Design and Analysis of Fair Content Tracing Protocols

Geong Sen Poh

Technical Report
RHUL–MA–2009–15
14 May 2009

Department of Mathematics
Royal Holloway, University of London
Egham, Surrey TW20 0EX, England

http://www.rhul.ac.uk/mathematics/techreports

Design and Analysis of Fair Content Tracing Protocols

Geong Sen Poh

Thesis submitted to the University of London
for the degree of Doctor of Philosophy

Information Security Group
Department of Mathematics
Royal Holloway, University of London

2009

Declaration

These doctoral studies were conducted under the supervision of Prof. Keith M. Martin.

The work presented in this thesis is the result of original research carried out by myself, whilst enrolled in the Information Security Group of Royal Holloway, University of London as a candidate for the degree of Doctor of Philosophy. This work has not been submitted for any other degree or award in any other university or educational establishment.

Geong Sen Poh
March 2009

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor, Keith Martin, for his supervision and encouragement throughout my study. His invaluable comments and unwavering support have played a key role in shaping my research ability and instilling in me the right research attitudes. I would also like to thank my advisor, Chris Mitchell, for his constructive comments and advice.

I am very grateful to Allan Tomlison, Jason Crampton, Kenny Paterson and Peter Wild for their guidance and support. Many thanks to Adrian Leung for fruitful discussions. I am also indebted to David, Goi, JiQiang and Raphael Phan for the feedback on my work. My sincere thanks to Hoon Wei for all his help and to Qiang and ShengLan, without whom my settling down in the UK would not have been so smooth. Thanks to all my colleagues and friends for making my stay at Royal Holloway (especially the ISG) a most rewarding and memorable one.

To my parents and my brothers and sister, I cannot thank you enough for your encouragement and support throughout my studies.

Special thank goes to my beloved wife, Fern Nee. Without her love and support, I could not have possibly completed this.

Finally, I thank MIMOS Bhd for the generous financial support.

Abstract

The work in this thesis examines protocols designed to address the issues of tracing illegal distribution of digital content in a fair manner.

In digital content distribution, a client requests content from a distributor, and the distributor sends content to the client. The main concern is misuse of content by the client, such as illegal distribution. As a result, digital watermarking schemes that enable the distributor to trace copies of content and identify the perpetrator were proposed. However, such schemes do not provide a mechanism for the distributor to prove to a third party that a client illegally distributed copies of content. Furthermore, it is possible that the distributor falsely accuses a client as he has total control of the tracing mechanisms. *Fair content tracing* (FaCT) protocols were thus proposed to allow tracing of content that does not discriminate either the distributor or the client.

Many FaCT protocols have been proposed, mostly without an appropriate design framework, and so there is no obvious and systematic way to evaluate them. Therefore, we propose a framework that provides a definition of security and which enables classification of FaCT protocols so that they can be analysed in a systematic manner. We define, based on our framework, four main categories of FaCT protocols and propose new approaches to designing them.

The first category is *protocols without trusted third parties*. As the name suggests, these protocols do not rely on a central trusted party for fair tracing of content. It is difficult to design such a protocol without drawing on extra measures that increase communication and computation costs. We show this is the case by demonstrating flaws in two recent proposals. We also illustrate a possible repair based on relaxing the assumption of trust on the distributor.

The second category is *protocols with online trusted third parties*, where a central online trusted party is deployed. This means a trusted party must always be available during content distribution between the distributor and the client. While the availability of a trusted third party may simplify the design of such protocols, efficiency may suffer due to the need to communicate with this third party.

The third category is *protocols with offline trusted third parties*, where a central offline trusted party is deployed. The difference between the offline and the online

trusted party is that the offline trusted party need not be available during content distribution. It only needs to be available during the initial setup and when there is a dispute between the distributor and the client. This reduces the communication requirements compared to using an online trusted party. Using a symmetric-based cryptographic primitive known as Chameleon encryption, we proposed a new approach to designing such protocols.

The fourth category is *protocols with trusted hardware*. Previous protocols proposed in this category have abstracted away from a practical choice of the underlying trusted hardware. We propose new protocols based on a Trusted Platform Module (TPM).

Finally, we examine the inclusion of payment in a FaCT protocol, and how adding payment motivates the requirement for fair exchange of buying and selling digital content.

Notation

FaCT	Fair Content Tracing
$\mathbb{R}$	The set of real numbers
$\mathbb{Z}$	The set of integers
$\mathcal{X}$	Content space
X	Original content
X'	Watermarked content (e.g. marked with V)
X''	Doubly marked content (e.g. marked with V and W)
$\widehat{X}$	A found copy of content
$\widetilde{X}$	A content that is marked with one or two watermarks
$\mathcal{W}$	Watermark space
V, W	Watermark
$\mathcal{K}$	Key space
C	A client who requests (or buys) content
D	A distributor who distributes (or sells) content
CA	A Certificate Authority who issues digital certificate
WCA	A Watermark Certification Authority who generates watermark
PA	A Payment Agent
KC	A Key Centre who generates and distributes keys
A	An arbiter who settles disputes between C and D
TTP	A Trusted Third Party
TPM	A trusted hardware known as the Trusted Platform Module
IMSR	Integrity Measurement, Storage and Reporting, a mechanism to validate the integrity of softwares and processes
RTM	Root of Trust for Measurement, a computing engine that measures the softwares and processes in a computing platform
RTS	Root of Trust for Storage, a mechanism to store the integrity measurements computed by the RTM
RTR	Root of Trust for Reporting, a mechanism to report the integrity measurements of a computing platform when requested
DAA	Direct Anonymous Attestation, a group signature scheme used to anonymously authenticate a TPM
DAA Issuer	A Trusted Third Party that uses DAA to provide anonymous keys for a client with TPM

Privacy CA	A Trusted Third Party that provides anonymous keys to a client with TPM
PKI	Public Key Infrastructure
$(pvk_{\mathcal{I}}, ssk_{\mathcal{I}})$	Signature key pair of $\mathcal{I}$
$(hek_{\mathcal{I}}, hdk_{\mathcal{I}})$	Homomorphic encryption key pair of $\mathcal{I}$
$(pek_{\mathcal{I}}, pdk_{\mathcal{I}})$	Asymmetric encryption key pair of $\mathcal{I}$
$(pvk^*_{\mathcal{I}}, ssk^*_{\mathcal{I}})$	Anonymous signature key pair of $\mathcal{I}$
$(hek^*_{\mathcal{I}}, hdk^*_{\mathcal{I}})$	Anonymous homomorphic encryption key pair of $\mathcal{I}$
(pvk^*, ssk^*)	One-time signature key pair
(hek^*, hdk^*)	One-time homomorphic encryption key pair
$[\cdot]_{E_{(\cdot)}}$	An encrypted message generated using a symmetric encryption scheme
$[\cdot]_{PE_{(\cdot)}}$	An encrypted message generated using an asymmetric encryption scheme
$[\cdot]_{HE_{(\cdot)}}$	An encrypted message generated using a homomorphic encryption scheme
$[\cdot]_{SIG_{(\cdot)}}$	A digital signature
$[\cdot]_{COM_{(\cdot)}}$	A commitment
$H(\cdot)$	A hash value
$Cert_{ssk_{\mathcal{I}}}(\cdot)$	A digital certificate produced by $\mathcal{I}$
$\{\}_{AKE}$	A secure communication channel with authenticated key exchange
$ID_{\mathcal{I}}$	The identity information of $\mathcal{I}$
AGR	A content agreement with content description and licensing terms
PAY	A payment token that contains payment information
info	A general message that may contain any information
`SIG`	A signature or a group of signatures
$f(\cdot)$	A general object representing cryptographic or watermarking algorithms, for example, it can be an encryption algorithm
$f^{WM}()$	A general object representing a watermark detection algorithm

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

This chapter provides the motivation, contributions and structure of the thesis.

1.1 Motivation

The growth of the Internet and the continual advancement in computing power and size of storage have made mass distribution of digital content such as digital music, photos and videos possible. It is now common for a client to view, purchase or share digital content on the Internet (e.g. YouTube [88], Apple iTune Store [68]) or through portable entertainment devices such as an iPhone. While these allow more convenient access to digital content in comparison to physical counterparts, they cause one major concern: *illegal distribution of copyrighted content.*

The application scenario that we are interested in is how to address this concern by *deterring* a client from illegally distributing copies of content. The main solution that we consider in this thesis is to allow the distributor to *trace* the owner of illegal copies found on a network. The content tracing application deploys digital watermarking schemes (also known as fingerprinting schemes) [14, 77, 132]. Briefly, a digital watermarking scheme is a scheme that embeds a unique string (known as a *watermark*) into content without damaging the quality of this content, and in such

a way that it is hard for unauthorised parties to remove this watermark from the content. At a later stage the watermark can be detected, for example, to reveal the identity of the client that bought the content.

Hence in a content tracing application, a watermark carrying the identity of the client is embedded into content before it is given to the client. When an illegal copy is found, a content distributor can then detect the embedded watermark in order to identify the client who distributed this illegal copy. However, there are two issues that arise from this approach due to the distributor being in control of generating and embedding the client watermark [104, 115].

1. An innocent client, instead of the real perpetrator, may be falsely accused of illegally distributing copies of content. This is possible as the distributor (or a disgruntled employee working for the distributor) may frame an innocent client by embedding the client watermark into content and distributing copies of this content.
2. A dishonest client can claim that illegal copies of content distributed by him are actually distributed by the distributor, since the distributor owns the watermark of the client.

This creates a deadlock situation where a distributor is not able to prove to a third party that the dishonest client has illegally distributed content, while at the same time it is also possible that an innocent client is being framed by an unscrupulous distributor.

Many protocols [85, 94, 102, 104, 105, 115] have been proposed to alleviate this deadlock situation using digital watermarking schemes and cryptographic building blocks. These protocols, which we term *fair content tracing* (FaCT) protocols, provide content tracing to the distributor in such a way that the tracing is *fair* to both the distributor and the client. In other words, while it is possible for the distributor to trace the identity of a client from a watermarked content, the distributor is not able to frame a client. At the same time, a dishonest client who illegally distributes a copy of some content cannot claim otherwise.

Various FaCT protocols have been proposed without an appropriate framework, which makes them difficult to analyse. More importantly, the full solution space

of such protocols has not yet been explored, hence it is not clear whether new and improved protocols can be constructed based on alternative approaches. In this thesis, our focus is to examine and analyse FaCT protocols, and to explore alternative and better approaches to constructing them.

1.2 Contributions

This thesis examines FaCT protocols and proposes new approaches to constructing them. The contributions of the thesis are as follows:

- **A design and analysis framework** is proposed to provide a firm foundation for constructing and analysing FaCT protocols. The framework is used to avoid the often ambiguous and ad-hoc design approaches of some existing protocols. It is also used to consolidate at the conceptual level the many different ways of building FaCT protocols. It defines threats, security requirements, trust assumptions and the various environments in which these protocols are based. As a result, we are able to point out design flaws in recent proposals and explore new approaches that have not been proposed before. The framework also includes the classification of existing FaCT protocols into four main categories. This work was partially published in [109].

- **Analysis of existing protocols and new approaches to constructing FaCT protocols**.

  Firstly, we look at *protocols without trusted third parties*. Existing protocols in this category normally have high communication and computation costs. Attempts were made by some recent proposals to reduce these costs, but we demonstrate that these are flawed. We then propose a possible approach that reduces these costs by relaxing the trust assumption on the distributor. In other words, by trusting the distributor a little bit more than the client, it is possible to construct an efficient protocol in this category. Our study suggests that it is a challenging task to design FaCT protocols without trusted third parties. Part of this work was published in [111, 113].

  Secondly, we examine *protocols with online trusted third parties*. We investigate three existing protocols and discuss security issues concerning them. This work was partially published in [112].

Thirdly, we examine *protocols with offline trusted third parties*. We propose a new approach that deploys a recently proposed symmetric cryptographic building block to reduce the reliance on the trusted third party. Our new approach is computationally efficient compared to existing FaCT protocols, mainly because existing FaCT protocols use asymmetric cryptographic building blocks, which are relatively computationally intensive in comparison. Part of this work was published in [110].

Fourthly, we examine *protocols with trusted hardware*. We propose two protocols based on trusted computing, using the now standardised Trusted Platform Module (TPM). Such a design has not been proposed before, and it is a more practical solution than existing proposals that are based only on an abstract definition of trusted hardware. This work was jointly conducted with Adrian Leung and was partially published in [86].

- Finally, we explore FaCT protocols involving payment. We examine the issues when payment is included in FaCT protocols in the four categories. We further propose a FaCT protocol that includes payment and provides the additional property of fair exchange.

1.3 Organisation of Thesis

In the following we outline the structure of the thesis:

Fair Content Tracing: In Chapter 2, we introduce content distribution, fair content tracing and fair content tracing protocols. We also describe in detail the underlying building blocks required to construct these protocols.

A Framework: In Chapter 3, we propose a framework for FaCT protocols. We further classify existing protocols into categories and illustrate as an example one of the earliest FaCT protocols known as the Memon-Wong buyer-seller watermarking protocol.

Protocols without Trusted Third Parties: In Chapter 4, we study and analyse FaCT protocols that do not require a trusted third party during content distribution between the distributor and the client. We describe the benefits and issues of the existing protocols in this category. We also show how two recently

proposed protocols contain flaws. Finally, we describe a possible approach to constructing an efficient protocol by reconsidering the trust assumption on the distributor.

Protocols with Online Trusted Third Parties: In Chapter 5, we examine FaCT protocols with online trusted third parties. We illustrate some existing protocols and discuss the benefits and security issues of these protocols.

Protocols with Offline Trusted Third Parties: In Chapter 6, we examine FaCT protocols with offline trusted third parties. We describe an existing protocol and then propose a new one based on Chameleon encryption. We demonstrate that the new protocol has better computational performance, while placing less reliance on the trusted third party.

Protocols with Trusted Hardware: In Chapter 7, we examine FaCT protocols with trusted hardware. We begin by looking at existing proposals that are constructed based on an abstraction of trusted hardware. We then construct protocols based on a Trusted Platform Module (TPM).

Protocols with Payment and Fair Exchange: In Chapter 8, we study the addition of payment and how it motivates the requirement for fair exchange. We describe a protocol with a fair exchange mechanism and analyse its security and performance.

Conclusions: In Chapter 9, we summarise our discussions by reinforcing the issues that motivate our research and our contributions toward solving them. We also suggest possible directions for future research on FaCT protocols.

Chapter 2

Fair Content Tracing Protocols

Contents

This chapter introduces fair content tracing protocols. We define fair content tracing, why it is important and the issues that it addresses. This in turn motivates the construction of fair content tracing protocols. We provide a definition and brief review of these protocols and survey in detail the fundamental building blocks that are required to construct these protocols.

2.1 Motivation

In this section we discuss content distribution, the issues of illegal distribution and one of the techniques proposed to address this issue. This technique is known as

content tracing. Next we reason why content tracing is not sufficient, resulting in the proposal of fair content tracing protocols.

2.1.1 Content Distribution

Digital content (or *content*) is multimedia files in the form of digital images, digital audio (e.g. songs) or digital video (e.g. movies). Some common examples are images in JPEG format [59], audio in MP3 format [60] and video in H.264/MPEG-4 AVC (Advanced Video Coding) format [133]. Distribution of content, such as sharing, viewing and purchasing of songs and movies, has become very common and can be performed with ease. This is especially true given today's convenient and fast access to widely available computer networks such as the Internet. Access to digital content is also getting more and more pervasive, as can be seen from the various computing devices that are now being used for this purpose. These include laptop computers, mobile phones and personal digital assistants (PDAs).

In conjunction with these developments, various models for the distribution of digital content have been developed. We briefly mention three common models:

- *Content Broadcast.* The first model is broadcast of content. In this model, a broadcaster broadcasts one copy of content to many clients. In order to gain access to this content, clients need to subscribe to the content broadcast services provided by the broadcaster. Examples of these are Pay-TV systems [128], and emerging IPTV systems.

- *Buyer-Seller Content Purchase.* The second model relates to buying and selling of content through online uploading/downloading facilities. One method is for a client to purchase content from the distributor. Examples are the services provided by iTunes Store [68] and Amazon Unbox video downloads [67]. In this case the clients subscribe to the content distribution service by registering on the websites of these distributors. The clients then purchase content by downloading from the provided downloading facilities. It is also possible that there is no purchasing involved. The clients need only to register on the websites provided by the distributor and proceed to download content. One such example is BBC iPlayer [7]. The second method is for a distributor to send content directly to a client when a client requests it. In this case no

subscription is required. The client provides the distributor with all necessary information (e.g. payment) together with the content request. This is commonly known as a "pay-per-view" service. An example is the service provided by CinemaNow [69]. As a final method, it is also possible for a content author to license content to many distributors. For example, a content author releases authorised copies of content to movie theatres for public screening.

- *Peer-to-Peer.* The last example is the peer-to-peer model. In contrast to the above models, this model does not have a central distributor, but many clients that also act as distributors. These are currently among the most popular models for file sharing. One example is Gnutella [52]. A framework has been proposed concerning how content distribution can be performed in a trusted and legal peer-to-peer environment [121], but there are concerns about the use of peer-to-peer networks for mass distribution of copyrighted content without the consent of the copyright holders [11].

2.1.2 Content Tracing

We have just discussed how content can be easily and efficiently distributed based on various distribution models. A significant problem is that in many of these models a client, after obtaining songs or movies, can easily make many copies and mass distribute them without the consent of the distributor. Therefore, methods have been proposed to alleviate this concern. One of these methods is *content tracing*. Other methods include Digital Rights Management (DRM) system such as the Window Media DRM [27].

In general, *content tracing* is a technique that gives a distributor the capability to *trace* the identity of a client based on a copy of content. To achieve this, the distributor generates and places a unique string (commonly known as a *watermark*) into content to create a marked copy. This marked copy is given to the client. The distributor stores the watermark as the client identifier, together with other information about the client. When marked content is found, the distributor can trace the identity of the client based on the detected watermark. The process of embedding and detecting the watermark is realised by schemes known as *digital watermarking schemes*, which we will discuss in Section 2.3.1. As an example, a practical content tracing system developed by Philips using digital watermarking

schemes has been deployed. It is used by Technicolor to distribute and trace screener copies of content (or pre-release movies) that are meant for voting members of the Oscar award [29]. Such tracing techniques can be deployed in all three models discussed in Section 2.1.1 to trace distribution of content. More importantly, letting clients know that there is a tracing mechanism in place may help to *deter* them from making copies and illegally distributing these copies.

2.1.3 Fair Content Tracing

We have just discussed how *content tracing* can deter a client from illegally distributing copies of content. However, Qiao and Nahrstedt [115], and Pfitzmann and Schunter [104] independently pointed out concerns with such an approach. In content tracing, the distributor generates and embeds a watermark into content in order to allow him to trace marked copies of content and identify the client that owns them. In this situation, the distributor has in his possession the original content, the watermark and the marked copy. It is clear that a client has no choice but to trust the distributor to act honestly. This is because the distributor can embed a watermark into any content, distribute copies of this content, and frame a client for illegally distributing content. Conversely, due to this framing possibility, the distributor is able to trace a client who redistributes copies of content but is not able to prove this fact to a third party. This is because a dishonest client can claim that illegal copies are distributed by the distributor.

To further clarify the above issues, let us again examine the *buyer-seller content purchase* model illustrated in Section 2.1.1. When content tracing is in place, an innocent client may be wrongly implicated if the client's marked copy is leaked by the distributor or by other parties working/sharing resources with this distributor. It is also possible that, due to operational errors, a marked copy supposedly meant for client C_1 is accidentally given to another client C_2. Similarly, such incidents may also happen in the *content broadcast* and *peer-to-peer* models. Conversely, if a dishonest client redistributes copies of content, it will be difficult for the distributor to provide evidence to prove that the client has redistributed these copies. A client can claim that the leaked marked copy is due to errors on the side of the distributor, or that the distributor is simply trying to frame him (which may be true).

In essence, the issues that render content tracing to be insufficient can be summarised

as follows:

1. The client is worried that they may be falsely accused of illegal distribution since the distributor has all the power to generate and embed a watermark into content. Meanwhile, a dishonest client who distributes copies illegally can deny doing so due to the fact that it is easy for the distributor to distribute these copies using the client's watermark.
2. The distributor is not able to prove to a third party that the client has illegally distributed copies of content.

Therefore, in addition to content tracing, techniques must be provided to address the above issues. In other words, content tracing must be performed in a way that is *fair* and does not discriminate either the client or the distributor. We term such techniques as *fair content tracing*. More formally, we say that *fair content tracing* is a content tracing technique that traces content in a fair manner for the distributor and the client by:

- preventing an unscrupulous distributor from being able to frame an innocent client,
- allowing the distributor to prove the illegal action of a dishonest client.

When Privacy Is a Concern. Additionally, one issue put forward by Pfitzmann and Waidner [105] is that clients should not need to reveal their identities just because a distributor wishes to be able to trace some dishonest clients who illegally distributed copies of content. The revealing of the identity should only be allowed for those clients who have misused their rights on the content that they owned. If this is the case then *fair content tracing* includes the additional goal of protecting the privacy of clients.

When Payment Is Involved. Similarly, when the distribution of content involves buying and selling, then the distributor will want to receive correct payment, while the client will want to receive correct content. If this is the case then *fair content tracing* includes the additional goal of ensuring that the distributor and the client

trade fairly. We remark that this issue has not been discussed before and we will examine it in more detail in Chapter 8.

Fair Content Tracing (FaCT) Protocols. Fair content tracing is provided by what we call *fair content tracing (FaCT) protocols*. A FaCT protocol is an interactive protocol that provides content distribution between a distributor and a client, in which the client who receives content can be traced in a fair manner if copies of this content are found to be illegally distributed. By *fair* we mean that a FaCT protocol fulfills the goals of content tracing and fair content tracing as discussed in Section 2.1.2 and Section 2.1.3. In the next chapter we re-examine the objectives of FaCT protocols by defining the threats faced by FaCT protocols and security requirements based on these threat scenarios.

