing Standards Publication 180-4*. CreateSpace Independent Publishing Platform, USA, 2012.

[78] WANG, G., BAEK, J., WONG, D. S., AND BAO, F. On the generic and efficient constructions of secure designated confirmer signatures. In *Proceedings of the 10th international conference on Practice and theory in public-key cryptography* (Berlin, Heidelberg, 2007), PKC'07, Springer-Verlag, pp. 43–60.

[79] WANG, G., QING, S., WANG, M., AND ZHOU, Z. Threshold undeniable rsa signature scheme. In *Proceedings of the Third International Conference on Information and Communications Security* (London, UK, UK, 2001), ICICS '01, Springer-Verlag, pp. 221–232.

[80] WANG, G., XIA, F., AND ZHAO, Y. Designated confirmer signatures with unified verification. In *Proceedings of the 13th IMA international conference on Cryptography and Coding* (Berlin, Heidelberg, 2011), IMACC'11, Springer-Verlag, pp. 469–495.

[81] WANG, G. AND XIA, F. A pairing based designated confirmer signature scheme with unified verification. Tech. rep., Univerisity of Birmingham.

[82] WEI, B., ZHANG, F., AND CHEN, X. Society-oriented designated confirmer signatures. In *Proceedings of the Third International Conference on Natural Computation - Volume 05* (Washington, DC, USA, 2007), ICNC '07, IEEE Computer Society, pp. 707–712.

[83] WEI, B., ZHANG, F., AND CHEN, X. A new type of designated confirmer signatures for a group of individuals. *International Journal of Network Security 7*, 2 (2008), 293–300.

[84] WEIL, A. Sur les fonctions algebriques a corps de constantes fini. *Les Comptes rendus de Academie des sciences 210* (1940), 592–594.

[85] WIKSTRÖM, D. Designated confirmer signatures revisited. In *Proceedings of the 4th conference on Theory of cryptography* (Berlin, Heidelberg, 2007), TCC'07, Springer-Verlag, pp. 342–361.

[86] XIA, F., WANG, G., AND XUE, R. On the invisibility of designated confirmer signatures. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security* (New York, NY, USA, 2011), ASIACCS '11, ACM, pp. 268–276.

[87] XIE, Q., WANG, G., XIA, F., AND CHEN, D. Improvement of provably secure self-certified proxy convertible authenticated encryption scheme. In *INCoS* (2012), pp. 360–364.

[88] ZHANG, F., CHEN, X., SUSILO, W., AND MU, Y. A new signature scheme without random oracles from bilinear pairings. In *Proceedings of the First in-*

ternational conference on Cryptology in Vietnam (Berlin, Heidelberg, 2006), VIETCRYPT'06, Springer-Verlag, pp. 67–80.

[89] ZHANG, F., CHEN, X., AND WEI, B. Efficient designated confirmer signature from bilinear pairings. In *Proceedings of the 2008 ACM symposium on Information, computer and communications security* (New York, NY, USA, 2008), ASIACCS '08, ACM, pp. 363–368.