2.2 Existing FaCT Protocols

Two variants of FaCT protocols have been proposed. These are *buyer-seller watermarking (BSW)* protocols and *asymmetric fingerprinting (AF)* protocols.

BSW protocols were first proposed by Qiao and Nahrstedt [115] and later improved by Memon and Wong [94]. More recently, Ju *et al.* [74] presented a protocol that also protects client privacy. Several BSW protocol variants have since been proposed, including [25, 26, 33, 34, 50, 54, 65, 81, 85, 123]. In most of these protocols:

- Digital watermarking schemes are deployed for content tracing.
- A special trusted third party is introduced to generate client watermarks, instead of letting the distributor generate them.
- Asymmetric homomorphic encryption schemes such as Paillier [99] are deployed, together with digital watermarking schemes such as the spread spectrum watermarking scheme [28], in such a way that the party (i.e. the distributor) who embeds a watermark into content has no idea what the watermark is. This technique is termed *watermarking in the encrypted domain* [41, 44, 108, 120]. We will examine these building blocks in Section 2.3.
- Digital signature schemes such as RSA-PSS [83] are used to ensure that a dishonest client cannot repudiate the fact that copies of content were illegally

distributed.

AF protocols were first proposed by Pfitzmann and Schunter in [104]. This idea was extended to include client privacy in [105]. In most of these protocols, watermarking in the encrypted domain also plays a key role and:

- Digital watermarking schemes are deployed for content tracing.
- Instead of introducing a trusted third party to generate watermarks for clients, the client is responsible for generating their own watermark, while the distributor is responsible for embedding this watermark into content. Homomorphic bit commitment schemes [17] are deployed in conjunction with zero-knowledge proof systems [57, 47] to prevent the client from manipulating the watermark generation process. The client, after generating the watermark, must prove in zero-knowledge to the distributor that the generated watermark is well-formed.
- Similar to BSW protocols, digital signature schemes are deployed to prevent a dishonest client from denying the act of illegal content distribution.

Other variants were later proposed in [19, 23, 40, 78, 80, 102, 103, 118].

We will examine in detail the properties of these protocols by categorising them based on a framework in Chapter 3 and analysing different protocols in the subsequent chapters. In the following section, we define and describe the building blocks that are used to construct FaCT protocols.

2.3 Building Blocks

We now discuss some building blocks required to construct FaCT protocols.

2.3.1 Digital Watermarking Schemes

We introduce digital watermarking and present two well-established schemes known as *Spread Spectrum* (SS) [28] and *Quantization Index Modulation* (QIM) [21] watermarking schemes.

Digital Watermarking. Watermarking can be traced back to 1292 in the era of paper making in Italy. The main idea was to embed identities of paper mills, and identities of the artists that refined these papers, as translucent images on the papers [75]. The buyers looked at these watermarks to differentiate and compare the quality of the produce. In the 17th and 18th centuries, the publishers of logarithm tables used the same concept by deliberately introducing errors in the least significant bits of the numbers [75].

Digital watermarking is in many ways similar to the traditional watermarking techniques illustrated above. However, for most computer applications, a digital watermark is normally imperceptible after it is embedded into content. A digital watermark can be thought of a message that, when embedded into content, can later be extracted in order to identify a client that owns this content. It can also be used for applications such as copy prevention, ownership identification and data authentication [29, 30, 62, 114].

Digital Watermarking Schemes. Many digital watermarking schemes [21, 28, 29, 62] have been proposed. A digital watermarking scheme consists of three algorithms: a *key generation algorithm* that generates a secret key, an *embedding algorithm* that uses the key to embed a watermark into content and a *detection algorithm* that detects the watermark from a marked copy of content. In addition, a *watermark generation algorithm* is also needed to generate the watermark. Both the key and the watermark must be kept secret.

A digital watermarking scheme can be classified as either *blind* or *non-blind* (also known as *blind* or *informed* in [29]) depending on the inputs to the detection algorithm [2, 75]. A *blind* watermarking scheme means that the detection algorithm detects the embedded watermark from a marked copy of content based only on this marked copy. A *non-blind* watermarking scheme means that the detection algorithm detects the embedded watermark from a marked copy of content based on the marked copy, and also the original content or other information related to the original content.

A digital watermarking scheme can further be classified as *symmetric* or *asymmetric* [2]. A *symmetric* watermarking scheme uses identical secret keys for embedding and detection, whereas an *asymmetric* watermarking scheme has a key pair: an *em-*

bedding key for watermark embedding and a *detection key* for watermark detection.

In this thesis we assume that the digital watermarking schemes being deployed are *non-blind* and *symmetric*. The reason for this is that in FaCT protocols, watermark embedding and detection are both performed by the distributor (or a trusted third party), who is in possession of the key and the original content. Hence we can use a *symmetric* scheme since there is no key distribution issue, and use a *non-blind* scheme since the original content is available for watermark detection. This is beneficial as it is known that watermark detection is more effective given the presence of the original content [28, 87]. Formally, we define a *non-blind* and *symmetric* watermarking scheme in Definition 2.1, based on the definition in [2].

Definition 2.1 ([2]) *A* non-blind and symmetric digital watermarking scheme *consists of three polynomial-time algorithms:*

- *A key generation algorithm, $\mathcal{G}_w$. On input of the security parameter p^w, $\mathcal{G}_w$ outputs a key wmk.*
- *A watermark embedding algorithm $[\cdot,\cdot]_{EMB_{(\cdot)}}$, where given watermark W and content X, the algorithm outputs a marked content X':*

$$X' \leftarrow [X, W]_{EMB_{wmk}}.$$

- *A watermark detection algorithm $[\cdot,\cdot,\cdot]_{DET_{(\cdot)}}$, where given a marked content X', watermark W, and the original content X, the algorithm outputs either* `true` *or* `false`*:*

$$\{\texttt{true}, \texttt{false}\} \leftarrow [X', W, X]_{DET_{wmk}}.$$

We require that, for all X, W, and $wmk \in \{\mathcal{G}_w\}$:

$$X' \leftarrow [X, W]_{EMB_{wmk}} \Longrightarrow [X', X]_{SIM} = \texttt{true},$$

where $[X', X]_{SIM}$ is a function that decides whether X' is similar to X, and for correctness,

$$X' \leftarrow [X, W]_{EMB_{wmk}} \Longrightarrow [X', W, X]_{DET_{wmk}} = \texttt{true}.$$

In Definition 2.1, the function $[\cdot,\cdot]_{SIM}$ is required to ensure that embedding of the watermark into content does not affect the quality of the content. In other words, the marked copy of content should be perceptibly similar to that of the original content. Cox *et al.* provide a good overview on designing such a function in Chapter 8 of [29].

Two security properties of a digital watermarking scheme are crucial for the effectiveness of content tracing. These are:

- *Robustness.* A watermarking scheme is said to be *robust* if it can detect the embedded watermark even when the marked content is modified (either due to common signal processing, such as compression, or intentional change), as long as the marked content is still perceptibly similar to the original content [2]. This also means that when a watermark is successfully removed, the modified content is of such low quality that it is of no value anymore.

- *Collusion resistance.* A watermarking scheme is said to have *collusion resistance* if it is *robust* to watermark removal based on comparing many unique copies of the marked content with distinct watermarks owned by the clients [28, 87, 104].

In this thesis we assume that the digital watermarking schemes used in FaCT protocols provide the above security properties. We describe two well-established watermarking schemes in the following.

Spread Spectrum Watermarking Schemes. *Spread Spectrum* (SS) watermarking schemes were first proposed by Cox *et al.* [28]. In their proposal, watermarking is modeled as a communication channel such as a radio transmission where signal jamming is possible. The watermark is considered as the signal to be transmitted through the host signal (which is the original content in our discussion), while the noise introduced is the jamming signal. If the watermark is carried in a relatively narrow frequency band in the host signal, a jammer can allocate all the power to this band of frequencies to remove the watermark.

The idea is to "spread" the watermark signal throughout the host signal so that the jammer has to spread its power over a wide range of frequencies, which greatly reduces the effect on the watermark signals, since only a small fraction of that power reaches the watermark signal. In other words, the watermark can be a sequence of small real numbers and these are added to many locations in the content in such a way that it is difficult for an attacker to remove them.

The SS scheme proposed by Cox *et al.* [28] remains one of the most well-established techniques. The watermark is embedded in the most significant parts of a content, while introducing only minimum distortion. This was different to most of the previous techniques that were based on embedding in the least significant parts of the

content. A basic SS scheme of Cox *et al.* [28] is presented in Figure 2.1.

1. Let content $X = (x_1, \ldots, x_n)$ and watermark $W = (w_1, \ldots, w_n)$, where both x_i, $w_i \in \mathbb{R}$, and n is the number of the most significant elements in the content to be watermarked. For example, $n = 1000$ and $(x_1, \ldots, x_{1000})$ are the one thousand most significant DCT (Discrete Cosine Transform) coefficients of a digital image [28].
2. The embedding algorithm is:

$$X' \leftarrow [X, W]_{EMB_{wmk}} \Longleftrightarrow x'_i = x_i + \rho w_i \qquad 1 \leq i \leq n, \tag{2.1}$$

   where ρ is a real number, which is determined by the robustness and content quality requirements of the watermarking scheme. Higher values of ρ result in more robustness but cause more distortion to the content.
3. The detection algorithm $\{\texttt{true}, \texttt{false}\} \leftarrow [X', W, X]_{DET_{wmk}}$ consists of two steps:
   - *Watermark Extraction.* This is performed by subtracting the original content X from the marked copy X':

$$\rho w'_i = x'_i - x_i \qquad 1 \leq i \leq n. \tag{2.2}$$

   - *Watermark Detection.* After the watermark W' is extracted, the correlation between this watermark W' and the original watermark W is computed:

$$\frac{W'W}{\sqrt{W'W'}} > t = \{\texttt{true}, \texttt{false}\}, \tag{2.3}$$

     where t is a predetermined threshold. If the result is `true`, then the watermark is present in the marked copy of content.

Figure 2.1: Cox *et al.*'s Spread Spectrum Watermarking Scheme

It is possible to construct an alternative embedding algorithm for the scheme shown in Figure 2.1 that has the following form:

$$X' \leftarrow [X, W]_{EMB_{wmk}} \Longleftrightarrow x'_i = x_i(1 + \rho w_i) \qquad 1 \leq i \leq n. \tag{2.4}$$

This alternative embedding algorithm is useful when the value of the content elements x_i vary widely. For example, if $x_i = 1000$ then adding 1 may not affect the content in the original embedding algorithm (2.1), but if $x_i = 1$ then adding 1 will totally distort the original value [28]. In this situation the alternative algorithm should be used. It is also worth noting that we denote the extracted watermark as W' in Figure 2.1 since it is possible that the marked copy of content was modified and the watermark extracted is not the exact copy of the embedded watermark W. We have also omitted details of the key generation algorithm. This is because this

scheme can be keyless (in this case the secret information is the watermark W). For example, based on equations (2.1) and (2.4), the SS scheme works by *adding* the watermark into many elements of content. Without the knowledge of the watermark and the original content, an attacker's best strategy is to try to guess each watermark element, or use signal processing techniques to remove the watermark. It is assumed that the attacker knows the watermarking schemes and all other parameters in use.

It has been known that for SS schemes, guessing the watermark is hard. Experiments have shown that such schemes are robust against signal processing [28, 87, 125]. Guessing the watermark may be made more difficult if the *locations* of the embedded watermark elements are kept secret. In this case the key *wmk* is the embedding locations.

Quantization Index Modulation (QIM) Watermarking Schemes. QIM watermarking schemes were first proposed by Chen and Wornell [21], based on the idea of quantisation, which we now describe.

In signal processing, before an analog signal is converted to a digital signal, each analog sample is assigned one of b values. For example, given $b = 4$ and an analog signal with continuous input from 0 to 4, the analog-to-digital conversion has the following input and output (Table 2.1).

Table 2.1: Quantisation: A Simple Example

Continuous Values Inputs	**Discrete Values Outputs**
$0.0000 \leq x < 1.0000$	$x = 0$
$1.0000 \leq x < 2.0000$	$x = 1$
$2.0000 \leq x < 3.0000$	$x = 2$
$3.0000 \leq x \leq 4.0000$	$x = 3$

This process is called *quantisation*. Briefly, it takes a large set of values and maps these values to a smaller set. This process results in a loss of information, and such losses are termed *quantisation errors*. The values $0, 1, 2$ and 3 represents the *quantisation levels*, and the interval between two levels is the *quantisation step size*. For example, the quantisation step size is 1 for the above example. The basic concept of QIM watermarking is based on the quantisation technique described above. A QIM watermarking scheme is shown in Figure 2.2. This example follows the simplest case of embedding one bit in a real-valued sample given in [96].

1. Let content $X = (x_1, \ldots, x_n)$ and watermark $W = (w_1, \ldots, w_n)$, where $x_i \in \mathbb{R}$ and $w_i \in \{0, 1\}$.
2. Let the key $wmk = d$, where d is the quantisation step size.
3. Let $Q(u) = d\lfloor u/d \rfloor$, $u_0 = x_i + d/4$, $u_1 = x_i - d/4$ and

$$\begin{array}{rcl} Q_0(u_0) & = & Q(u_0) - d/4 \; ; \\ Q_1(u_1) & = & Q(u_1) + d/4 \end{array} \quad 1 \leq i \leq n. \tag{2.5}$$

4. The embedding algorithm $X' \leftarrow [X, W]_{EMB_{wmk}}$ is defined as

$$x'_i = \begin{cases} Q_0(u_0) & \text{if} \quad w_i = 0 \; ; \\ Q_1(u_1) & \text{if} \quad w_i = 1 \end{cases} \quad 1 \leq i \leq n. \tag{2.6}$$

5. The detection algorithm $\{\texttt{true}, \texttt{false}\} \leftarrow [X', W, X]_{DET_{wmk}}$ consists of two steps:
   - *Watermark Extraction.* This is performed as follows:

$$\begin{array}{lll} w'_i = 0 & \text{if} & |x'_i - Q_0(u_0)| < |x'_i - Q_1(u_1)| \\ w'_i = 1 & \text{if} & |x'_i - Q_1(u_1)| < |x'_i - Q_0(u_0)| \end{array} , \quad 1 \leq i \leq n, \tag{2.7}$$

     where $|.|$ denotes absolute value.
   - *Watermark Comparison.* The extracted watermark W' is compared with the original watermark W. If $W' = W$ then the output is `true`. It is `false` otherwise.

Figure 2.2: Chen and Wornell's Scalar-QIM Algorithm [96] [21]

We provide a hypothetical example of the working of the scheme shown in Figure 2.2. With $d = 4$, $x_i = 50$, we have:

$$\begin{array}{rcl} u_0 & = & 50 + 1 = 51, \\ u_1 & = & 50 - 1 = 49, \\ Q_0(u_0) & = & 4\lfloor 51/4 \rfloor - 1 = 47 \quad \text{if} \quad w_i = 0, \\ Q_1(u_1) & = & 4\lfloor 49/4 \rfloor + 1 = 49 \quad \text{if} \quad w_i = 1. \end{array}$$

So if we are to embed $w_i = 0$, then $x_i = 50$ is replaced by $x_i = 47$. Similarly, if we are to embed $w_i = 1$, then $x_i = 50$ is replaced by the value 49. To extract the watermark, let $w_i = 0$, we have $x'_i = 47$ and thus the detected value can be calculated as:

$$\begin{array}{lll} w'_i = 0 & \text{since} & |47 - 47| < |47 - 49| \\ w'_i \neq 1 & \text{since} & |47 - 49| \not< |47 - 47| \end{array} .$$

The security of the scheme depends on keeping both d and W secret. This simple scheme may cause visible distortion if the noise added to the marked sample exceeds

$d/4$. Hence an improved scheme, named *distortion-compensated scalar QIM*, was also proposed. The embedding algorithm for this improved scheme is:

$$x_i{}' = \begin{cases} Q_0\left(\alpha u_0\right) + \left(1-\alpha\right) x_i & \text{if } \mathrm{w} = 0\ ; \\ Q_1\left(\alpha u_1\right) + \left(1-\alpha\right) x_i & \text{if } \mathrm{w} = 1\ , \end{cases} \quad 1 \leq i \leq n, \tag{2.8}$$

where $\alpha \in [0, 1]$. It can be observed that (2.8) is identical to (2.6) when $\alpha = 1$. In general, adjusting the value of α allows one to adjust the distortion introduced to the content by the watermark. Details of the QIM schemes can be found in [21].

In summary, the SS and QIM watermarking schemes can be deployed for content tracing, and they currently serve as the main watermarking schemes used in most of the existing FaCT protocols, such as the protocols in [80, 85, 94].

2.3.2 Encryption Schemes

An *encryption scheme* is a method that enables two parties to communicate with one another through an insecure communication channel without a third party knowing what the message being transmitted is [124]. It consists of three algorithms. These are the *key generation algorithm* that generates key(s), an *encryption algorithm* that encrypts a message (a *plaintext*) to produce an encrypted message (a *ciphertext*), and a *decryption algorithm* that decrypts the *ciphertext* to recover the *plaintext*. Anyone who gets hold of the *ciphertext* is not able to determine what the *plaintext* is if he does not have possession of the decryption key. Whether an identical key is used for encryption and decryption, or different keys are used, depends on the type of the encryption scheme.

Symmetric Encryption Schemes. In these schemes, two parties who wish to communicate securely with one another share an identical secret key for encryption and decryption of messages. Prior to sending a secret message, both parties must find a secure way to agree on and obtain the secret key, such as through courier or use of a trusted third party. After that, the sender uses the encryption algorithm to encrypt the plaintext using this secret key and sends the resulting ciphertext to the receiver. The receiver uses the identical secret key to decrypt the ciphertext into plaintext. Symmetric encryption schemes can further be categorised into *block ciphers* and *stream ciphers*. The main difference between these two categories is that block ciphers encrypt the plaintext in blocks of bits, while stream ciphers encrypt the

plaintext bit-by-bit. One example of a block cipher is AES [72], and an example of a stream cipher is SNOW [42]. In this thesis, when we use a symmetric encryption scheme we will normally imply use of a block cipher. We formalise a symmetric encryption scheme in Definition 2.2.

Definition 2.2 ([124]) *A* symmetric encryption scheme *is a triple* $(\mathcal{G}_h, [\cdot]_{E_{(\cdot)}}, [\cdot]_{D_{(\cdot)}})$ *of polynomial-time algorithms:*

- *On input* 1^k*, where* k *is the security parameter, the key generation algorithm* $\mathcal{G}_h$ *outputs a secret key* sk*.*
- *On input of content* X *and by using the secret key* sk*, encryption is performed as* $Y \leftarrow [X]_{E_{sk}}$*, where* Y *is the encrypted content.*
- *On input of encrypted content* Y *and by using the secret key* sk*, decryption of an encrypted content is performed as* $X \leftarrow [Y]_{D_{sk}}$*.*

For correctness, we require that for all $Y \in \{[X]_{E_{sk}}\}$*:*

$$[Y]_{D_{sk}} = X.$$

Asymmetric Encryption Schemes. In an asymmetric encryption scheme, two keys are generated. One is the *public key* for encryption, and the other is the *private key* for decryption. A public key is not secret and can be distributed to anyone who wishes to encrypt messages intended for the owner of this public key. For example, if a distributor wishes to encrypt content meant for the client, the distributor encrypts this content with the client's public key. When the client receives the encrypted content, the client decrypts it by using his private key. This private key is kept secret by the client. While asymmetric encryption schemes avoid the need to distribute identical keys to sender and receiver, they are generally more computationally expensive than symmetric schemes. One example of a well-known asymmetric encryption scheme is RSA [117]. We formalise an asymmetric encryption scheme in Definition 2.3.

Homomorphic Encryption Schemes. A homomorphic encryption scheme is an encryption scheme having a special property known as a *privacy homomorphism*, which we formalise in Definition 2.4. Most of the homomorphic encryption schemes proposed to date are asymmetric. Very few symmetric schemes have been proposed and flaws have been discovered in most of them [48]. Examples of asymmetric ho-

Definition 2.3 ([124]) *An* asymmetric encryption scheme *is a triple* $(\mathcal{G}_h, [\cdot]_{PE_{(\cdot)}}, [\cdot]_{PD_{(\cdot)}})$ *of polynomial-time algorithms:*

- *On input* 1^k*, where* k *is the security parameter, the key generation algorithm* $\mathcal{G}_h$ *outputs a key pair (pek, pdk).*
- *On input of content* X *and by using the public encryption key pek, encryption is performed as* $Y \leftarrow [X]_{PE_{pek}}$*, where* Y *is the encrypted content.*
- *On input of encrypted content* Y *and by using the private decryption key pdk, decryption of an encrypted content is performed as* $X \leftarrow [Y]_{PD_{pdk}}$*.*

For correctness, we require that for all $Y \in \{[X]_{PE_{pek}}\}$*:*

$$[Y]_{PD_{pdk}} = X.$$

momorphic schemes are the original RSA [117], Goldwasser-Micali [56], Paillier [99] and Okamoto-Uchiyama [97].

Definition 2.4 ([48, 124]) *An* asymmetric homomorphic encryption scheme *is a triple* $(\mathcal{G}_h, [\cdot]_{HE_{(\cdot)}}, [\cdot]_{HD_{(\cdot)}})$ *of polynomial-time algorithms:*

- *On input* 1^k*, where* k *is the security parameter, the key generation algorithm* $\mathcal{G}_h$ *outputs a key pair (hek, hdk).*
- *On input of content* X *and by using the public encryption key hek, encryption is performed as* $Y \leftarrow [X]_{HE_{hek}}$*, where* Y *is the encrypted content.*
- *On input of encrypted content* Y *and by using the private decryption key hdk, the decryption of an encrypted content is performed as* $X \leftarrow [Y]_{HD_{hdk}}$*.*

For correctness, we require that for all $Y \in \{[X]_{HE_{hek}}\}$*:*

$$[Y]_{HD_{hdk}} = X.$$

Privacy homomorphism. *We further require that for content* X_1 *and content* X_2*,*

$$[X_1]_{HE_{hek}} \circ [X_2]_{HE_{hek}} = [X_1 \circ X_2]_{HE_{hek}},$$

where $\circ$ *denotes either addition or multiplication depending on the underlying asymmetric homomorphic encryption scheme.*

The standard notion of security for an encryption scheme is known as *indistinguishability* [56]. This means that an efficient (polynomial-time) attacker is not able to learn any bit about the plaintext from the ciphertext, except the length of the plaintext. An encryption scheme that fulfills this requirement is known as a *semantically secure* scheme. The Paillier encryption scheme [99] is one such scheme.

In this thesis we assume that the encryption schemes used in a FaCT protocol are at least semantically secure. In the following we provide three examples of asymmetric homomorphic encryption schemes that have been used in existing FaCT protocols.

RSA Encryption Scheme. The RSA encryption scheme was proposed by Rivest, Shamir and Addleman [117] and is one of the most well-known schemes. We describe the RSA encryption scheme and its homomorphic property in Figure 2.3. In this scheme, every time the same message is given, the encryption algorithm (2.9) will output the same ciphertext. The RSA scheme is thus a *deterministic* scheme. This *deterministic* nature means that the scheme is not semantically secure. Such a characteristic is not desirable, especially when the plaintext space is small, for example $X \in \{0, 1\}$. If this is the case then there are only two possible outputs of ciphertexts! Hence, if the original RSA encryption scheme is used, an attacker can trivially guess the plaintext from the ciphertext when the plaintext space is small.

The RSA encryption scheme is used in many practical applications. The actual implementation used is normally a variant known as RSA-OAEP [10, 82], which is no longer homomorphic and thus cannot be used in a FaCT protocol to prevent framing. Some of the earlier FaCT protocols, such as the protocol proposed in [94], suggested the use of RSA for homomorphic encryption. These protocols are thus exposed to the deterministic nature of the original RSA scheme, which in the context of a FaCT protocol may not be desirable. In this case alternative homomorphic encryption schemes should be used, such as the Goldwasser-Micali [56], Paillier [99], El-Gamal [43] and Okamoto-Uchiyama [97] schemes. These encryption schemes are *probabilistic* in the sense that every time an identical message is encrypted, the resulting ciphertext is different with high probability. The basic idea is to use random strings to randomise the encryption process, as discussed in the following Goldwasser-Micali and Paillier encryption schemes.

Goldwasser-Micali Encryption Scheme. Goldwasser and Micali proposed the first semantically secure encryption scheme in 1984 [56]. The Goldwasser-Micali scheme achieves semantic security by randomising the encryption of the plaintext with a random integer r_1 (or r_2), as shown in Figure 2.4. Thus a plaintext results in different ciphertexts when it is encrypted using a different random integer r. It is not difficult to observe that the scheme is inefficient in terms of the size of the

1. Let $m = pq$ where p and q are two large distinct primes.
2. Let $\phi(m) = (p-1)(q-1)$.
3. Choose a random integer a, where $1 < a < \phi(m)$ and $\gcd(a, \phi(m)) = 1$, where gcd denotes greatest common divisor.
4. Compute $ab \equiv 1 \pmod{\phi(m)}$ where $1 < b < \phi(m)$.
5. The public encryption key *hek* is (m, a).
6. The private decryption key *hdk* is b.
7. Let messages X_1, X_2 and encrypted messages Y_1, Y_2, where $X_1, X_2, Y_1, Y_2 \in \mathbb{Z}_m$, given $\mathbb{Z}_m$ a group of integers modulo m.
8. Encryption of messages X_1 (or X_2) is:
$$Y_1 \leftarrow [X_1]_{HE_{hek}} \Longleftrightarrow Y_1 = {X_1}^a \bmod m. \tag{2.9}$$
9. Decryption of encrypted messages Y_1 (or Y_2) is:
$$X_1 \leftarrow [Y_1]_{HD_{hdk}} \Longleftrightarrow X_1 = {Y_1}^b \bmod m. \tag{2.10}$$
10. Homomorphic encryption is:
$$Y_1 \cdot Y_2 = {X_1}^a \cdot {X_2}^a \bmod m = (X_1 \cdot X_2)^a \bmod m. \tag{2.11}$$
In this case the homomorphic operator $\circ = \cdot$ (as defined in Definition 2.4) is modular multiplication. This means RSA demonstrates a *multiplicative homomorphic property.*

Figure 2.3: RSA Encryption Scheme with privacy homomorphism

ciphertext. As can be seen from the encryption operation, the original message is one bit, but the resulting ciphertext is much larger, having the size of m, where $m = 768$ or $m = 1024$ bits is currently the recommended bit-length for security assurance of a small or large organisation [51]. However, the scheme can be made more computationally efficient. As is mentioned in [124], if the two large distinct primes are generated such that $p \equiv 3 \bmod 4$ and $q \equiv 3 \bmod 4$ (these are known as *Blum integers*), then the integer g can be -1 ($g = -1$). In this case the computation of g^X, where X is the message, will not involve any exponentiation.

Paillier Homomorphic Encryption Scheme. This semantically secure encryption scheme was proposed by Paillier in 1999 [99]. We describe the scheme in Figure 2.5. The Paillier homomorphic encryption scheme has been used widely in digital voting schemes [61] and more recently in the field of watermarking in the encrypted

1. Similar to RSA, let $m = pq$ where p and q are two large distinct primes.
2. Randomly choose an integer $g \in \mathbb{Z}_m^*$, where g is a *quadratic non-residue* modulo m, and $\mathbb{Z}_m^*$ a multiplicative group of integers modulo m. (An integer y is a quadratic residue modulo m if there exists an integer $z \in \mathbb{Z}_m^*$ such that $y = z^2 \bmod m$. A quadratic non-residue means otherwise.) In this case, we can find a quadratic non-residue g if g satisfies

$$\left(\frac{g}{p}\right) = \left(\frac{g}{q}\right) = -1,$$

   where $\left(\frac{g}{p}\right) = g^{(p-1)/2} \bmod p$ and $\left(\frac{g}{q}\right) = g^{(q-1)/2} \bmod q$.
3. The keys are: $hek = (m, g)$ and $hdk = (p, q)$.
4. Let messages $X_1, X_2 \in \{0, 1\}$ and encrypted messages $Y_1, Y_2 \in \mathbb{Z}_m^*$.
5. Randomly choose an integer r_1 (or r_2) $\in \mathbb{Z}_m^*$.
6. Encryption of message X_1 with r_1 (or X_2 with r_2) is:

$$Y_1 \leftarrow [X_1]_{HE_{hek}} \Longleftrightarrow Y_1 = g^{X_1} \cdot r_1^2 \bmod m. \tag{2.12}$$

7. Decryption of message Y_1 (or Y_2) is:

$$X_1 \leftarrow [Y_1]_{HD_{hdk}} \Longleftrightarrow X_1 = \begin{cases} 0 & \text{if} \quad Y_1 = \left(\frac{Y_1}{p}\right) = \left(\frac{Y_1}{q}\right) = 1 \ ; \\ 1 & \text{if} \quad Y_1 = \left(\frac{Y_1}{p}\right) = \left(\frac{Y_1}{q}\right) = -1. \end{cases} \tag{2.13}$$

8. Homomorphic encryption is:

$$Y_1 \cdot Y_2 = g^{X_1} \cdot g^{X_2} \cdot r_1^2 \cdot r_2^2 \bmod m = g^{X_1 \oplus X_2} \cdot (r_1 \cdot r_2)^2 \bmod m. \tag{2.14}$$

   This means $[X_1]_{HE_{hek}} \cdot [X_2]_{HE_{hek}} = [X_1 \oplus X_2]_{HE_{hek}}$, where $\oplus$ means bit-wise XOR (i.e. $0 \oplus 0 = 1 \oplus 1 = 0$ and $0 \oplus 1 = 1 \oplus 0 = 1$). Decryption of $Y_1 \cdot Y_2$ will result in $X_1 \oplus X_2$. In this case, as defined in Definition 2.4, the homomorphic operator $\circ$ is modular multiplication for multiplying Y_1 and Y_2 ($\circ = \cdot$) and is bit-wise XOR for adding X_1 and X_2 ($\circ = \oplus$).

Figure 2.4: Goldwasser-Micali Encryption Scheme

domain [44]. As was noted in [99], the Paillier scheme has similar computational efficiency as RSA, since both are based on modular exponentiation.

1. Similar to RSA, let $m = pq$ where p and q are two large distinct primes.
2. Let $\lambda = \text{lcm}(p-1, q-1)$, where lcm is the least common multiple.
3. Randomly choose an integer $g \in \mathbb{Z}^*_{m^2}$, where $\mathbb{Z}^*_{m^2}$ is a multiplicative group of integers modulo m^2. Ensure g has order k that is a multiple of m (i.e. m divides k) by checking

$$\gcd(L(g^\lambda \bmod m^2), m) = 1,$$

   where gcd denotes greatest common divisor and $L(u) = \frac{u-1}{m}$.
4. The keys are: $hek = (m, g)$ and $hdk = \lambda$.
5. Let messages $X_1, X_2 \in \mathbb{Z}_m$ and encrypted messages $Y_1, Y_2 \in \mathbb{Z}^*_{m^2}$.
6. Randomly choose an integer r_1 (or r_2) $\in \mathbb{Z}^*_m$.
7. Encryption of message X_1 with r_1 (or X_2 with r_2) is:

$$Y_1 \leftarrow [X_1]_{HE_{hek}} \Longleftrightarrow Y_1 = g^{X_1} \cdot r_1^m \bmod m^2. \tag{2.15}$$

8. Decryption of message Y_1 (or Y_2) is:

$$X_1 \leftarrow [Y_1]_{HD_{hdk}} \Longleftrightarrow X_1 = \frac{L({Y_1}^\lambda \bmod m^2)}{L(g^\lambda \bmod m^2)} \bmod m. \tag{2.16}$$

9. Homomorphic encryption is:

$$Y_1 \cdot Y_2 = g^{X_1} \cdot g^{X_2} \cdot r_1^m \cdot r_2^m \bmod m^2 = g^{X_1+X_2} \cdot (r_1 \cdot r_2)^m \bmod m^2. \tag{2.17}$$

   This means $[X_1]_{HE_{hek}} \cdot [X_2]_{HE_{hek}} = [X_1 + X_2]_{HE_{hek}}$. Decryption of $Y_1 \cdot Y_2$ will result in $X_1 + X_2$. In this case, as defined in Definition 2.4, the homomorphic operator $\circ$ is modular multiplication for multiplying Y_1 and Y_2 ($\circ = \cdot$) and is modular addition for adding X_1 and X_2 ($\circ = +$). This means Paillier demonstrates an *additive homomorphic property.*

Figure 2.5: Paillier Homomorphic Encryption Scheme

2.3.3 Watermarking in the Encrypted Domain

In this section we examine how digital watermarking schemes and homomorphic encryption schemes can be integrated to achieve a technique known as *watermarking in the encrypted domain* [41, 44, 108, 120]. This technique embeds a watermark into content while both the watermark and content are in encrypted form. This is useful when the party that performs the watermark embedding process should not have access to the watermark. Taking the Paillier homomorphic encryption scheme and SS watermarking scheme as examples, we show the working of watermarking in the

encrypted domain in Figure 2.6.

1. Let content $X = (x_1, \ldots, x_n)$ and watermark $W = (w_1, \ldots, w_n)$, where both x_i, $w_i \in \mathbb{R}$, and n is the number of the most significant elements in the content to be watermarked.
2. Let the SS embedding algorithm be:
$$x'_i = x_i + \rho w_i \qquad 1 \leq i \leq n, \tag{2.18}$$
identical to the SS algorithm presented in Figure 2.1.
3. Let $[.]_{HE_{hek}}$ be the encryption algorithm of the Pailler encryption scheme with public encryption key *hek*, identical to the algorithm presented in Figure 2.5.
4. Watermarking in the encrypted domain is performed as:
$$[x_i]_{HE_{hek}} \cdot [\rho w_i]_{HE_{hek}} = [x_i + \rho w_i]_{HE_{hek}} = [x'_i]_{HE_{hek}} \qquad 1 \leq i \leq n. \tag{2.19}$$
5. Hence $[X']_{HE_{hek}} = ([x'_1]_{HE_{hek}}, [x'_2]_{HE_{hek}}, \ldots, [x'_n]_{HE_{hek}})$.
6. Note that the modulo operator of the encryption algorithm only allows computation of integers, but X, ρ and W are all based on real numbers. This issue can be addressed by representing a real value as an integer by scaling. As an illustrative example, 15.687 can be represented as 15687 or 1568700, depending on the requirement of the underlying application [1].

Figure 2.6: Watermarking in the Encrypted Domain: Paillier and Spread Spectrum

Security for watermarking in the encrypted domain depends on the security of the underlying homomorphic encryption scheme and the digital watermarking scheme. One issue, which is due to the privacy homomorphism property (Definition 2.4), is that given an encrypted marked content of a client C, $[X']_{HE_{hek_C}}$, an attacker can modify this encrypted marked content by multiplying it by another ciphertext of his choosing $[X_a]_{HE_{hek_C}}$:

$$[X']_{HE_{hek_C}} \circ [X_a]_{HE_{hek_C}} = [X' \circ X_a]_{HE_{hek_C}},$$

thus modifying the marked content from X' to $X' \circ X_a$. In FaCT protocols, where a distributor sends the encrypted marked content to a client, this issue can be addressed by the distributor signing the encrypted marked content so that the recipient of this content can verify that it has not been modified during transmission. Alternatively, it is assumed the distributor and the client communicate through a secure channel.

As for computational efficiency, it depends on the size of content n and the computational efficiency of the underlying homomorphic encryption scheme. We observe in Figure 2.6 that $2n$ asymmetric homomorphic encryptions are required for encrypting the content and the watermark, and n modular multiplications for embedding the watermark into content.

2.3.4 Cryptographic Hash Functions

A *cryptographic hash function* is used to protect data integrity [124]. This means that it allows a party to check whether a message has been changed since the message was created. Given a cryptographic hash function $H(.)$ and a message X, computing the function results in a *hash value* $H(X)$. This hash value serves as an identifier that links to the message X. The message X can be of arbitrary length but the hash value $H(X)$ normally has much smaller, fixed length. A common length of the hash value is 160bits.

Suppose that $H(X)$ is securely stored, but the message X is publicly accessible. If someone changes X to X', the party who originally created message X can detect that X has been altered by computing the hash function on X', resulting in $H(X')$, and noting that $H(X) \neq H(X')$. Two well-known hash function are SHA-2 [70] and RIPEMD-160 [37]. We formalise a cryptographic hash function in Definition 2.5.

Definition 2.5 ([124]) *A* cryptographic hash function $H(.)$ *is a polynomial-time algorithm that on input a message X of arbitrary length, outputs a hash value $H(X)$ of fixed length. In addition $H(.)$ has the following three properties:*

- *$H(X)$ is pre-image resistant. This means given $H(X)$, it is computationally infeasible to find X.*
- *$H(X)$ is second pre-image resistant. This means given X, it is computationally infeasible to find a message $X' \neq X$ such that $H(X') = H(X)$.*
- *$H(X)$ is collision resistant. This means it is computationally infeasible to find* any *two different messages $X' \neq X$ such that $H(X') = H(X)$.*

2.3.5 Digital Signature Schemes

Handwritten signatures have been widely used to prove the validity of documents (such as letters and contracts). Signatures on these documents serve as evidence

that the party who signed them agrees with the terms and conditions listed in these documents. Digital signatures can be thought of as electronic versions of handwritten signatures with broadly similar aims. More precisely, they provide data origin authentication and non-repudiation of a message. A digital signature scheme consists of a *key generation algorithm*, a *signing algorithm* and a *signature verification algorithm*. We formalise a digital signature scheme in Definition 2.6.

Definition 2.6 ([124]) *A* digital signature scheme *is a triple* $(\mathcal{G}_s, [\cdot]_{SIG_{(\cdot)}}, [\cdot,\cdot]_{VER_{(\cdot)}})$ *of polynomial-time algorithms:*

- *On input* 1^k*, where* k *is the security parameter, the key generation algorithm* $\mathcal{G}_s$ *outputs a key pair (pvk, ssk).*
- *On input of a message* X*, the signature generation algorithm* $[X]_{SIG_{ssk}}$ *with the private signing key ssk outputs a signature* σ*. This can be represented as* $\sigma \leftarrow [X]_{SIG_{ssk}}$*. The signed message is* (X, σ)*.*
- *On input of the signed message* (X, σ)*, the verification algorithm* $[X, \sigma]_{VER_{pvk}}$ *with the public verification key pvk outputs* `true` *if the verification is successful, otherwise it outputs* `false`*. This can be represented as* $[X, \sigma]_{VER_{pvk}} \in \{\texttt{true}, \texttt{false}\}$*.*

We require that for all $\sigma \in \{[X]_{SIG_{ssk}}\}$*:*

$$[X, \sigma]_{VER_{pvk}} = \texttt{true}.$$

The standard security notion for digital signature schemes is *unforgeability* [58]. This means that an efficient (polynomial-time) attacker who can repeatedly obtain a signature on a message of his choice will not be able to generate a signature on a newly created message. Hence digital signatures used in FaCT protocols are always assumed to be *unforgeable*. The exact notion of *unforgeability* can be found in [58].

In terms of efficiency, instead of signing the message directly using the signature generation algorithm, it is common practice to sign the hash value of the message, which is normally much shorter. In our subsequent discussion we assume that this is the case, although for brevity we only illustrate the direct signing of a message, unless explicitly stated otherwise.

We describe the RSA signature scheme as an example. Other digital signature schemes include the El-Gamal scheme [43], elliptic curve scheme ECDSA [71], and ID-based schemes such as [100]. All these signature schemes can be used in a FaCT protocol. For example, if better computation and storage performance are required, ECDSA can be used since elliptic curve schemes are known to have much smaller

key size than the RSA scheme. If it is preferred that a signature scheme where the public key is a recognisable text (such as an email address) instead of a random string, then ID-based schemes can be used. We choose to describe RSA scheme since it is the most well-known and widely deployed scheme.

RSA Signature Scheme. This was proposed by Rivest, Shamir and Adleman in [117] and is similar to the RSA encryption scheme illustrated in Figure 2.3. We describe the RSA signature scheme in Figure 2.7. We remark that this is the originally proposed basic scheme and, in practice, a scheme such as RSA-PSS [83] should be used.

1. Let $m = pq$ where p and q are two large distinct primes.
2. Let $\phi(m) = (p-1)(q-1)$.
3. Choose a random integer a, where $1 < a < \phi(m)$ and $\gcd(a, \phi(m)) = 1$, where gcd denotes the greatest common divisor.
4. Compute $ab \equiv 1 \pmod{\phi(m)}$, where $1 < b < \phi(m)$.
5. The public verification key *pvk* is (m, a).
6. The private signing key *ssk* is b.
7. Let message X and signature σ, where $X, \sigma \in \mathbb{Z}_m$, given $\mathbb{Z}_m$ is a group of integers modulo m.
8. Signing of message X is:
$$\sigma \leftarrow [X]_{SIG_{ssk}} \iff \sigma = X^b \bmod m. \tag{2.20}$$
9. Verification of signature σ is:
$$[X, \sigma]_{VER_{pvk}} = \texttt{true} \iff X = \sigma^a \bmod m. \tag{2.21}$$

Figure 2.7: RSA Signature Scheme

2.3.6 Zero-Knowledge Proofs

These were first introduced by Goldwasser, Micali and Rackoff [57]. This mechanism allows one party to *prove* to another party that a statement is true without revealing any information in the statement [55]. A common application is a *zero-knowledge proof of knowledge*, where a party C proves to a party D that C knows a certain secret without revealing the secret itself. We describe one proof mechanism that is used in the FaCT protocol proposed by Pfitzmann and Schunter [104]. This technique is the

minimum disclosure proof of knowledge proposed by Brassard, Chaum and Crepeau (BCC) in [17], which is based on schemes known as bit commitment schemes, also introduced in the same proposal.

As stated in [17], a *bit commitment scheme* is a scheme that allows a party C to commit himself to some bits in such a way that another party D, who is given the commitments of these bits, cannot know what these bits are without the help of C. We present one of the proposed bit commitment schemes. This scheme is based on the Goldwasser-Micali homomorphic encryption scheme (Figure 2.4) and is described in Figure 2.8. Other implementations, such as schemes based on factoring and discrete logarithms can be found in [17].

1. Following the setting of the Goldwasser-Micali encryption scheme illustrated in Figure 2.4:
   - Let $m = pq$ where p and q are two large distinct primes.
   - Let $g \in \mathbb{Z}_m^*$, where g is a quadratic non-residue.
   - Let $X_1 \in \{0, 1\}$, $Y_1 \in \mathbb{Z}_m^*$,
   - Let $hek = (m, g)$, $hdk = (p, q)$, and
   - $r_1 \in \mathbb{Z}_m^*$.
2. *Committing* a bit $[X_1]_{COM_{hek}}$ is identical to encryption in Goldwasser-Micali encryption scheme:
$$Y_1 \leftarrow [X_1]_{COM_{hek}} \Longleftrightarrow Y_1 = g^{X_1} \cdot r_1^2 \bmod m. \tag{2.22}$$
3. To *open* a commitment Y_1, the party that generates the commitment Y_1 reveals r_1, and verification $[X_1, Y_1]_{VCOM}$ of the commitment is as follow:
$$[X_1, Y_1]_{VCOM} = \begin{cases} 0 & \text{if } Y_1 \leftarrow [0]_{COM_{hek}} \ ; \\ 1 & \text{if } Y_1 \leftarrow [1]_{COM_{hek}} \ ; \\ \bullet & \text{Otherwise,} \end{cases} \tag{2.23}$$
   where $\bullet$ means *undefined*. In other words, Y_1 is irrelevant to $X_1 = 0$ or $X_1 = 1$.

Figure 2.8: BCC Homomorphic Bit Commitment Scheme based on Goldwasser-Micali [17]

We will discuss how this scheme is used to provide a zero-knowledge proof of knowledges in the description of the Pfitzmann and Schunter protocol in the next chapter. In addition, as can be observed from the description in Figure 2.8, this bit commitment scheme has the identical homomorphic property of the Goldwasser-Micali encryption scheme, due to its commitment algorithm being identical to that of the Goldwasser-Micali encryption algorithm. Hence it can also be used for embedding

a watermark into content in the encrypted domain.

2.4 Summary

In this chapter, we first discussed how digital content can be easily and effectively distributed due to the proliferation of increasingly powerful computing devices. We illustrated three common content distribution models. We then discussed concerns relating to illegal distribution of digital content in these models. We illustrated how content tracing can be used to address these concerns. We further raised the issue of framing and denial caused by content tracing, where the distributor has the ability to frame an honest client, while at the same time is unable to prove illegal content distribution by a dishonest client. These issues motivated *fair content tracing*. We introduced *fair content tracing (FaCT) protocols* as a means to address these issues. Finally we briefly introduced some of the watermarking and cryptographic building blocks required to realise these protocols.

Chapter 3

A Design Framework for FaCT Protocols

Contents

This chapter presents a design framework for FaCT protocols. We discuss the design issues in existing protocols that motivate the proposal of a general framework. We define the components of this framework and demonstrate how the framework provides a means to address the design issues, and to analyse and classify existing FaCT protocols. Using the framework, we examine one of the earliest FaCT protocols as an example.

3.1 Why A Design Framework?

There is no obvious way to analyse the various protocols proposed to date, where different design notions and approaches are used. The existing protocols use different terms and different interpretations of security requirements in their construction. This is especially true in the BSW protocols that we briefly discussed in Section 2.2. More importantly, in many cases there are no assumptions, threat model or appropriate security requirements defined for many of these protocols.

Without a framework, protocols that build on earlier protocols tend to become more complicated and prone to error. Newer proposals introduce new features (or "fixes") on top of the existing ones, or propose an alternative approach in an ad hoc way. These are exemplified by the many protocols proposed since Memon and Wong [94], and Pfitzmann and Schunter [104], as previously discussed in Section 2.2. The result is that additional features are not always added with a clear definition of purpose. Furthermore, a proper analysis of security is often lacking. As a consequence, these ad hoc designs are hard to analyse and some even contain flaws, as we will show in Chapter 4 and Chapter 5.

A general design framework with proper notions and definitions is necessary in order to design FaCT protocols in a systematic manner. In the next section we outline such a framework.

3.2 Overview of the Framework

The proposed design framework consists of two components—*fundamentals* and *environment*—that provides general guidelines on the construction of FaCT protocols. It is used as a basis to examine existing protocols and design new ones, and to identify the different classes of FaCT protocols. We first overview the two main components:

- **Fundamentals.** This aims to provide a clear view on the *objectives (or goals)* of a FaCT protocol. It consists of the *parties involved*, *threats*, *security requirements* and *phases* of a FaCT protocol.
  - The *parties involved* identifies the roles of the many players in a FaCT protocol. It also defines the trust assumptions between these players.
  - The *threats* establish a basic threat model that identifies the main threats faced by the players in a FaCT protocol.
  - The *security requirements* identify precisely what a FaCT protocol should achieve.
  - The *phases* categorise the protocol steps of a FaCT protocol into communication *before*, *during* and *after* content distribution between the parties involved.
- **Environment.** This aims to identify the elements that are available for constructing a FaCT protocol. It consists of *computing resources*, *trust infrastructure* and *building blocks*.
  - The *computing resources* identify the resources that the distributor and the client possess for processing, viewing and sharing of content. For example, the distributor may have more powerful computing devices than the client. The resource available can be used to determine whether limitations must be put in place.
  - The *trust infrastructure* identifies whether trusted third parties or any trusted platforms can be used in the construction of a FaCT protocol.
  - The *building blocks* identify the underlying components (Section 2.3) that can be used to construct a FaCT protocol.

Figure 3.1 illustrates the framework. A preliminary study of this framework was presented in [109].

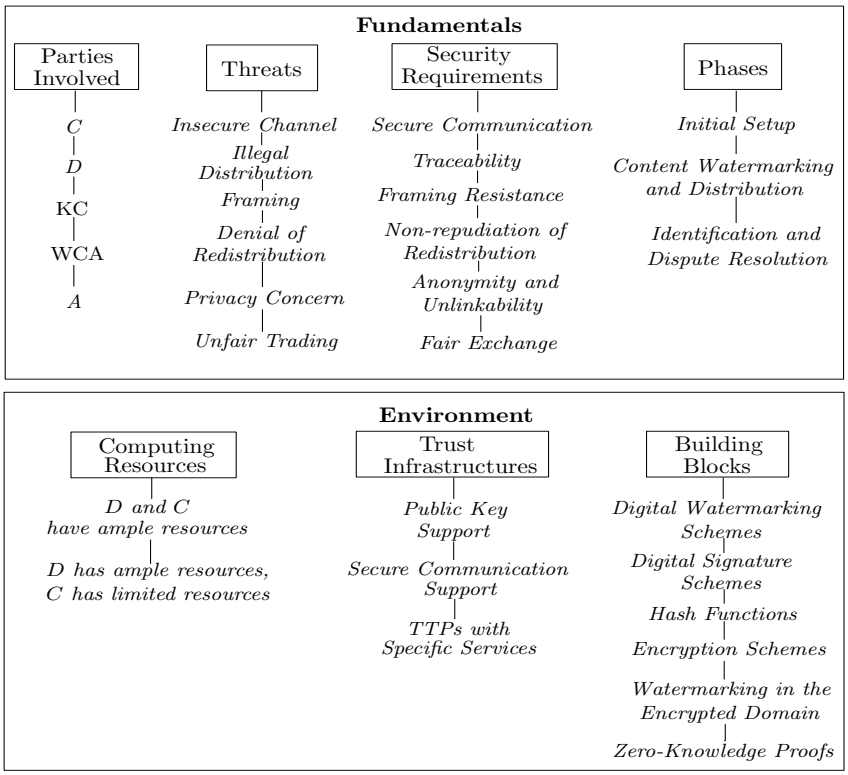

Figure 3.1: A General Framework

3.3 Fundamentals

These aim to provide a proper definition of the objectives of a FaCT protocol.

3.3.1 Parties Involved

The two main parties are the client C and the distributor D:

- The *distributor* D is a service provider who distributes (or sells) digital content to the client C.
- The *client* C is a user of digital content who receives (or purchases) digital content from the distributor D.

There are also potentially one or more trusted third parties (TTPs):

- A *key centre* KC, generates and distributes key materials. For example, in the Pfitzmann-Schunter protocol that we discuss in Section 4.2, the KC takes the form of a certificate authority CA and is required to certify the public keys of C.
- A *watermark certification authority* WCA, generates client watermarks, such as in the Memon-Wong protocol that we discuss in Section 3.7.
- An *arbiter* A, attempts to resolve disputes between C and D.

Trust Assumptions. We now define the trust relationships between these parties:

- It is assumed that C and D *do not trust each other*. By this we mean that it is possible that C and D may try to gain unfair advantage against one another. For example, in our context, a dishonest C gains advantage against D when he successfully redistributes a copy of content illegally without being traced by D.
- The arbiter A is *fully trusted*. This means that C, D, and other parties involved, such as the WCA, trust A to behave honestly. This also means that A will not collude with D or any other parties to falsely accuse an innocent client of illegal content distribution. Similarly, A will not collude with C or any other parties so that a dishonest C can freely redistribute copies of content.
- The key centre KC is *fully trusted*. This means that signatures generated by the KC are always viewed as valid when verification of these signatures based on the KC's public verification key is successful.
- The watermark certification authority WCA is *fully trusted*. This means C, D, A, and other parties involved, such as the KC, trust the WCA to behave honestly. This also means that the WCA will not collude with D to frame an innocent client, or collude with a dishonest C in such a way that C can freely redistribute copies of content.

We remark that it is important to explicitly state the trust relationships between the parties involved since security issues may arise if these are ambiguous, as in the

case of an attack known as the conspiracy problem that we discuss in Chapter 4. Also, as stated above, collusion with trusted third parties does not occur. This reasonably reflects our common practical relationships with trusted parties, e.g. where banks, and legal authorities do not typically conspire with customers. Conspiracies involving trusted parties would render many security applications unworkable.

We further note that the stated trust assumptions do not represent an exhaustive list. Depending on the design of a FaCT protocol, there may be other parties involved with different associated trust assumptions. What is crucial is that these trust assumptions are well defined.

3.3.2 Threats

We define three threat scenarios for FaCT protocols.

Threats to the communication channel captures the threats *during the communication between two parties*. We follow the standard Dolev-Yao threat model [38], since this model is well-established. In this model an attacker has the following abilities:

- The attacker is in possession of all the messages passing through the communication channel between the parties involved.
- The attacker can be a legitimate party of the protocol, and thus can initiate communication with any other parties involved.
- The attacker can redirect messages in the communication channel, hence he can be a recipient of messages not intended for him, and he can impersonate another party in order to send message to other parties.

In other words, the attacker has complete control of the communication channel. For example, a dishonest C_a (or a dishonest D_a) can listen to, and try to modify, the communication between an honest C and an honest D. The threat model also assumes that the underlying building blocks applied in the construction of a FaCT protocol are secure. For example, the adversary cannot compute a private signing key from a digital signature.

Security against the threats to the communication channel is very important since, if the communication channel is insecure, it is possible for outsiders, or any of the legitimate clients and distributors, to launch standard protocol attacks such as replaying the messages or impersonating other parties, as described in [16]. This also means that any dishonest clients or distributors can use such a weakness to defeat fair content tracing.

Threats due to content tracing captures the specific issues that a FaCT protocol intends to address, as discussed in Section 2.1.3. This captures the specific content distribution threats that occur *after the communication between two parties has ended successfully.* The threats are:

- *Dishonest clients.* A dishonest C may redistribute copies of content either for monetary gain or for sharing with others, and will deny this fact when confronted by D.
- *Dishonest distributors.* An unscrupulous D may frame C by falsely accusing C of illegal content distribution.

Threats to client privacy and fair trading captures the threats that occur when privacy is a concern and/or payment is involved. We remark that *payment*, in a broad sense, means the information requested by D in order for D to agree to provide content to C. In most circumstances, this means monetary gain, hence the term *payment*.

- *Privacy concern: dishonest distributors.* D gathers information about C, such as for marketing purposes, even when C prefers to get content from D without revealing his real identity.
- *When payment is involved:*
  - *Dishonest clients.* A dishonest C can choose not to follow the protocol honestly, so as to gain advantage over D, such as running away with content without paying (or without providing the information requested by D).
  - *Dishonest distributors.* A dishonest D may refuse to provide content (or provide corrupted content) to C after payment has been received.

3.3.3 Security Requirements

We now define the security requirements of a FaCT protocol.

1. *Secure Communication.* A communication channel between two parties is secure if:
   - *data secrecy (or confidentiality)* is preserved when needed,
   - the parties involved can be *authenticated*,
   - and *data integrity* can be checked.

This addresses the threats on the communication channel. Following this, there are three specific security requirements for FaCT protocols:

2. *Traceability.* A legitimate, but dishonest C who illegally distributes content can be traced to their identity by D.
3. *Framing Resistance.* An honest C cannot be falsely accused of illegal distribution by D.
4. *Non-repudiation of Redistribution.* A dishonest C who illegally distributes copies of content cannot deny such an act. This means that D is able to prove the illegal act of C to a third party. *Framing resistance* is a prerequisite since, without it, C can claim that it is D who redistributed the content.

A FaCT protocol is a *weak FaCT protocol* if it fulfills only traceability and framing resistance, while it is a *strong FaCT protocol* if it fulfills all three requirements. These three requirements address the threats due to content tracing. We note that the protocols discussed in this thesis are strong FaCT protocols unless we state otherwise (e.g. the DAA protocol in Section 7.3.2). The next two requirements are optional:

5. *Anonymity and Unlinkability.* In the case where client privacy is a concern, C can obtain content anonymously from D. Furthermore, given any two contents, D cannot tell whether they are from the same C.

6. *Fair Exchange.* In the case where there is *payment* involved, either both parties (D and C) are satisfied, or no party gains advantage over the other. For example, in our context, C does not receive content if D does not receive payment (see Chapter 8 for further discussion).

Table 3.1 summarises these security requirements. We note that except for the first and the last requirements, all other requirements have been previously mentioned (in various forms and combinations) in [19, 25, 54, 74, 78, 85, 102, 105]. Consolidating these security requirements allows for FaCT protocols to be analysed more meaningfully.

Table 3.1: Issues and Requirements

Issues	**Requirements**
Insecure channel	Secure communication
Illegal distribution	Traceability (deterrence using digital watermarking)
Framing	Framing resistance
Denial of redistribution	Non-repudiation of redistribution
Privacy concern	Anonymity and unlinkability
Fair distribution (when payment is involved)	Fair exchange

3.3.4 The Three Phases

This section describes the three main phases of a FaCT protocol. It is assumed that all phases are conducted under secure communication support, which we discuss in Section 3.4.2, unless it is explicitly stated otherwise.

Phase 1: `Initial Setup`. In this phase C and D generate key materials. The KC verifies and signs the public keys of C and D. Hence the main aim of this phase is to provide key materials for C and D to support the required cryptographic services. The signed public keys can then be publicly distributed to other parties who need to use them. We assume that the activities in this offline phase are carried out *before* content distribution. We also assume that prior to this the major trusted third parties, such as the KC, WCA and A, have obtained their respective authenticated public keys based on a public key support, which we discuss in Section 3.4.2.

Phase 2: `Content Watermarking and Distribution`. This is the phase where content distribution is carried out. It is an online process. In this phase C requests

content from D, and D sends encrypted marked content to C. Provision of *traceability*, *framing resistance* and *non-repudiation of redistribution* takes place during this phase using the authenticated keys obtained in the initial setup. The objective of this phase is to provide C with the requested content and at the same time provide all the main security requirements of a FaCT protocol.

Phase 3: `Identification and Dispute Resolution`. In this phase D identifies C from a found copy of content. If necessary, D proves to A that C illegally distributed content by showing A some evidence. This evidence normally consists of at least the client watermark and the client's signature on an agreement describing the content. Most FaCT protocols require that A obtains the client watermark from another party (e.g. C or a trusted third party).

3.4 Environment

In this section we describe the second component of our framework. The main purpose of this component is to provide the design "modules" (or parameters) that can be chosen and then assembled to construct a FaCT protocol.

3.4.1 Computing Resources

This refers to the computing power and storage available to the parties involved. We categorise the computing resources, notably for D and C, based on the following scenarios:

- *The distributor and the client have ample resources.* In this scenario both D and C are assumed to be capable of performing resource intensive computations (e.g. computing many public-key operations). It is also assumed that both D and C have ample storage. We can further divide this into two different settings:
  - *A distributor acts as a central server.* In this case we will expect D to have significantly more resources than C, so that D has the capacity to provide services to many clients at one time. For example, the content broadcast

and buyer-seller content purchase models discussed in Section 2.1 are such scenarios, where iTune Store [68] is an example.

– *A client that also acts as a distributor.* In this case both C and D are assumed to have similar computing resources, but typically less than the distributor in the central server setting above. This scenario applies to the peer-to-peer model described in Section 2.1, where Gnutella [52] is an example.

- *The distributor has ample resources, while the client has limited resources.* In this setting D has no problem performing resource intensive computation but C has limited computing resource and storage capability. In this scenario, C may not be able to compute many public-key operations. This scenario applies to applications where C is a resource-limited portable device, such as a mobile phone.

The computing resources available to D and C influence the choice of building blocks for a FaCT protocol. In the scenario where C has limited resources then resource intensive mechanisms such as watermarking in the encrypted domain based on asymmetric homomorphic encryption schemes (Section 2.3.3) may not be suitable. Efficient methods potentially suitable for this scenario were proposed recently [81, 110, 137], as discussed in Chapters 5 and 6. However we note that most existing FaCT protocols implicitly assume that both C and D have ample resources. All the new protocols proposed in this thesis are also for this scenario.

3.4.2 Trust Infrastructures

These are infrastructures that can be thought of as central points of contact for providing security services. They represent a crucial element of any FaCT protocol. For example, in Internet banking or e-shopping applications, there must be a certain trust infrastructure to ensure the authenticity of the banking or shopping website as well as the authenticity of the user. The trust infrastructure in this case is commonly provided by a PKI supporting the use of digital certificates. These certificates are generated by a trusted party known as a certificate authority CA. A FaCT protocol may involve the following trust infrastructures:

Public Key Support. The security requirements stated in Section 3.3.3, notably *non-repudiation of redistribution*, requires the generation of non-repudiable proofs (e.g. a digital signature) so that it is possible to prove the guilt of a dishonest client. A standard way of providing this proof is by deploying a digital signature scheme, which we defined and gave examples of in Section 2.3.5. In addition, many existing protocols use asymmetric homomorphic encryption schemes (Section 2.3.2) as tools for providing framing resistance. Both digital signatures and homomorphic encryption schemes require the distribution of public verification and encryption keys prior to the content distribution process. These keys must be authenticated so that a party who uses these keys knows that they belong to the legitimate parties. To achieve this, we assume the existence of a PKI.

Secure Communication Support. In a FaCT protocol, before C requests content from D, C will want to know that he is indeed talking to D. Similarly, D will want to be sure that he is indeed communicating with C. In other words, C and D must authenticate each other.

One way to provide this is by using entity authentication and key establishment (AKE) protocols [8, 9, 13, 16]. These protocols are run between two parties to achieve the following goals:

- *Mutual Entity Authentication.* Each party gains confidence in the identity of their communication partner [9]. In content distribution and tracing environments this is particularly important, since it is reasonable to assume that one distributor can distribute content to many clients at the same time.
- *Key Establishment.* Distribution of a secret "session key" that is agreed between the two communicating parties, which can be used for protecting data secrecy, data integrity and other security services. One important property of the key is that it is distinct for each different protocol run between the two parties.

There are many different AKE protocols. Some have formal proofs of security [8, 9, 13], and some have been standardised [35, 73]. A FaCT protocol should deploy a well-established AKE protocol in order to fulfill the secure communication requirement,

which we defined in Section 3.3.3. We explicitly assume the existence of secure communication support in most of our protocol discussions, instead of the implicit assumption of many existing protocols [85, 94].

Trusted Third Parties (TTPs) with Specific Services. In addition, there may be extra trusted third parties required for specific services (see Section 3.3.1). Regardless of their functions, they can be classified as either:

- *Online TTPs.* These are TTPs that are involved *during* content distribution. In this case, D (or C) needs to contact the TTP to obtain information after C (or D) initiates execution of the protocol. Since it is possible for many clients (or many distributors) to request information from the distributor (or the client) at anytime, the TTP has to always be online. One example protocol where an online TTP is present is the Lei-Yu-Tsai-Chan protocol in [85], which we will describe in Section 5.2.

- *Offline TTPs.* These are TTPs that are involved *before* content distribution. In general, C (or D) contacts such a TTP to request information before continuing the execution of the protocol with D (or C). During the protocol run, the TTP is not involved. For example, in the Memon-Wong protocol that we discuss in Section 3.7, C contacts the WCA to obtain client watermarks before contacting D to obtain content.

- *Trusted Hardware.* This refers to specific hardware devices that provide the functionality of the online or offline TTPs described above. Such trusted hardware allows information to be generated in a trusted environment, which means that the information can only be generated and changed by authorised computer processes. The *Trusted Platform Module (TPM)* of the trusted computing initiative [95, 130] is one such device. The main characteristic of trusted hardware is that it is embedded as part of the computing platform itself. So instead of the need to contact a central TTP, trusted hardware within the computing platform can be used, making it most suitable for distributed computing environments. We will describe FaCT protocols that rely on a TPM in Chapter 7. We also remark that smart cards may also be deployed [91]. Although with a more restricted functionality compared to the TPM, their

present as a component of the computing platforms of C and D in a FaCT protocols offer at least a secure storage of private key materials.

The choice of type of TTP depends on the underlying applications. For example, one may choose to deploy a trusted hardware module if extra hardware cost is not an issue, since these are embedded into many devices and are more scalable for distributed environments. Similarly, one may choose an offline TTP if the underlying application dictates it most convenient for C to request information from the TTP before the actual execution of the protocol, keeping communication overheads during execution to a minimum.

3.4.3 Building Blocks

These are the technical means to fulfill the core security services needed by a FaCT protocol. These were presented in Section 2.3, and so in the following we only briefly identify the role of each of these building blocks:

- *Digital watermarking schemes* (Section 2.3.1) are used to provide *traceability*.
- *Digital signature schemes* (Section 2.3.5), which make use of *cryptographic hash functions* (Section 2.3.4), are used to provide *non-repudiation of redistribution*. Together with public key support, these are also used to provide *anonymity and unlinkability* and *fair exchange*.
- For some FaCT protocols, *homomorphic encryption schemes* (Section 2.3.2), together with *digital watermarking schemes*, are used to provide *watermarking in the encrypted domain* (Section 2.3.3). This is used to provide *framing resistance*.
- For some FaCT protocols, *zero-knowledge proofs* (Section 2.3.6) are used together with *watermarking in the encrypted domain* to provide *framing resistance*.

3.5 Classification

The aim of this section is to classify FaCT protocols into categories that facilitates analysis. This classification also provides a platform from which to explore new designs. We choose to use the presence of trusted third parties with specific services described in Section 3.4.2 as our major classification criterion. This is because the presence and role of trusted third parties significantly influences the underlying building blocks that can be deployed. More importantly, it raises the crucial issue of *who is to generate the clients' watermark*, which has security ramifications for all three of the specific security requirements of a FaCT protocol, as we shall see in Chapters 4, 5 and 6.

We identify four categories of protocols. These are *protocols without trusted third parties*, *protocols with online trusted third parties*, *protocols with offline trusted third parties* and *protocols with trusted hardware*.

We remark that the classification criterion is based on trusted third parties that provide watermark related information. It does not take into account trusted third parties that only provide public key support, as described in Section 3.4.2 (for example a KC that functions only as a CA) since all FaCT protocols require such a third party to provide services for entity authentication and non-repudiation.

We now describe the general constructions of protocols in the four categories. The main aim is to illustrate how these categories are designed differently from each other based on the parties involved and their protocol flows. We avoid protocol specifics and discuss this only in terms of the following common objects:

- $f(W)$. The object that contains the client watermark W. It plays a crucial role in providing framing resistance in many existing FaCT protocols. As an example, $f(W)$ can be an encrypted watermark $[W]_{HE_{hek_C}}$ produced using C's encryption key and an asymmetric homomorphic encryption scheme (see Section 3.7), or $f(W)$ can be a decryption key that contains the client watermark (see Section 6.3).
- $f(\widetilde{X})$. The object that contains the marked content $\widetilde{X}$. It ensures traceability and framing resistance. For example, $f(\widetilde{X})$ can be an encrypted marked content $[X'']_{HE_{hek_C}}$ produced using C's encryption key and an asymmetric homo-

morphic encryption scheme, or an encrypted marked content $[X']_{E_K}$ produced using a secret key K and a symmetric encryption scheme (see Sections 3.7 and 6.2).

- `SIG`. Digital signatures of objects such as the client watermark. These signatures provide data origin authentication and non-repudiation of redistribution. For example, `SIG` can be a signature $[AGR]_{SIG_{ssk^*}}$ on a content agreement AGR signed by C using key ssk^*, or a digital certificate $Cert_{ssk_{CA}}(pvk^*_C, hek^*_C)$ containing public keys and identity information of C (see Section 5.2).

- AGR. A content agreement that contains a description of content and licensing terms for using the content. When AGR is signed together with $f(W)$ by C, the resulting signature shows that C agrees to the terms stipulated in the agreement. Furthermore, it binds $f(W)$ and AGR together so that it is not possible for any party to place an extracted watermark from a found copy into different content. We will look at this issue in the next chapter (Section 4.3).

We further introduce a general term *info*, which may contain objects such as a content agreement AGR, a signature `SIG` and other necessary information, depending on the specific FaCT protocol. We also use the notation $\{\cdot\}_{AKE}$ to mean that the message is transmitted under secure communication support.

3.5.1 Category 1: Protocols without Trusted Third Parties

Our first category is a family of FaCT protocols that do not use the WCA. Their main common characteristic is that C generates the watermark, and this watermark is embedded into content by D in such a way that D has no knowledge of the watermark. Allowing C to generate the watermark means that extra security measures must be put in place to prevent C from generating an ill-formed watermark. A watermark is said to be *well-formed* if it is a pseudo-random sequence of real numbers. Otherwise it is *ill-formed*. For example, a common ill-formed watermark is a string of zero, $W = (0, 0, \ldots, 0)$. FaCT protocols of this type were first proposed by Pfitzmann and Schunter in [104]. Subsequently many protocols were proposed [12, 19, 33, 39, 40, 54, 65, 66, 78, 102, 103, 104, 105, 138, 139]. The three phases are described below:

`Initial Setup`. In this phase C and D register with the KC to obtain authenticated public keys. The main protocol messages are shown in Figure 3.2, and the protocol steps are as follows:

(I) C and D register with the KC to obtain authenticated public keys.

1. The protocol message is:

$$\mathcal{I} \rightarrow \text{KC} : \{info\}_{AKE} .$$

The parties involved $\mathcal{I}$ (which can either be C or D) generate their respective asymmetric encryption and/or signature key pairs. The public keys are given to the KC together with their identity or personal information (e.g. ID_C and ID_D). Hence *info* may contain the public keys and ID_C or ID_D.

(II) The KC generates and provides C and D with the authenticated key materials.

2. This is shown as:

$$\text{KC} \rightarrow \mathcal{I} : \{info,\ \texttt{SIG}\}_{AKE} .$$

The KC (e.g. a CA) generates a signature `SIG` that contains information to prove the authenticity of the public keys of C (or D). As an example, `SIG` can be a signature $[hek_C, pvk_C, ID_C]_{SIG_{ssk_{KC}}}$, where hek_C and pvk_C are the public encryption and verification keys of C, and ID_C is the identity or personal information of C. In the description of certain existing protocols, we sometimes represent `SIG` as $Cert_{ssk_{CA}}(ID_C)$, following the notation used in the existing protocols where the central trusted third party is a CA. The *info* denotes other information that may need to be included and can be an empty string.

① `Initial Setup:`		
$\mathcal{I} \rightarrow$ KC	: $\{info\}_{AKE}$	**Before content distribution**
KC $\rightarrow \mathcal{I}$	: $\{info,\ \texttt{SIG}\}_{AKE}$	

Figure 3.2: Protocols without TTPs – Initial Setup

`Content Watermarking and Distribution`. In this phase C receives marked content from D. It only involves C and D. The protocol messages are shown in Figure 3.3 and the protocol steps are as follows:

② `Content Watermarking and Distribution:`		
$C \rightarrow D$	: $\{info,\ f(W),\ \texttt{SIG}\}_{AKE}$	**Content distribution**
$D \rightarrow C$	: $\left\{info,\ f(\widetilde{X})\right\}_{AKE}$	

Figure 3.3: Protocols without TTPs – Content Watermarking and Distribution

(I) C requests content and generates watermark information.

1. The protocol message is:

$$C \rightarrow D : \{info,\ f(W),\ \texttt{SIG}\}_{AKE}\,.$$

In this step C sends a request for content to D. C also generates watermark information $f(W)$. For example, $f(W)$ can be an encrypted client watermark. In many FaCT protocols C also approves a content agreement AGR with D. If this is the case then the signature `SIG` denotes a signature produced by C on $f(W)$ and AGR. The *info* may consists of AGR, the public keys of C, and the signature on these keys to prove their validity so that D can use these keys when needed.

(II) D produces a marked copy of the requested content and sends it to C.

2. The protocol message is:

$$D \rightarrow C : \left\{info,\ f(\widetilde{X})\right\}_{AKE}\,.$$

In this step D produces a marked content. The common method is to use watermarking in the encrypted domain, with the watermark information provided by C (see Section 2.3.3). The purpose is to embed the watermark, while at the same time preventing D from knowing what the watermark is. Since D has no way of determining the watermark, he may also embed another watermark so that he is able to identify the client that owns a specific content. We will observe these operations when we discuss the existing protocols in Chapter 4.

The resulting encrypted marked content $f(\widetilde{X})$ can then be sent to C. It is also common for many protocols to sign this encrypted marked content so as to ensure data authenticity. Hence *info* may include a signature.

`Identification and Dispute Resolution`. In this phase D identifies C from a found copy of content, and proves the client's act of illegal distribution to a third party. It involves at least D and A. Most of the protocols also require C or the KC to be involved. The protocol messages are shown in Figure 3.4.

③ `Identification and Dispute Resolution:`

D	: $\{\texttt{true}, \texttt{false}\} \leftarrow f^{WM}()$	
$D \rightarrow A$	: $\left\{info,\ \widehat{X},\ \texttt{SIG}\right\}_{AKE}$	**After content distribution**
$A \rightarrow C$ or KC	: $\{watermark\ info?\}_{AKE}$	
C or KC $\rightarrow A$	: $\{watermark\ info\}_{AKE}$	
A	: $\{\texttt{true}, \texttt{false}\} \leftarrow f^{WM}()$	

Figure 3.4: Protocols without TTPs – Identification and Dispute Resolution

The protocol steps are as follows:

(I) D detects a watermark from the found copy of content in order to identify the client that owns the content.

1. The detection is:
$$D : \{\texttt{true}, \texttt{false}\} \leftarrow f^{WM}(),$$
where $f^{WM}()$ means a watermark detection algorithm of a digital watermarking scheme, which we discussed in Section 2.3.1. The purpose is to identify the client based on the detected watermark.

(II) D proves to A that C illegally distributed copies of content.

2. D sends evidence that shows C is the owner of the found content to A. This information can be C's signatures produced on the content agreement and the encrypted watermark during the execution of the `Content Watermarking and Distribution` phase. Since for different protocol designs the signatures required are different, we represent such information in the protocol message as `SIG`. D also needs to send the found copy of content $\widehat{X}$ to A. There may be

other information such as AGR that needs to be passed on to A. Hence *info* may contain AGR.

$$D \rightarrow A : \left\{ info,\ \widehat{X},\ \texttt{SIG} \right\}_{AKE}.$$

Upon verifying the validity of information given by D, A requests from either C or KC the client watermark, or information on the watermark. This will allow A to detect the watermark from $\widehat{X}$.

$$A \rightarrow C \text{ or KC} : \{watermark\ info?\}_{AKE}.$$

C or the KC then sends the watermark information to A.

$$C \text{ or KC} \rightarrow A : \{watermark\ info\}_{AKE}.$$

A uses the provided watermark information and the watermark detection algorithm to detect the client watermark. If the detection is successful, then based on all the information provided by D, A declares C guilty. Otherwise, A declares C innocent.

$$A : \{\texttt{true}, \texttt{false}\} \leftarrow f^{WM}().$$

3.5.2 Category 2: Protocols with Online Trusted Third Parties

This category includes all protocols that require a special online trusted third party (TTP), which we refer to as a *watermark certification authority* WCA, in addition to the KC. The main characteristic of protocols in this category is that the WCA is tasked with generating client watermarks, thus avoiding the issue of C generating ill-formed watermarks faced in the previous category. However, the WCA always needs to be available for either D or C to request the client watermark. Hence requiring such an online trusted third party adds to the communication overhead. Example protocols based on this model include proposals in [3, 50, 85, 137].

`Initial Setup`. This phase is identical to that of **Category 1**.

`Content Watermarking and Distribution`. The difference between this phase and the similar phase in **Category 1** is that the client watermarks are generated by the WCA, instead of by C. The protocol messages are illustrated in Figure 3.5 and the protocol steps are as follows:

② Content Watermarking and Distribution:		
$C \rightarrow D$	: $\{info,\ \texttt{SIG}\}_{AKE}$	**content distribution**
$D \rightarrow$ WCA	: $\{watermark\text{-}request\ info\}_{AKE}$	
WCA $\rightarrow D$	: $\{info,\ f(W),\ \texttt{SIG}\}_{AKE}$	
$D \rightarrow C$	: $\left\{info,\ f(\widetilde{X})\right\}_{AKE}$	

Figure 3.5: Protocols with Online TTPs – Content Watermarking and Distribution

(I) C requests content from D.

1. The protocol message is:

$$C \rightarrow D : \{info,\ \texttt{SIG}\}_{AKE}\,.$$

Here *info* may contain AGR and the public keys of C, while SIG can be a signature on AGR.

(II) D requests watermark information from the WCA.

2. The protocol message is:

$$D \rightarrow \text{WCA} : \{watermark\text{-}request\ info\}_{AKE}\,.$$

D sends necessary information to convince the WCA that a client has requested content, so that the WCA will provide D with the watermark information. Such information is normally a client watermark. For example, the information submitted to the WCA may consist of all the information provided by C in Step (I):

$$watermark\text{-}request\ info = (info,\ \texttt{SIG})\,.$$

(III) The WCA sends the watermark information to D.

3. The protocol message is:

$$\text{WCA} \rightarrow D : \{info,\ f(W),\ \texttt{SIG}\}_{AKE}\,.$$

In this step the WCA generates a client watermark W, encrypts it with C's public key and signs the encrypted watermark W. The encrypted watermark $f(W)$ and the signature SIG are given to D. The signature may also contain AGR and a signature generated by C on this agreement.

(IV) D produces a marked copy of the requested content and sends it to C.

4. This step is identical to step (II) of the `Content Watermarking and Distribution` phase in **Category 1**.

`Identification and Dispute Resolution`. This involves D, the WCA and A. The WCA is required since it is the only party that has access to the client watermark, and A will have to contact this WCA to obtain this watermark for watermark detection. It is similar to the `Identification and Dispute Resolution` phase in **Category 1**, except that A contacts the WCA to request a client watermark, instead of contacting C or the KC. The protocol messages are shown in Figure 3.6.

③ `Identification and Dispute Resolution:`

D	: $\{\texttt{true}, \texttt{false}\} \leftarrow f^{WM}()$	
$D \rightarrow A$	: $\left\{info,\ \widehat{X},\ \texttt{SIG}\right\}_{AKE}$	**After content distribution**
$A \rightarrow$ WCA	: $\{watermark\ info?\}_{AKE}$	
WCA $\rightarrow A$	: $\{watermark\ info\}_{AKE}$	
A	: $\{\texttt{true}, \texttt{false}\} \leftarrow f^{WM}()$	

Figure 3.6: Protocols with Online TTPs – Identification and Dispute Resolution

3.5.3 Category 3: Protocols with Offline Trusted Third Parties

This model uses an offline TTP to generate watermark information (e.g. a client watermark), which is then passed to C. The TTP is offline since it is not involved in the actual content distribution between D and C. In other words, the TTP can be offline once C receives the watermark information. The Memon-Wong protocol [94] that will be presented in Section 3.7 falls into this category. Other protocols following this model are [24, 25, 74, 81, 110, 123].

`Initial Setup`. In this phase C and D obtain authenticated keys from the KC. In contrast to **Category 1** and **2**, C also contacts the TTP to obtain a client watermark (or a batch of distinct client watermarks). This means that the TTP need not be online when C requests content from D. The protocol messages are illustrated in Figure 3.7 and the protocol steps are as follows:

[*(I-II)*] The first two steps are identical to Step (I) and Step (II) in the `Initial Setup` phase of **Category 1** and **2**.

(III) C requests watermark information from the TTP.

3. The protocol message is:

$$C \rightarrow TTP : \{\textit{watermark-request info}\}_{AKE} .$$

In this step C sends the necessary information to convince the TTP that he is a legitimate user. For example:

$$\textit{watermark-request info} = (\textit{info},\ \texttt{SIG}) ,$$

where *info* contains C's identity information ID_C and his public keys, while `SIG` contains the KC's signature on these keys obtained from Step (I) and (II).

(IV) The TTP sends the watermark information to C.

4. The protocol message is:

$$TTP \rightarrow C : \{\textit{info},\ f(W),\ \texttt{SIG}\}_{AKE} .$$

Upon verifying C's request, the TTP provides C with the watermark information $f(W)$. This can be a client watermark, or key materials that contain a watermark, and it is signed to ensure data authenticity.

① `Initial Setup:`		
$\mathcal{I} \rightarrow$ KC	: $\{\textit{info}\}_{AKE}$	**Before content distribution**
KC $\rightarrow \mathcal{I}$	: $\{\textit{info},\ \texttt{SIG}\}_{AKE}$	
$C \rightarrow TTP$	: $\{\textit{watermark-request info}\}_{AKE}$	
$TTP \rightarrow C$	: $\{\textit{info},\ f(W),\ \texttt{SIG}\}_{AKE}$	

Figure 3.7: Protocols with Offline TTPs – Initial Setup

The `Content Watermarking and Distribution` phase is similar to the phase in **Category 1**. The only difference is that the watermark information is provided by the TTP to C, instead of being generated by C. The `Identification and Dispute Resolution` phase, on the other hand, is identical to that of **Category 2**.

3.5.4 Category 4: Protocols with Trusted Hardware

Protocols in this category deploy trusted hardware in D's and/or C's computing platforms. The main idea is to use this trusted hardware to generate and/or verifying client watermark information. It can also be used to securely embed watermarks into content. These devices can play the roles of either the online or offline TTP in the previous categories. For example, instead of contacting the WCA in **Category 2**, D may deploy trusted hardware to replace the WCA. We discuss one such protocol in Section 7.2. The major advantage is that, especially if the device resides on C's computing platform, there is no single central TTP, but many different devices in each client's computing platform. This allows for scalability and so is more suitable for distributed computing environments. Such setup also allows the device to play the role of an offline TTP, albeit a distributed one. However, the extra cost of hardware implementation must be factored into the design of such protocols. In [129] a general framework was suggested for how trusted hardware can be used to design secure content distribution infrastructures. Other protocols proposed in this category are [46] and [86]. The general construction of protocols in this category is presented below.

`Initial Setup`. This phase is identical to that of **Category 1**.

`Content Watermarking and Distribution.` In addition to C and D, it also involves trusted hardware TH. The protocol messages are shown in Figure 3.8 and the protocol steps are as follows:

② Content Watermarking and Distribution:		
$C \rightleftharpoons TH$	: $\{info,\ W\}_{AKE}$	**content distribution**
$C \rightarrow D$	: $\{info,\ f(W),\ \texttt{SIG}\}_{AKE}$	
	OR	
$C \rightarrow D$	: $\{info,\ \texttt{SIG}\}_{AKE}$	
$D \rightleftharpoons TH$	: $\{info,\ f(W)\}_{AKE}$	
$D \rightarrow C$	: $\left\{info,\ f(\tilde{X})\right\}_{AKE}$	

Figure 3.8: Protocols with TH – Content Watermarking and Distribution

There are two cases. If the trusted hardware is in C's computing platform then:

(Ia) C (or TH) generates the watermark and the trusted hardware verifies the wa-

termark to be well-formed.

1. In this step, before C requests content from D, C (or the trusted hardware) generates a watermark W. The trusted hardware checks W to ensure that the watermark is well-formed, and passes W and the watermark well-formed measurement to C. The interactions between C and TH are shown as:

$$C \rightleftharpoons TH : \{info,\ W\}_{AKE},$$

where *info* may contain the measurement on the watermark and $\rightleftharpoons$ means the communication between C's computing platform and TH.

(IIa) C requests content and provides D with the watermark information.

2. This step is identical to Step (I) in the `Content Watermarking and Distribution` phase in **Category 1**, except that, as shown in Step (Ia) above, the watermark is processed by TH, instead of C.

If the trusted hardware is in D's computing platform then:

(Ib) C requests content from D.

1. This step is identical to Step (I) in the `Content Watermarking and Distribution` phase in **Category 2**.

(IIb) D and the trusted hardware generate the watermark. The trusted hardware verifies the watermark to be well-formed.

2. In this step, after verifying the request from C, D generates the watermark together with TH. It is generated and given to D in such a way that D cannot determine what the watermark is. The protocol message is

$$D \rightleftharpoons TH : \{info,\ f(W)\}_{AKE},$$

where *info* denotes other information and can be an empty string.

The final step for both cases is:

(III) D produces a marked copy of the requested content and sends it to C.

3. This is identical to Step (II) of the `Content Watermarking and Distribution` phase in **Category 1**.

`Identification and Dispute Resolution`. This phase involves D and A. If the watermark is generated by C together with TH then C is also involved. D identifies C through the watermark detected from a found copy and proceeds to prove C's act of illegal content distribution to A. The steps and protocol messages are identical to the `Identification and Dispute Resolution` phase of the previous categories, except that instead of A interacting with either a WCA, TTP or C, A requires the trusted hardware in C or D's computing platform to reveal the client watermark. The protocol messages are shown in Figure 3.9.

③ Identification and Dispute Resolution:		
D	: $\{\texttt{true}, \texttt{false}\} \leftarrow f^{WM}()$	
$D \rightarrow A$	: $\left\{info,\ \widehat{X},\ \texttt{SIG}\right\}_{AKE}$	**After content distribution**
$A \rightleftharpoons TH$	: $\{watermark\ info\}_{AKE}$	
A	: $\{\texttt{true}, \texttt{false}\} \leftarrow f^{WM}()$	

Figure 3.9: Protocols with TH – Identification and Dispute Resolution

3.5.5 Adding Anonymity and Unlinkability

In addition to providing the three main security requirements discussed in Section 3.3.3, many protocols include the protection of client privacy. This was first introduced by Pfitzmann and Waidner in [105]. Protocols with this additional property may fall into any of the four main categories discussed above. Such protocols can be found in [19, 24, 25, 26, 33, 39, 40, 53, 54, 74, 78, 85, 102, 103, 118, 123]. The main mechanism for providing anonymity and unlinkability is, instead of communicating with D based on the long term public keys, C communicates with D using a set of temporary keys called *pseudonyms*. Similarly to the long term keys, these *pseudonyms* are signed by the KC to prove their validity. These *pseudonyms* do not contain identity information for C. Therefore, when C uses these *pseudonyms* to

request content from D, D will not know the real identity of C, but can verify that such *pseudonyms* are valid based on the signatures of the KC. Protocols with this property are discussed in Chapters 5, 7 and 8. In the following we present the three phases (Figure 3.10).

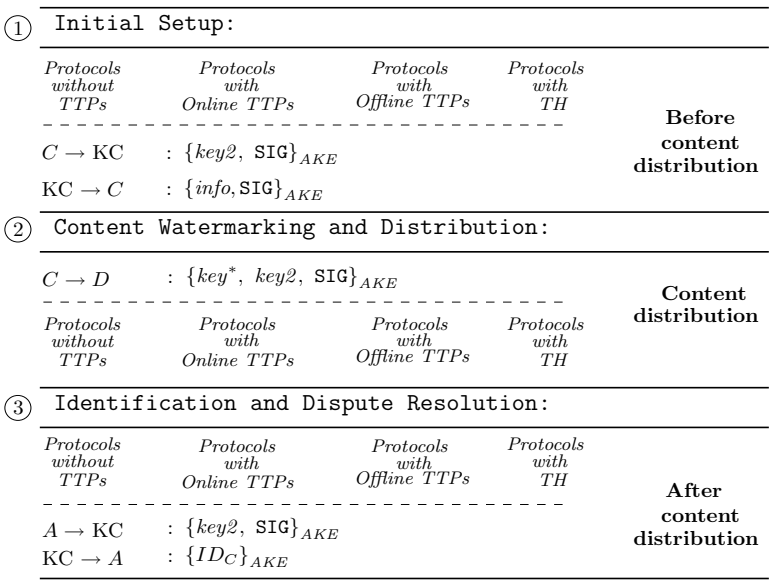

Figure 3.10: Protocols with Anonymity and Unlinkability

`Initial Setup`. This is the main phase that provides the required mechanism for anonymity and unlinkability. It involves C, D and the KC. Its major difference from other `Initial Setup` phases is that instead of just obtaining the KC's signatures on the long term public keys that contain C's identity information, C further requests another distinct signature on newly generated public keys that do not contain such information. These new keys, together with the new signature, are known as the *pseudonyms*. In the following we present the protocol steps:

(I-II) The first two steps are identical to Step (I) and Step (II) in the `Initial Setup` phase in **Category 1**. In these two steps, C and D register with the KC. The KC signs the public keys (*key*) of C and D.

(III) Anonymous certification of new randomly generated key pair by the KC.

3. The protocol message is

$$C \rightarrow KC : \{key2,\ \texttt{SIG}\}_{AKE}\,.$$

C then generates a pseudonym *key2*, signs this key with his long term signing key and sends the signature `SIG` together with *key2* to the KC.

(IV)The KC sends signed key2 to C.

4. The protocol message is:

$$KC \rightarrow C : \{info, \texttt{SIG}\}_{AKE} .$$

The KC verifies the client signature on *key2*, re-signs *key2* and sends the signature `SIG` back to C.

`Content Watermarking and Distribution`. This is similar to the `Content Watermarking and Distribution` phase in whichever category applies. The only difference is that C generates arbitrary keys (key^*). These keys are signed by C using the private half of *key2*. So C uses key^*, instead of his authenticated long term keys (*key*), to request content. This is shown as:

$$C \rightarrow D : \{key^*,\ key2,\ \texttt{SIG}\}_{AKE} ,$$

where `SIG` contains a signature produced using *key2* and a signature on *key2*, which is generated by the KC.

`Identification and Dispute Resolution`. This phase is similar to the `Identification and Dispute Resolution` phase in whichever category applies. The only difference is the two extra protocol messages at the end of the process, as shown in Figure 3.10:

$$A \rightarrow \text{KC} : \{key2,\ \texttt{SIG}\}_{AKE} ,$$

for A to request C's identity information and

$$\text{KC} \rightarrow A : \{ID_C\}_{AKE} ,$$

where the KC retrieves and sends C's identity information ID_C to A. These are required for A to identify C, since only the KC has the identity information.

3.5.6 Adding Payment and Fair Exchange

Payment. One other aspect that has not been studied before in existing protocols is how to include a payment mechanism. Many protocols, such as [85] and [104], include a purchase agreement as an integral part of the protocols, which implicitly means that payment is involved, but do not provide details on how payment is conducted.

Based on [127], we show in Figure 3.11 how this is possible by including a payment infrastructure on top of a FaCT protocol. A payment mechanism is normally agreed between C and D with their respective banks, entering into contractual relationships. This can be performed before the `Initial Setup` phase of a FaCT protocol in whichever category applies. With the payment infrastructure in place, a *payment* token can be included as one of the messages in the communication between C and D. Depending on the mechanism chosen, the *payment* token can mean, for example, *digital coins* as in an electronic cash system such as NetCard [4], or credit or debit card details of C [119]. It may also contain specific account details of the parties involved. For example, from the e-payment protocol proposed by Zhang and Markantonakis in [140], the payment token contains the amount a client should pay, the distributor's and the client's bank details, both encrypted using the respective bank's public keys.

Although the inclusion of a payment mechanism may be seen as simple and straightforward, it brings out the issue of *fair exchange* between D and C.

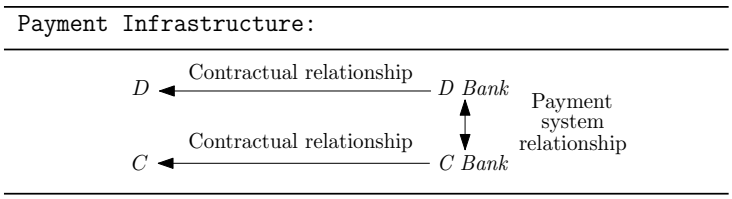

Figure 3.11: Payment Infrastructure

Fair Exchange. This is important since in the context of fair content tracing, D and C *do not trust each other*. Thus when payment is included, in addition to ensuring the other requirements, we must also ensure that D and C will exchange the payment and content in a fair manner, which motivates the optional requirement of *fair exchange* introduced in Section 3.3.3.

In order to fulfill this requirement, a FaCT protocol can be constructed based on existing well-established schemes such as the fair exchange scheme in [5] or concurrent signature schemes in [22]. We also note that a FaCT protocol with fair exchange may fall into any of the main categories of Section 3.5.

Assuming that C and D have agreed on a payment mechanism, we discuss a general construction, based on the framework in [5], in the following. As shown in Figure 3.12, the `Initial Setup` phase and the `Identification and Dispute Resolution` phase follow any of the FaCT protocol in whichever category applies.

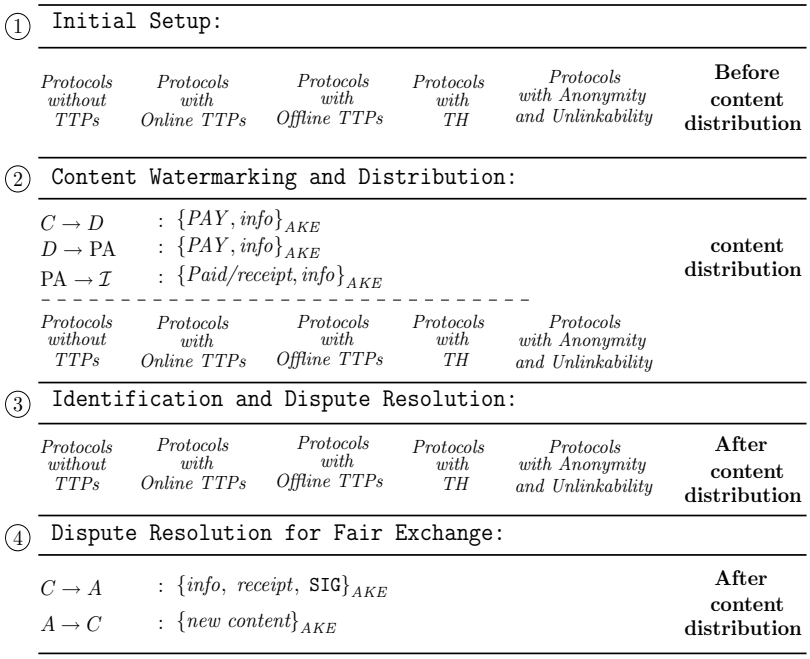

Figure 3.12: Protocols with Fair Exchange

`Initial Setup`. The protocol steps in this phase follow any of the protocol steps in the `Initial Setup` phase of other categories. We remark that the role of the KC and/or the special TTP might be played by the payment agent PA (i.e. C's or D's banks or a payment gateway). This is because, since C and D need to register with the PA to setup the payment infrastructure, the PA may also issue authenticated keys to the client and the distributor or generate client watermarks.

`Content Watermarking and Distribution`. The protocol steps in this phase also follow any of the protocol steps in the `Content Watermarking and Distribution` phase of other categories. However, three extra messages must be included. This is the payment information PAY of the client, which is sent with the request for

content when C starts the interaction with D. This PAY will allow D to receive the correct amount of payment for the content given to C. In Figure 3.12, we show the payment message as:

$$C \rightarrow D : \{PAY,\ info\}_{AKE},$$

where *info* may contain a signature on the payment information, but this may not be necessary, for example, in the case of purchasing based on credit card details. We note that this message can be included in the request for content message in any of the protocols in other categories.

Upon receiving this payment information, D forwards it to the PA so that the PA can process the payment and give D the correct amount of payment. This is shown as:

$$D \rightarrow \text{PA} : \{PAY,\ info\}_{AKE}.$$

Another message is the *payment receipt* issued by the PA, whenever the correct amount has been paid into D's account. The *payment receipt* plays a crucial role in resolving payment disputes between C and D. The *payment receipt* is shown as:

$$\text{PA} \rightarrow \mathcal{I} : \{receipt,\ info\}_{AKE},$$

where *info* may contain the signature on *receipt* produced by the PA and $\mathcal{I}$ denotes either C or D.

`Identification and Dispute Resolution`. This phase is identical to the `Identification and Dispute Resolution` phase for the protocols in other categories, as identifying and proving illegal content distribution are independent from the fair exchange issue.

`Dispute Resolution for Fair Exchange`. This is the additional phase that addresses the issue when C does not receive content (or correct content), guaranteeing *fair exchange* between C and D. The two main steps are (Figure 3.12):

$$C \rightarrow A : \{info,\ receipt,\ \texttt{SIG}\}_{AKE},$$

where C sends the received payment receipt together with other information to A as evidence that C has paid for the said content, and:

$$A \rightarrow C : \{new\ content\}_{AKE},$$

where A requests new content from D and forward it to C. We note that there is no need for a payment dispute resolution process for D as D receives the payment token before sending content to C, as shown in Figure 3.12. Hence if the payment information is not correct, D halts the protocol without losing anything.

In Table 3.2, we summarise the main characteristics in each of the four categories, while Table 3.3 summarises the additional features when privacy protection or fair exchange (or both) are added.

Table 3.2: Main Characteristics of Existing FaCT Protocols

Category	**Existing Designs**
Protocols without TTPs	C generates watermark information. *Benefit*: No special TTP. *Issue*: Extra measure needed to prevent C generating ill-formed watermarks.
Protocols with online TTPs	TTP generates watermark information. D or C contacts TTP during content distribution. *Benefit*: Avoid the issue of ill-formed watermarks since TTP generates them. *Issue*: Central TTP must always be available.
Protocols with offline TTPs	TTP generates watermark information. D or C contacts TTP during initial setup. *Benefit*: Avoid the need of a central TTP that must always be available during content distribution. *Issue*: Central TTP required during initial setup.
Protocols with TH	Trusted hardware generates watermark information. *Benefit*: No special TTP. Suitable for distributed systems. *Issue*: Extra hardware cost.

Table 3.3: Adding Privacy Protection, Payment and Fair Exchange

Requirements	**Existing Designs**
Anonymity & Unlinkability	C generates *pseudonyms*.
Fair Exchange	C includes *payment* information. An additional dispute resolution phase for fair exchange.

Table 3.4 identifies protocols that will be discussed in the subsequent chapters based on the four categories described above. In the table, **Std.** denotes protocols fulfilling the three standard requirements of a FaCT protocol, while **AU** denotes anonymity and unlinkability, **FE** denotes fair exchange and **AU+FE** denotes anonymity and unlinkability and fair exchange. The protocols in *italics* are either existing protocols that we cryptanalyse or new protocols that we propose.

Table 3.4: FaCT Protocols Discussed in Subsequent Chapters

	Std.	AU	FE	AU+FE	Sec.
Protocols without TTPs	Pfitzman-Schunter				4.2
	Ibrahim-ElDin-Hegazy				4.3
	Semi-Fair				4.4
Protocols with online TTPs		Lei-Yu-Tsai-Chan			5.2
	Wu-Pang				5.3
	Ahmed-Sattar-Siyal-Yu				5.4
				Fair Exchange	8.2
Protocols with offline TTPs	Memon-Wong				3.7
	Kuribayashi-Tanaka				6.2
	Chameleon Encryption-based				6.3
Protocols with TH		Fan-Chen-Sun			7.2
		TPM-DAA-based			7.3.2
		TPM-PrivacyCA-based			7.3.3

3.6 Evaluation Criteria

In this section we present evaluation criteria for the analysis of FaCT protocols based on our design framework. These target aspects of efficiency and can be used to compare protocols that are proposed within one category, or indeed to compare protocols in different categories. The criteria are assessed based on the processes related to the generation of the final marked content by D and C, since this represents the most important (and generally most expensive) operation. The evaluation criteria are presented as follows:

1. **Bandwidth.** The size of the encrypted marked content transmitted from D to C affects the bandwidth required. For evaluation purposes, we assume that the number of elements in content (or a watermark) is n and the size of each element in content (or a watermark) is Z. We further denote the bit-length of Z by $|Z|$. Based on [1], for example, a common value for $|Z|$ is 16 to 32 bits. We also assume that the modulus of the asymmetric building blocks (i.e. homomorphic encryption and digital signature schemes) is m, and its bits-length is $|m|$. As stated in [51], a current practical value for $|m|$ is 1024 bits.

2. **Trusted Third Parties.** The existence of a special TTP adds extra protocol messages to the execution of a FaCT protocol compared to protocols that do not require such a TTP.

3. **Computation.** This means the degree of computation that each party needs to perform to produce the encrypted marked content. It relies on the performance of the underlying building blocks, which consist of modular exponentiation (**E**), modular multiplication (**M**), and modular addition (**A**). We assume that **E** requires $O(k^3)$ bit operations, **M** requires $O(k^2)$ bit operations and **A** requires $O(k)$ bit operations. This is based on the basic bit operations discussed in [79].

   We further note that, based on the description provided in [84], a DES encryption is at least 100 times faster than an RSA encryption. This increases to 1000 or 10000 times in hardware, depending on different implementations. Based on the benchmark of Crypto++ [31], which has been deployed in many applications, AES with a 128 key requires an average 21 processor cycles per byte, while RSA-1024 encryption requires 130000 processor cycles per operations. Assuming that each AES encryption processes 16 bytes, then an AES encryption is nearly 400 times faster than an RSA-1024 encryption. We thus assume, conservatively, that a symmetric encryption scheme is 100 times faster than an asymmetric encryption scheme. By further assuming that computing an asymmetric encryption as equivalent to a modular exponentiation, then computing a symmetric encryption (**S**) is equivalent to **E**/100.

4. **Storage.** This means the amount of storage required by C and D to store their respective key materials and the watermark information used for generating and retrieving the encrypted marked content.

3.6.1 Brief Analysis of the Four Categories

Bandwidth. For all the categories, the bandwidth depends on the underlying building blocks that are used to produce the encrypted marked content. For example, although the size of each element of the original content may only be $|Z| = 32$, the encrypted marked element of content may have size $|m| = 1024$ bits. This is due to the use of homomorphic encryption to encrypt each element of content, as in the Memon-Wong protocol (see Section 3.7) and other protocols discussed in subsequent chapters.

Trusted Third Parties. If we compare the four categories discussed previously, a protocol without trusted third parties is more efficient than the other three cate-

gories. This is because the protocols with TTPs or trusted hardware require at least two extra protocol messages when D (or C) communicate with these TTPs or the trusted hardware. However, the protocols with trusted hardware are more suitable for distributed environments since there is no central TTP, as in the protocols with online and offline TTPs. Nevertheless, extra hardware cost must be factored in. In addition to the above, extra communication with the TTP is also required when privacy protection and fair exchange are added as extra requirements. For privacy protection, C needs to communicate with the TTP to obtain pseudonyms, while for fair exchange, the parties involved need to communicate with a payment agent.

Computation. Similar to determining the required bandwidth, the computation requirement of a FaCT protocol relies on the performance of the underlying building blocks. For example, some protocols without trusted third parties, such as the Pfitzmann-Schunter protocol (see Section 4.2), use zero-knowledge proof techniques and asymmetric homomorphic bit commitment schemes. Compared to protocols in other categories that use asymmetric homomorphic encryption schemes, using zero-knowledge proof techniques requires more computation and hence these protocols are less efficient. More recent proposals, such as the Wu-Pang protocol, with an online trusted third party, and our protocol based on Chameleon Encryption with an offline trusted third party, use symmetric building blocks. Thus these protocols are more efficient than the conventional ones that use asymmetric homomorphic schemes. These two protocols are described in Sections 5.3 and 6.3 respectively.

Storage. The storage size required for the key materials and watermark information also depends on the underlying building blocks. For example, if the key materials are the public and private keys of an asymmetric homomorphic encryption scheme, then the required storage size would be $2|m|$. The size of storage for the watermark information depends on whether a watermark or an encrypted watermark is stored.

Also, depending on the underlying category, either C or D, or the TTP store the watermark information. For example, in the protocol without trusted third parties, C generates the client watermark. Hence C needs to store the watermark since, when there is a dispute, he will be requested to reveal the watermark. Using the previous notation, the size of the watermark is $n|Z|$. This is also the case when trusted hardware resides on C's computing platform for a protocol with trusted hardware.

In contrast, in the categories that involve online and offline trusted third parties, either D or the TTP stores the watermark information. The watermark information stored by D is typically the encrypted watermark, since D should not know the watermark as described in Section 2.2. Hence if the watermark is encrypted using an asymmetric homomorphic scheme as in the Memon-Wong Protocol (see Section 3.7), then the storage size is $n|m|$, which is much larger than $n|Z|$. This is also the case when trusted hardware resides in D's computing platform for a protocol with trusted hardware.

Table 3.5 provides a summary of this brief analysis, where **AU+FE** denotes the addition of privacy protection and fair exchange.

Table 3.5: Brief Evaluation of the Existing FaCT Protocols

Category	Criteria
Protocols without TTPs	Bandwidth, computation & storage depends on the building blocks. No TTP: no extra communication. C stores watermark information.
Protocols with online TTPs	Bandwidth, computation & storage depends on the building blocks. TTP: At least two extra communications during content distribution. D or TTP stores watermark information.
Protocols with offline TTPs	Similar to protocols with online TTPs except that the two extra communications happen during initial setup.
Protocols with TH	Bandwidth, computation & storage depends on the building blocks. TH: At least two extra communications with TH. Hardware implementation cost. D or C stores watermark information.
AU+FE	Extra communication with TTP to obtain pseudonym or to ensure fair exchange.

3.7 An Example: The Memon-Wong Protocol

We now illustrate the use of the framework to describe the Memon and Wong (MW) FaCT protocol from [94]. This is one of the earliest proposals. The protocol is a *protocol with offline trusted third parties*.

Fundamentals. It involves five parties. These are D, C, the WCA, the CA and A. The WCA, CA and A are fully trusted and the protocol fulfills the three specific requirements of a FaCT protocol.

Environment. The MW protocol implicitly assumes that both D and C have ample computing resources. This is based on the observation that both D and C need many computations of homomorphic encryption in order to embed the watermark into content and retrieve the marked content. It also implicitly assumes the existence of public key and secure communication support. The WCA is a trusted third party that generates client watermarks. The underlying building blocks required are digital watermarking schemes, the original RSA encryption scheme (Section 2.3.2), the spread spectrum watermarking scheme (Section 2.3.1) and digital signature schemes. We note that most of the more recent protocols advocate the use of the Paillier encryption scheme (Section 2.3.2) instead of the original RSA scheme due to the Paillier scheme being semantically secure. This is due to the deterministic nature of the RSA scheme as described in Section 2.3.2. Table 3.6 summarises the design framework. As can be observed, we have not included in the table the *threats* as discussed in Section 3.3.2. We reason that the inclusion of *security properties* are suffice to reflect both the underlying threats and the properties of the protocol. This also applies to our protocol discussions in the subsequent chapters. In the following we describe the three phases of the protocol.

Table 3.6: The Design Framework of the MW Protocol

Fundamentals	
Parties Involved	C, D, CA, WCA, A
Trust Assumptions	CA, WCA, A are *fully trusted*
Security Properties	Traceability (**TR**), Framing resistance (**FR**), Non-repudiation of redistribution (**NR**)
Environment	
Comp. Resources	Implicitly assumed D and C have ample resources
Sec. comm. Support	Required
Pub. Key Support	Required
TTPs	offline TTP (WCA)
Building Blocks	Digital watermarking scheme, homomorphic encryption scheme and digital signature scheme

`Initial Setup.` In this phase both D and C register with the CA and obtain signatures on their public keys. C further requests a watermark from the WCA. Figure 3.13 illustrates the protocol messages. We describe the protocol steps as follows:

(I) C and D register with the CA to obtain authenticated public keys.

① Initial Setup:		
$C\&D \rightarrow$ CA	: $\{Request\ authenticated\ keys\}_{AKE}$	**Before content distribution**
CA $\rightarrow C$	: $\left\{[hek_C, pvk_C, ID_C]_{SIG_{ssk_{CA}}}\right\}_{AKE}$	
CA $\rightarrow D$	: $\left\{[hek_D, pvk_D, ID_D]_{SIG_{ssk_{CA}}}\right\}_{AKE}$	
$C \rightarrow WCA$	: $\{\text{Request watermark}\}_{AKE}$	
$WCA \rightarrow C$	: $\left\{[W]_{HE_{hek_C}}, [[W]_{HE_{hek_C}}]_{SIG_{ssk_{WCA}}}\right\}_{AKE}$	

Figure 3.13: MW Protocol – Initial Setup

1. C and D register with the CA by providing CA with their personal information and their public keys.

(II) The CA generates and provides C and D with the authenticated public keys.

2. Upon verifying the identity information of C and D, the CA produces signatures on the respective public keys provided by both parties and sends the signatures to these parties. The signatures allow other parties to verify the authenticity of the public keys.

(III) C requests a client watermark from the WCA.

3. With the possession of the authenticated public keys, C requests a watermark W from the WCA by convincing the WCA that he is a legitimate user.

(IV) The WCA sends an encrypted client watermark to C.

4. the WCA, after verifying C as a valid client (such as through C's public keys and the signature generated by the CA), generates a watermark $W = (w_1, \ldots, w_n)$. Next the WCA encrypts every element of W with a homomorphic encryption scheme using C's public encryption key hek_C:

$$[w_i]_{HE_{hek_C}} \qquad 1 \leq i \leq n.$$

We denote $[W]_{HE_{hek_C}} = ([w_1]_{HE_{hek_C}}, [w_2]_{HE_{hek_C}}, \ldots, [w_n]_{HE_{hek_C}})$.

5. Next, the WCA digitally signs the encrypted watermark $[W]_{HE_{hek_C}}$ with his signing key, resulting in a signature:

$$[[W]_{HE_{hek_C}}]_{SIG_{ssk_{WCA}}}.$$

Finally, the WCA sends $[W]_{HE_{hek_C}}$ and $[[W]_{HE_{hek_C}}]_{SIG_{ssk_{WCA}}}$ to C. Clearly, C can determine W by simply decrypting $[W]_{HE_{hek_C}}$. However, C can not replace W with any other watermark since C does not have the private signing key to generate the signature on this other watermark.

`Content Watermarking and Distribution.` This phase involves C and D for the distribution of content. In constrast to the general construction described in Section 3.5.2, there is no content agreement involved.

②	`Content Watermarking and Distribution:`		
$C \rightarrow D$	: $\left\{[W]_{HE_{hek_C}}, [[W]_{HE_{hek_C}}]_{SIG_{ssk_{WCA}}}\right\}_{AKE}$	**Content distribution**
$D \rightarrow C$	: $\left\{[X'']_{HE_{hek_C}}\right\}_{AKE}$	

Figure 3.14: MW Protocol – Content Watermarking and Distribution

The protocol steps are described in the following (Figure 3.14):

(I) C requests content and provides D with the encrypted watermark obtained from the TTP.

1. C sends $[W]_{HE_{hek_C}}$ and $[[W]_{HE_{hek_C}}]_{SIG_{ssk_{WCA}}}$ to D.

(II) D produces a marked copy of the requested content and sends it to C.

2. D verifies $[[W]_{HE_{hek_C}}]_{SIG_{ssk_{WCA}}}$ using the WCA verification key pvk_{WCA}. If the signature is valid, D generates a watermark V using a digital watermarking scheme of his choice and embeds this watermark into the content X that C wishes to purchase.

$$X' \leftarrow [X, V]_{EMB_{wmk_V}}.$$

This watermark V allows D to trace the content to C's identity.

3. D encrypts every element of X' one-by-one with the same homomorphic encryption scheme used by the WCA, using C's public encryption key hek_C. After that, D permutes (or changes the order of) every encrypted element of W). For example, given

$$[W]_{HE_{hek_C}} = ([w_1]_{HE_{hek_C}}, [w_2]_{HE_{hek_C}}, \ldots, [w_n]_{HE_{hek_C}}),$$

D permutes the encrypted elements using a permutation q as:

$$q\left([W]_{HE_{hek_C}}\right) \quad = \quad ([w_{q(1)}]_{HE_{hek_C}}, [w_{q(2)}]_{HE_{hek_C}}, \ldots, [w_{q(n)}]_{HE_{hek_C}}).$$

Next D multiplies every encrypted element of X' with every permuted and encrypted element of W. It is assumed that the underlying building blocks are the RSA encryption scheme described in Figure 2.3 and the SS watermarking scheme described in Figure 2.1:

$$\left.\begin{array}{ll} & [x'_i]_{HE_{hek_C}} \cdot [1 + \rho q(w_i)]_{HE_{hek_C}} \\ = & [x'_i \cdot (1 + \rho q(w_i))]_{HE_{hek_C}} \\ = & [x''_i]_{HE_{hek_C}} \end{array}\right\} \; 1 \leq i \leq n.$$

We denote $[X'']_{HE_{hek_C}} = ([x''_1]_{HE_{hek_C}}, [x''_2]_{HE_{hek_C}}, \ldots, [x''_n]_{HE_{hek_C}})$. The multiplication of these elements effectively embeds the watermark $q(W)$ into X', resulting in an encrypted marked content $[X'']_{HE_{hek_C}}$. Clearly, it is $q(W)$ that is embedded instead of W. The reason for this is that C knows W, and it will be trivial for him to remove the watermark if W is embedded. He can do this by subtracting W from the marked content X''. Hence the permutation is crucial for the security of the protocol.

4. After that, D sends $[X'']_{HE_{hek_C}}$ to C.

5. Finally, C decrypts $[X'']_{HE_{hek_C}}$ to recover the marked content X' using his private decryption key hdk_C.

`Identification and Dispute Resolution.` In this phase D proves to A that a dishonest C illegally distributed copies of content. The protocol messages are shown in Figure 3.15, and the following describes the protocol steps:

(I) D detects a watermark from the found copy of content in order to identify the client that owns the content.

1. When an illegal copy of content $\widehat{X}$ is found, D runs the watermark detection algorithm to try to detect watermark V:

$$\{\texttt{true}, \texttt{false}\} \leftarrow [\widehat{X}, V, X]_{DET_{wmk}}.$$

If the detection algorithm returns `true`, then detection of V is successful and D proceeds to identify C.

(II) D proves to A that C illegally distributed copies of content.

2. Next, D proves to A the illegal act of C by providing evidence in the form of the identity of C (based on the detection of V), found content $\widehat{X}$, marked content X' and the signed and encrypted watermark $[[W]_{HE_{hek_C}}]_{SIG_{ssk_{WCA}}}$, together with the permutation q. C will be proved guilty if the permuted watermark $q(W)$ can be detected from $\widehat{X}$:

$$\texttt{true} \leftarrow [\widehat{X}, q(W), X']_{DET_{wmk}}.$$

Clearly, the WCA will need to store a copy of W, so that when disputes such as the above occur, W can be retrieved to match $q(W)$ and hence serve as evidence to prove C has indeed illegally distributed this copy of content.

③ Identification and Dispute Resolution:

D	: $\{\texttt{true}, \texttt{false}\} \leftarrow [\widehat{X}, V, X]_{DET_{wmk}}$	
$D \rightarrow A$	: $\left\{ID_C, \widehat{X}, [[W]_{HE_{hek_C}}]_{SIG_{ssk_{WCA}}}, X', q\right\}_{AKE}$	**After content distribution**
$A \rightarrow WCA$	: $\{\text{Request watermark } W\}_{AKE}$	
$WCA \rightarrow A$	: $\{W\}_{AKE}$	
A	: $\texttt{true} \leftarrow [\widehat{X}, q(W), X']_{DET_{wmk}}$	

Figure 3.15: MW Protocol – Identification and Dispute Resolution

3.7.1 Security

The MW protocol provides the three security requirements as follows:

Traceability. For every client, two watermarks V and W are generated to allow tracing. The watermark V retrieved from a copy of content allows D to identify C, while watermark W, which is not known to D, allows tracing of C by A when A is provided with the necessary information.

Framing resistance. D cannot determine watermark W since it is encrypted with C's public encryption key. This means that it is not possible for D to frame C by embedding the watermark W in any other content and distribute the marked copy of this other content. However, as stated by Lei *et al.* [85], when an illegal copy is found, it is possible for D to extract the watermark W, embed this watermark

into another higher value content and thus frame C by illegally distributing this higher value content. This is because there is no binding agreement that binds the watermark with the exact content where this watermark is embedded. This issue is known as the *unbinding problem*.

Non-repudiation of redistribution. If a copy of content $\widehat{X}$ is found, C cannot claim that this copy is distributed by D since D has no knowledge of W and the final marked copy. This is because D cannot decrypt the final encrypted marked content $[X'']_{HE_{hek_C}}$ given to C. More importantly, C cannot deny W being his watermark since the encrypted W is signed by the trusted WCA, and only C is able to decrypt $[X'']_{HE_{hek_C}}$.

3.7.2 Efficiency

Table 3.7 summarises the performance of the MW protocol.

Bandwidth. From Table 3.7, we see that the size of the encrypted marked content $[X'']_{HE_{hek_C}}$ transmitted from C to D is $n|m|$. This is due to the encryption and watermarking of n elements of content based on the homomorphic encryption scheme with modulus m, as described in the `Content Watermarking and Distribution` phase.

Trusted Third Parties. The protocol also requires an *offline TTP* in the form of a WCA. This means that during the `Initial Setup` phase, two extra protocol messages are required for C to obtain the watermark from WCA, in addition to the standard messages for acquiring authenticated keys.

Computation. C requires $n\mathbf{E}$ operations to decrypt the encrypted marked content given by D. However, D needs to perform more computation. As can be observed in the `Content Watermarking and Distribution` phase, D encrypts all n elements of content ($n\mathbf{E}$) and multiplies these encrypted elements with the encrypted watermark provided by C ($n\mathbf{M}$). This is to produce the final encrypted marked content. Before this, D also embeds a watermark V into content for the purpose of later identifying the owner of content ($n\mathbf{A}$). Hence D needs to perform $n(\mathbf{E} + \mathbf{M} + \mathbf{A})$ computations.

Storage. In the context of producing the encrypted marked content, C stores the decryption and encryption keys of the homomorphic encryption scheme used

$(2|m|)$. C also stores the watermark W, which has size $n|Z|$, instead of the encrypted watermark, since C can decrypt it. D requires the encryption key of C $(|m|)$, the encrypted watermark provided by the WCA $(n|m|)$ and the watermark V $(n|Z|)$.

Table 3.7: Performance of the MW Protocol

Bandwidth	*TTP*	*Computation*[1]	*Storage*[2]
$[X'']_{HE_{hek_C}} = n\|m\|$	offline WCA	C: $n\mathbf{E}$ D: $n(\mathbf{E} + \mathbf{M} + \mathbf{A})$	C: $2\|m\| + n\|Z\|$ D: $(n+1)\|m\| + n\|Z\|$

[1] $\mathbf{E}$=$O(k^3)$, $\mathbf{M}$=$O(k^2)$, $\mathbf{A}$=$O(k)$
[2] $|Z| < |m|$

3.8 Summary

In this chapter we presented a framework that can be used to analyse FaCT protocols in a systematic manner. We first put forward our thoughts on why a design framework is necessary. We then described the framework in detail, and proposed a classification of existing FaCT protocols. Finally we used the Memon-Wong protocol as an example to illustrate use of the framework.

Chapter 4

FaCT Protocols without Trusted Third Parties

Contents

This chapter examines FaCT protocols that do not require trusted third parties. We describe two different approaches and discuss existing protocols based on these two approaches. The first approach is more computationally intensive than the second approach, but the second approach faces security issues. We examine these issues by demonstrating design flaws in two recently proposed protocols. We further propose a semi-fair protocol that is relatively efficient and does not face the issues but requires a stronger trust assumption. We conclude that it is difficult to design an efficient FaCT protocol without a trusted third party.

4.1 Overview

As described in Section 3.5.1, FaCT protocols without trusted third parties are protocols that do not involve a WCA. The main characteristic of these protocols is that the clients are responsible for generating their own watermarks. There are two approaches:

- *Approach I: C generates the watermark and proves to D that the watermark is well-formed.* This approach requires C to prove to D that the generated watermark is of a certain structure. This is to prevent C from generating an ill-formed watermark that can easily be removed from marked content at a later stage. In the existing protocols, the proving process is performed based on a zero-knowledge proof of knowledge that uses a homomorphic commitment scheme (Section 2.3.6). The general idea is that C commits to the watermark and sends the committed watermark to D. C then opens some of the commitments selected by D to prove in zero-knowledge that the committed values are of the structure required by D. If this is the case then D embeds the watermark into content using the committed watermark. This operation is identical to watermarking in the encrypted domain, as described in Section 2.3.3. Hence D does not know the watermark.

  One general issue with this approach is the extra computation required to perform the zero-knowledge proof of knowledge. Protocols known as *asymmetric fingerprinting schemes* and *anonymous fingerprinting schemes* [12, 19, 26, 39, 40, 78, 102, 103, 104, 105, 106, 118] use this approach.

- *Approach II: C freely generates any watermark.* The second approach, which is proposed in more recent protocols [33, 53, 54, 65, 66, 107, 138, 139], lets C generate the watermark without the need to convince D that the generated watermark is well-formed. The mechanism is for C to generate and encrypt the watermark with a homomorphic encryption scheme and send the encrypted watermark to D. Next D embeds the watermark into content based on watermarking in the encrypted domain.

We will discuss the Pfitzmann-Schunter protocol [104] in Section 4.2 as an example of a protocol that represents approach I. This is because subsequent proposals using

this approach follow the same zero-knowledge techniques discussed in this protocol. We also note the existence of a proposal by Kuribayashi and Tanaka [80] aimed at improving the efficiency of protocols using this approach in Section 4.2.1. However, this proposal has been found to contain flaws [136].

Approach II is simpler and is more efficient, but faces certain security issues. This can be observed in the two protocols proposed by Ibrahim *et al.* [65, 66], which we examine in Section 4.3. From our study of these two protocols, certain attacks that we defined, and attacks defined by Williams, Treharne and Ho [134], can be successful against these, and other protocols in this category.

In view of the issues concerning these two approaches, we suggest an alternative approach that does not face the computational performance issue of approach I and avoid the security issues of approach II. As we observe in Section 4.4, this new approach requires a stronger trust assumption, where the distributor is trusted more than the client. Hence it cannot be regarded as a standard FaCT protocol where the distributor and the client are not trusted, but only a "semi-fair" one.

We conclude by observing that it seems to be hard to construct a *secure* and *efficient* FaCT protocol without a special trusted third party. The same can be said for constructing a FaCT protocol without any control mechanism that allows the distributor to ensure that the watermark generated by the client is well-formed.

4.2 The Pfitzmann-Schunter Protocol

Pfitzmann and Schunter (PS) proposed a protocol known as an *asymmetric fingerprinting scheme* in [104].

Fundamentals. The PS protocol involves D, C, a CA and A. The CA is fully trusted. It provides traceability, framing resistance and non-repudiation of redistribution.

Environment. The PS protocol implicitly assumes that both D and C possess ample computing resources since both of them need to conduct many rounds of a zero-knowledge proof process and also compute homomorphic bit commitments. It also requires a secure communication channel and public key support. As op-

posed to the MW protocol described in Section 3.7, there is no special trusted third party in this protocol. The main building blocks are digital watermarking schemes, zero-knowledge proof systems, homomorphic bit commitment schemes and digital signature schemes. Table 4.1 states the design criteria of the PS protocol.

Table 4.1: The Design Framework of the PS Protocol

Fundamentals	
Parties Involved	C, D, CA, A
Trust Assumptions	CA is *fully trusted*
Security Properties	Traceability (**TR**), Framing resistance (**FR**), Non-repudiation of redistribution (**NR**)
Environment	
Comp. Resources	Implicitly assumed D and C have ample resources
Sec. comm. Support	Required
Pub. Key Support	Required
TTPs	No special TTP
Building Blocks	Digital watermarking scheme, homomorphic bit commitment scheme, zero-knowledge (ZK) proof and digital signature scheme

In the following we describe the three phases of the PS protocol.

`Initial Setup.` In this phase C and D obtain authenticated public keys the CA. Figure 4.1 shows the protocol messages, and we describe the protocol steps below:

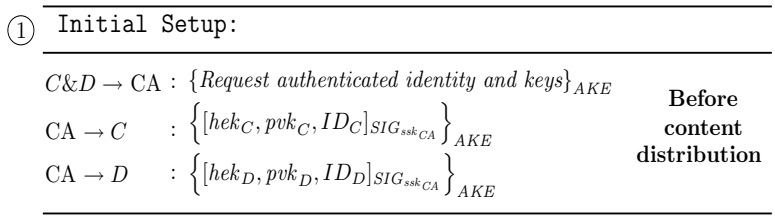

Figure 4.1: PS Protocol – Initial Setup

(I) C and D register with the CA to obtain authenticated public keys.

1. C generates a key pair (pvk_C, ssk_C) based on a digital signature scheme. He also generates a key pair (hek_C, hdk_C) based on a homomorphic bit commitment scheme. C sends the public verification key pvk_C and the public commitment key hek_C to the CA. The same process is also followed by D.

(II) The CA generates and provides C and D with the authenticated key materials.

2. The CA, upon verifying the identity information provided by C and D, provides both parties with authenticated public keys. This means that digital signatures are produced on these keys. These signatures can later be used by C and D to prove the validity of their respective public keys to others.

`Content Watermarking and Distribution.` In this phase D produces marked content and C receives the content. Figure 4.2 illustrates the protocol messages.

② Content Watermarking and Distribution:	
$C \rightarrow D$:	$\left\{[W]_{COM_{hek_C}}, AGR, [[W]_{COM_{hek_C}}, AGR]_{SIG_{ssk_C}}\right\}_{AKE}$
$D \rightarrow C$:	$\{seq\_no\}_{AKE}$
$C \leftrightarrow D$:	$\{C$ proves to D in *zero-knowledge* that he knows $ID_C\}_{AKE}$
$D \rightarrow C$:	$\left\{[X']_{COM_{hek_C}}\right\}_{AKE}$ **Content distribution**

Figure 4.2: PS Protocol – Content Watermarking and Distribution

Below we discuss the protocol steps:

(I) C requests content, generates a client watermark and approves a content agreement with D.

1. For a first time client, D runs a *place_marks* algorithm. This algorithm selects a set of positions in the content (i.e. a black-and-white image) that are suitable for watermark embedding For example, if the image contains pixel 1 to pixel L, then these *positions* are a subset of these pixels (We remark that in current watermarking techniques, embedding is generally performed under a transform domain, such as the DCT as stated in Figure 2.1, instead of embedding in pixels as in the case of the PS protocol). We represent the positions as $X = (x_1, \ldots, x_n)$. In addition, D initialises an l_1-bit length counter *seq_no* to zero.

2. D sends *seq_no* to C and increments *seq_no*.

3. C randomly chooses a string *ID_proof* $\in \{0,1\}^{l_2}$, where l_2 represents the length of the string. Then C sets his identity as ID_C = (*seq_no*, *ID_proof*). We further denote $ID_C = (id_1, \ldots, id_{l_3})$, where $l_3 = l_1 + l_2$. C encodes ID_C into a binary string $W = (w_1, \ldots, w_n)$ based on an encoding algorithm, for example, the algorithm proposed by Boneh and Shaw in [15]. This is effectively the watermark to be embedded into content. C then commits to the watermark

W, based on the homomorphic bit commitment scheme presented in Figure 2.8, by committing to each element of the watermark as:

$$[w_i]_{COM_{hek_C}} \qquad 1 \leq i \leq n.$$

We denote $[W]_{COM_{hek_C}} = ([w_1]_{COM_{hek_C}}, \ldots, [w_n]_{COM_{hek_C}})$. C also signs $[W]_{COM_{hek_C}}$ and a content agreement AGR that contains a description of the content that C wishes to obtain. This results in a signature:

$$[[W]_{COM_{hek_C}}, AGR]_{SIG_{ssk_C}}.$$

C sends $[W]_{COM_{hek_C}}$, AGR and $[[W]_{COM_{hek_C}}, AGR]_{SIG_{ssk_C}}$ to D.

4. C also proves in zero-knowledge that he knows ID_C, where the first part of ID_C is *seq_no*, and the second part is *ID_proof*. To do this, C can open the first l_1 commitments to reveal *seq_no*. After that, C proves that the commitments on the coding of *ID_proof* contain a valid code that conform to the requirements of the underlying coding algorithm. For example, if *ID_proof* contains a number of zeros and ones, then C arbitrarily generates a sequence of commitments containing zeros and ones, and proves to D that he can map the commitments of *ID_proof* to this sequence of commitments. D can be assured that ID_C is well-formed after many rounds of the above process. More details on the zero-knowledge proof process are provided by Pfitzmann and Schunter in [104].

(II) D produces a marked copy of the requested content and sends it to C.

5. D verifies $[[W]_{COM_{hek_C}}, AGR]_{SIG_{ssk_C}}$. After that, D commits to each element of the content $X = (x_1, \ldots, x_n)$ with the same homomorphic bit commitment scheme used by C as:

$$[x_i]_{COM_{hek_C}} \qquad 1 \leq i \leq n.$$

We denote $[X]_{COM_{hek_C}} = ([x_1]_{COM_{hek_C}}, \ldots, [x_n]_{COM_{hek_C}})$. Next, D generates the marked copy of content by multiplying every element of $[W]_{COM_{hek_C}}$ by every element of $[X]_{COM_{hek_C}}$:

$$\left.\begin{array}{rl} & [x_i]_{COM_{hek_C}} \cdot [w_i]_{COM_{hek_C}} \\ = & [x_i \oplus w_i]_{COM_{hek_C}} \\ = & [x'_i]_{COM_{hek_C}} \end{array}\right\} \; 1 \leq i \leq n.$$

We represent the result of this multiplication (and embedding) as

$$[X']_{COM_{hek_C}} = ([x'_1]_{COM_{hek_C}}, [x'_2]_{COM_{hek_C}}, \ldots, [x'_n]_{COM_{hek_C}}).$$

Finally, D sends $[X']_{COM_{hek_C}}$ to C.

6. C decrypts $[X']_{COM_{hek_C}}$ and obtains $X' = (x_1 \oplus w_1, \ldots, x_n \oplus w_n)$.

`Identification and Dispute Resolution.` In this phase, D determines from the found copy of content the identity of a dishonest C who illegally distributed content. D further proves this fact to A. This phase also involves C, since C is the only party that knows the watermark. Figure 4.3 shows the protocol messages.

③ `Identification and Dispute Resolution:`

D	:	$x_i \oplus w_i \oplus x_i = w_i$, for $1 \leq i \leq n$ *seq_no* detected from w_i?	**After content distribution**
$D \rightarrow A$	:	$\left\{W, [W]_{COM_{hek_C}}, AGR, [[W]_{COM_{hek_C}}, AGR]_{SIG_{ssk_C}}\right\}_{AKE}$	
$A \rightarrow C$	:	$\{$*watermark info*?$\}_{AKE}$	
$C \rightarrow A$	:	$\{C$ proves to A in *zero-knowledge* that he is innocent$\}_{AKE}$	

Figure 4.3: PS Protocol – Identification and Dispute Resolution

In the following we describe the protocol steps:

(I) D detects a watermark from the found copy of content in order to identify the client that owns the content.

1. When an illegal copy of content $\widehat{X} = (\widehat{x}_1, \ldots, \widehat{x}_n)$ is found, D uses the original content X and the positions to extract the embedded watermark W. This can be performed by bit-wise XORing ($\oplus$) the original content X and the found content $\widehat{X}$ at the positions to retrieve W. Mathematically, this is $x_i \oplus w_i \oplus \widehat{x}_i = w_i$, for $1 \leq i \leq n$, assuming $x_i = \widehat{x}_i$. If the first part of the decoded W matches *seq_no* then D can be sure that $\widehat{X}$ belongs to C.

(II) D proves to A that C illegally distributed copies of content.

2. Next, D proves to A the illegal act of C by providing evidence to A. These are the extracted watermark W, the committed watermark $[W]_{COM_{hek_C}}$, the agreement AGR and the client signature $[[W]_{COM_{hek_C}}, AGR]_{SIG_{ssk_C}}$.

3. A first verifies the validity of the signature $[[W]_{COM_{hek_C}}, AGR]_{SIG_{ssk_C}}$. If this signature is valid, arbiter A needs to verify that $[W]_{COM_{hek_C}}$ is the commitment of W. Since A cannot verify this without the help of C, A asks C to prove his innocence. C can open his commitments and show that these are different from the extracted W to demonstrate that he is innocent. If C does not want to expose his identity string *ID_proof*, he can use a zero-knowledge proof of knowledge to show to arbiter A that at least one commitment in $[W]_{COM_{hek_C}}$ is not the bit in W. C will be declared guilty if he fails in this proving process.

4.2.1 Improvement Attempts by Kuribayashi and Tanaka

In view of the relatively heavy computation required due to bit commitments and zero-knowledge proofs, Kuribayashi and Tanaka [80] proposed a more efficient protocol. The proposed protocol modifies the process of the zero-knowledge proof so that homomorphic bit commitment schemes can be replaced by the Okamoto-Uchiyama homomorphic encryption scheme [97]. The improvement is twofold. Firstly, the zero-knowledge proof is performed in two protocol messages instead of y rounds of protocol messages between C and D, as required in the PS protocol. Secondly, instead of committing the content and watermark bit-by-bit, the content and watermark can be encrypted as integers using the Okamoto-Uchiyama scheme, hence increasing the encryption rate. However, Wu [136] demonstrated that the proposal by Kuribayashi and Tanaka is flawed and is not zero-knowledge, since D is able to extract the hidden information. We will not discuss further the technical details of the Kuribayashi and Tanaka proposal or Wu's attacks. What we aim to highlight here is that it seems to be difficult to design a secure and efficient protocol in approach I without using zero-knowledge proof systems.

4.3 The Ibrahim-ElDin-Hegazy Protocols

Ibrahim, ElDin and Hegazy (IEH) proposed two protocols following approach II. We denote these protocols by IEH-1 [65] and IEH-2 [66]. These protocols aimed to improve existing protocols such as the MW protocol presented in Section 3.7 by modifying the design and introducing new properties. One such modification is to allow C to freely generate the watermark W. However, modification can come at

the expense of existing safeguards. We will show that this is precisely the case for these two protocols. This work was published in [111, 113].

Fundamentals. Both protocols involve C, D, a CA and A. The CA and A are fully trusted. There is an additional party known as the reseller R in IEH-2, who plays the role of a reselling agent that obtains content from D and provides this content to C. Both protocols claimed to provide traceability, framing resistance and non-repudiation of redistribution.

Environment. These protocols assume that C and D have ample computing resources. They also rely on public key support. However, they do not assume secure communication support, but rather depend on the construction of the protocols to withstand attacks on the communication channel. The protocols do not require any special trusted third party. The main building blocks are digital watermarking schemes, homomorphic encryption schemes and digital signature schemes. Table 4.2 shows the design framework of the IEH protocols.

Table 4.2: The Design Framework of the IEH Protocols

Fundamentals	
Parties Involved	C, D, CA, A, (R)
Trust Assumptions	CA and A are *fully trusted*
Security Properties	Traceability (**TR**), Framing resistance (**FR**), Non-repudiation of redistribution (**NR**)
Environment	
Comp. Resources	Implicitly assumed D and C have ample resources
Sec. comm. Support	Depends on protocol design
Pub. Key Support	Required
TTPs	No special TTP
Building Blocks	Digital watermarking scheme, homomorphic encryption scheme and digital signature scheme

In addition to the three security properties listed in Table 4.2, IEH-1 and IEH-2 further claimed to address the following "problems":

- *Conspiracy problem*, which refers to the possibility for a distributor to *conspire* with a third party (e.g. a WCA) in order to reveal the client's watermark. By revealing this watermark, it is then possible for the distributor to frame the client by embedding the watermark into content and distributing copies of it.
- *Unbinding problem*, in which, given a found illegal marked content, the distrib-

utor can extract the watermark, re-embed this watermark into a more valuable content and accuse the client of illegally distributing copies of the found content and the more valuable content. This is possible when the watermark is not *bound* to the content itself.

- *Client's participation in the dispute resolution problem.* This issue refers to the assumption that during dispute of illegal distribution, the distributor is solely responsible for proving the guilt of the client to a third party, and the client should not be required to participate in such a process.
- *Man in the middle attack.* This issue refers to the ability of an adversary to insert and modify the messages in transmission, without either the distributor or the client knowing that the communication channel has been compromised.
- *Practice applicability problem.* This issue refers to the need for the client to contact not just the distributor, but also another party to obtain the content, which is inconvenient for the client.

4.3.1 The First Ibrahim-ElDin-Hegazy Protocol

This protocol, IEH-1, is intended for the secure selling of digital content between D and C. It involves four parties. These are D, C, the CA and A. The CA is a fully trusted third party who provides public key support as described in Section 3.4.2. In the following we describe the three phases of the protocol.

`Initial Setup.` This phase is similar to the `Initial Setup` phase of the PS protocol described in the previous section. Figure 4.4 shows the protocol messages between D, C and the CA.

① `Initial Setup:`		
$C\&D \rightarrow$ CA	: *Request authenticated keys*	**Before content distribution**
CA $\rightarrow C$	: $Cert_{ssk_{CA}}(ID_C)$	
CA $\rightarrow D$	: $Cert_{ssk_{CA}}(ID_D)$	

Figure 4.4: IEH-1 – Initial Setup

The protocol steps are as follows:

(I) C and D register with the CA to obtain authenticated public keys.

1. C and D generate signature key pairs (pvk_C, ssk_C) and (pvk_D, ssk_D) based on a digital signature scheme, and encryption key pairs (hek_C, hdk_C) and (hek_D, hdk_D) based on a homomorphic encryption scheme. The public keys pvk_C, pvk_D, hek_C and hek_D are sent to the CA.

(II) The CA generates and provides C and D with the authenticated key materials.

2. The CA signs the public verification and encryption keys of all parties involved so that any parties using these keys can be sure that the keys are authentic. The public keys and the generated signatures are published for public access by all parties involved. These public keys and the signatures generated by the CA, together with identity information, is known as the certificate $Cert_{ssk}(.)$.

`Content Watermarking and Distribution.` This is the main phase for content purchase and watermarking. In brief, D embeds the watermark into content in the encrypted domain and sends the encrypted marked content to C. We describe the protocol steps in the following and the protocol messages are shown in Figure 4.5.

② Content Watermarking and Distribution:		
$C \rightarrow D$	: $[H(AGR)]_{SIG_{ssk_C}}$,$[W]_{HE_{hek_C}}$, $[[W]_{SIG_{ssk_C}}]_{HE_{hek_{CA}}}$, $[H(H(W), H(AGR))]_{SIG_{ssk_C}}$,$Cert_{ssk_{CA}}(ID_C)$	
$D \rightarrow$ CA	: $[[W]_{SIG_{ssk_C}}]_{HE_{hek_{CA}}}$,$Cert_{ssk_{CA}}(ID_C)$	**Content distribution**
CA $\rightarrow D$	: $[[W]_{HE'_{hek_C}}]_{SIG_{ssk_{CA}}}$	
$D \rightarrow C$	: $[[X'']_{HE_{hek_C}}]_{SIG_{ssk_D}}$,$Cert_{ssk_{CA}}(ID_D)$	

Figure 4.5: IEH-1 – Content Watermarking and Distribution

(I) C requests content, generates a client watermark and approves a content agreement with D.

1. C initiates the protocol by sending a purchase request to D.
2. D sends his certificate $Cert_{ssk_{CA}}(ID_D)$ to C.
3. A purchase agreement AGR is approved between C and D. This agreement states the rights, obligations and specifies content X.

4. C generates hash value $H(AGR)$. This hash value can be generated using a cryptographic hash function $H(.)$ such as SHA-2 [70] or RIPEMD-160 [37]. C then generates a signature $[H(AGR)]_{SIG_{ssk_C}}$. This signature allows A to confirm C's purchase during a dispute.

5. C generates a watermark W and signs it as $[W]_{SIG_{ssk_C}}$. This is further encrypted using CA's encryption key as $[[W]_{SIG_{ssk_C}}]_{HE_{hek_{CA}}}$. During a dispute, D will send this encrypted object to CA so that C need not participate in the dispute resolution phase.

6. C encrypts W, resulting in $[W]_{HE_{hek_C}}$.

7. C generates $[H(H(W), H(AGR))]_{SIG_{ssk_C}}$. The purpose of this signature is to bind W to AGR.

8. C sends the signature $[H(AGR)]_{SIG_{ssk_C}}$, the encrypted watermark $[W]_{HE_{hek_C}}$, the encrypted signature $[[W]_{SIG_{ssk_C}}]_{HE_{hek_{CA}}}$, the signature binding the watermark and the agreement $[H(H(W), H(AGR))]_{SIG_{ssk_C}}$, and C's certificate $Cert_{ssk_{CA}}(ID_C)$ to D.

(II) D produces a marked copy of the requested content and sends it to C.

9. D forwards the encrypted signature $[[W]_{SIG_{ssk_C}}]_{HE_{hek_{CA}}}$ and the client certificate $Cert_{ssk_{CA}}(ID_C)$ to the CA. Next the CA decrypts the encrypted signature $[[W]_{SIG_{ssk_C}}]_{HE_{hek_{CA}}}$ to obtain the signature $[W]_{SIG_{ssk_C}}$, which is then verified to obtain the watermark W. After that, the CA re-encrypts W with C's encryption key as $[W]_{HE'_{hek_C}}$ and signs it to obtain $[[W]_{HE'_{hek_C}}]_{SIG_{ssk_{CA}}}$. This signature is sent to D. This is to prevent C from encrypting watermark W in $[W]_{HE_{hek_C}}$ while including a different watermark W' in $[[W']_{SIG_{ssk_C}}]_{HE_{hek_{CA}}}$.

10. When the CA's message is received, D first verifies $[[W]_{HE'_{hek_C}}]_{SIG_{ssk_{CA}}}$. Next, D generates the hash value $H([W]_{HE_{hek_C}})$ and the hash value $H([W]_{HE'_{hek_C}})$. These two hash values are compared and, if they are identical, then D continues to run the protocol. If they are not identical, the protocol is halted.

11. D generates a unique watermark V and embeds it into content X. The computation is:

$$X' \leftarrow [X, V]_{EMB_{wmk_V}}.$$

12. D generates an encrypted marked content as follows, using the watermarking in the encrypted domain scheme described in Section 2.3.3:

$$\left.\begin{array}{rl} & [x'_i]_{HE_{hek_C}} \cdot [w_i]_{HE_{hek_C}} \\ = & [x'_i \circ w_i]_{HE_{hek_C}} \\ = & [x''_i]_{HE_{hek_C}} \end{array}\right\} \; 1 \leq i \leq n,$$

where $\circ$ represents either modular addition, modular multiplication or bit-wise XOR, depending on the underlying homomorphic encryption used. We denote $[X'']_{HE_{hek_C}} = ([x''_1]_{HE_{hek_C}}, [x''_2]_{HE_{hek_C}}, \ldots, [x''_n]_{HE_{hek_C}})$.

13. D stores in the database: V, AGR, $[H(AGR)]_{SIG_{ssk_C}}$, $[[W]_{SIG_{ssk_C}}]_{HE_{hek_{CA}}}$, $[H(H(W), H(AGR))]_{SIG_{ssk_C}}$ and $Cert_{ssk_{CA}}(ID_C)$.

14. D generates and sends the signature $[[X'']_{HE_{hek_C}}]_{SIG_{ssk_D}}$ and D's certificate $Cert_{ssk_{CA}}(ID_D)$ to C.

15. C verifies $[[X'']_{HE_{hek_C}}]_{SIG_{ssk_D}}$ to retrieve $[X'']_{HE_{hek_C}}$ and then decrypts it to obtain X''.

`Identification and Dispute Resolution.` When an illegal copy $\widehat{X}$ is found, D starts this phase to prove that a dishonest C distributed $\widehat{X}$. Figure 4.6 illustrates the protocol messages and the protocol steps are described as follows:

③ `Identification and Dispute Resolution:`		
D	: {`true`, `false`} $\leftarrow [\widehat{X}, V, X]_{DET_{wmk}}$	
$D \rightarrow A$	: $Cert_{ssk_{CA}}(ID_C)$,$[H(H(W), H(AGR))]_{SIG_{ssk_C}}$, $[H(AGR)]_{SIG_{ssk_C}}$,V,AGR, $[[W]_{SIG_{ssk_C}}]_{HE_{hek_{CA}}}$, $\widehat{X}$,X'	**After content distribution**
$A \rightarrow$ CA	: $[[W]_{SIG_{ssk_C}}]_{HE_{hek_{CA}}}$,$Cert_{ssk_{CA}}(ID_A)$	
CA $\rightarrow A$	: $[[W]_{SIG_{ssk_C}}]_{HE_{hek_A}}$	
A	: {`true`, `false`} $\leftarrow [\widehat{X}, W, X']_{DET_{wmk}}$ verify $[H(AGR)]_{SIG_{ssk_C}}$ verify $[H(H(W), H(AGR))]_{SIG_{ssk_C}}$	

Figure 4.6: IEH-1 – Identification and Dispute Resolution

(I) D detects a watermark from the found copy of content in order to identify the client that owns the content.

1. D detects the watermark V from the illegal copy $\widehat{X}$ using a watermarking detection algorithm corresponding to the embedding process. If V is detected,

C's certificate $Cert_{ssk_{CA}}(ID_C)$, two signatures, $[H(H(W), H(AGR))]_{SIG_{ssk_C}}$ and $[H(AGR)]_{SIG_{ssk_C}}$, the watermark V, the agreement AGR, the encrypted signature $[[W]_{SIG_{ssk_C}}]_{HE_{hek_{CA}}}$, the illegal copy $\widehat{X}$ and the marked content X' are sent to A.

(II) D proves to A that C illegally distributed copies of content.

2. A forwards the encrypted signature $[[W]_{SIG_{ssk_C}}]_{HE_{hek_{CA}}}$ and $Cert_{ssk_{CA}}(ID_A)$ to the CA. The CA decrypts $[[W]_{SIG_{ssk_C}}]_{HE_{hek_{CA}}}$ and re-encrypts the retrieved signature, $[W]_{SIG_{ssk_C}}$, with A's encryption key, resulting in $[[W]_{SIG_{ssk_C}}]_{HE_{hek_A}}$. This encrypted object is sent back to A. Then A decrypts this encrypted signature $[[W]_{SIG_{ssk_C}}]_{HE_{hek_A}}$ to retrieve $[W]_{SIG_{ssk_C}}$ and verifies it to obtain the watermark W.
3. A detects the watermark W from the illegal copy $\widehat{X}$.
4. If the watermark W is detected, A verifies signature $[H(AGR)]_{SIG_{ssk_C}}$ based on the agreement AGR provided by D. If the verification is successful, the protocol continues. Otherwise it halts.
5. As the final step, A verifies signature $[H(H(W), H(AGR))]_{SIG_{ssk_C}}$ using the watermark W and the agreement AGR given by D. If the verification is successful, which proves C bought the content, then C is found guilty.

Figure 4.7 shows the flow diagram of all three phases of the protocol. It provides a comparison of the original protocol flows with that of Figure 4.10, where we discuss one of our attacks in Section 4.3.3.

4.3.2 The Second Ibrahim-ElDin-Hegazy Protocol

This protocol, IEH-2, is similar to the previous protocol except that it involves a legitimate reseller R, who acts as an agent for D and sells content bought from D to C. The main differences between this protocol and the previous one are:

- the creation of an object license OL by D to monitor the selling of content by R. Each time R wants to sell content, D generates a new object license OL' counting down the number of resells allowed.

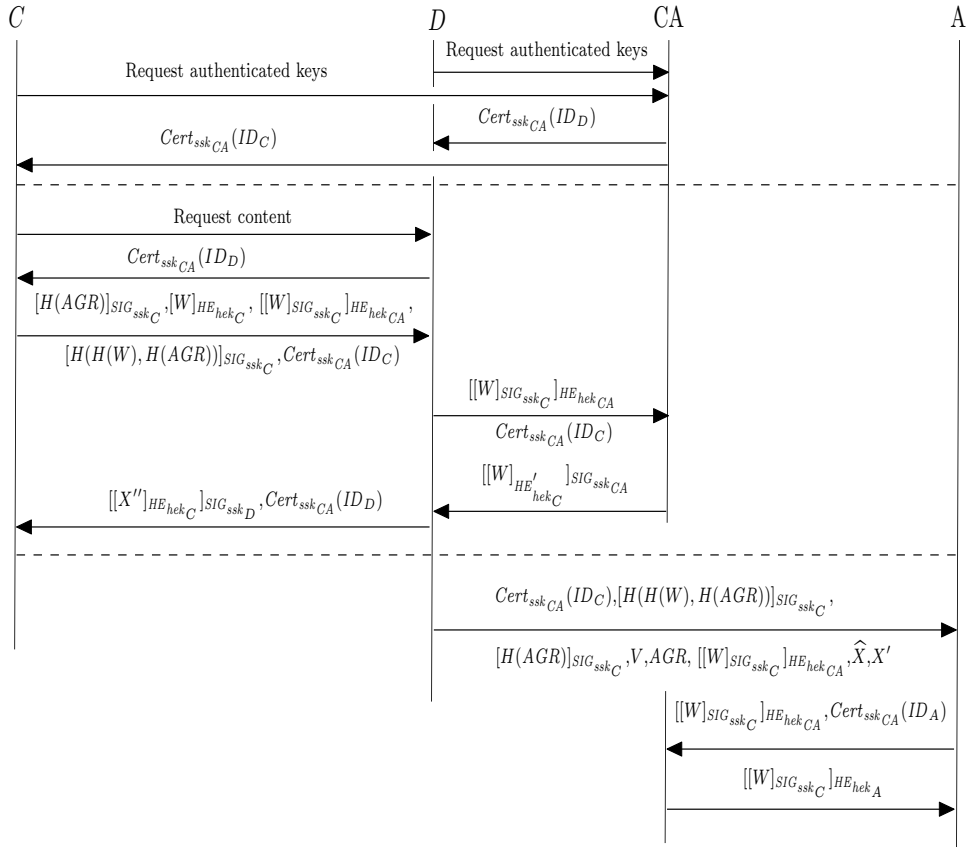

Figure 4.7: IEH-1 – Protocol Flows Diagram for All Three Phases

- Instead of sending messages to D, C sends messages to R, who then contacts D.

We do not provide further details of this protocol since our analysis works on both protocols in a similar way. The protocol is illustrated in Figure 4.8 and the flows diagram in Figure 4.9. As can be observed, the protocol flows between D, the CA and A are identical with IEH-1.

4.3.3 Flaws in the Protocols

In this section we analyse the two IEH protocols by demonstrating attacks that can be mounted on them [111].

①	`Initial Setup:`		
	$C\&D \rightarrow$ CA	: *Request authenticated keys*	**Before content distribution**
	CA $\rightarrow C$	: $Cert_{ssk_{CA}}(ID_C)$	
	CA $\rightarrow D$	: $Cert_{ssk_{CA}}(ID_D)$	
②	`Content Watermarking and Distribution:`		
	$C \rightarrow R$	: *request content*	**Content distribution**
	$R \rightarrow C$	: $Cert_{ssk_{CA}}(ID_R)$	
	$C \rightarrow R$	: $[H(AGR)]_{SIG_{ssk_C}}$,$[W]_{HE_{hek_C}}$, $[[W]_{SIG_{ssk_C}}]_{HE_{hek_{CA}}}$, $[H(H(W), H(AGR))]_{SIG_{ssk_C}}$,$Cert_{ssk_{CA}}(ID_C)$	
	$R \rightarrow D$	: $[H(AGR)]_{SIG_{ssk_C}}$,$[W]_{HE_{hek_C}}$, $[[W]_{SIG_{ssk_C}}]_{HE_{hek_{CA}}}$, $[H(H(W), H(AGR))]_{SIG_{ssk_C}}$,$Cert_{ssk_{CA}}(ID_C)$ $[OL]_{SIG_{ssk_R}}$,$[H(AGR)]_{SIG_{ssk_R}}$,$Cert_{ssk_{CA}}(ID_R)$, AGR	
	$D \rightarrow$ CA	: $[[W]_{SIG_{ssk_C}}]_{HE_{hek_{CA}}}$,$Cert_{ssk_{CA}}(ID_C)$	
	CA $\rightarrow D$	: $[[W]_{HE'_{hek_C}}]_{SIG_{ssk_{CA}}}$	
	$D \rightarrow R$	: $[[X'']_{HE_{hek_C}}]_{SIG_{ssk_D}}$,$[[OL']_{HE_{hek_R}}]_{SIG_{ssk_D}}$	
	$R \rightarrow C$	: $[[X'']_{HE_{hek_C}}]_{SIG_{ssk_D}}$,$Cert_{ssk_{CA}}(ID_D)$	
③	`Identification and Dispute Resolution:`		
	D	: $\{\texttt{true}, \texttt{false}\} \leftarrow [\widehat{X}, V, X]_{DET_{wmk}}$	
	$D \rightarrow A$	: $Cert_{ssk_{CA}}(ID_C)$,$[H(H(W), H(AGR))]_{SIG_{ssk_C}}$, $[H(AGR)]_{SIG_{ssk_C}}$,V,AGR, $[[W]_{SIG_{ssk_C}}]_{HE_{hek_{CA}}}$, $\widehat{X}$,X',$[H(AGR)]_{SIG_{ssk_R}}$,$Cert_{ssk_{CA}}(ID_R)$	**After content distribution**
	$A \rightarrow$ CA	: $[[W]_{SIG_{ssk_C}}]_{HE_{hek_{CA}}}$,$Cert_{ssk_{CA}}(ID_A)$	
	CA $\rightarrow A$	: $[[W]_{SIG_{ssk_C}}]_{HE_{hek_A}}$	
	A	: $\{\texttt{true}, \texttt{false}\} \leftarrow [\widehat{X}, W, X']_{DET_{wmk}}$ verify $[H(AGR)]_{SIG_{ssk_R}}$ verify $[H(AGR)]_{SIG_{ssk_C}}$ verify $[H(H(W), H(AGR))]_{SIG_{ssk_C}}$	

Figure 4.8: IEH-2

Attack 1: Client-generate-watermark Attack on both protocols. The idea behind this attack is for C to remove the watermark W from the marked content X'' that C received from D. The attack will be successful on Ibrahim *et al.*'s protocols since in these protocols C *generates watermark* W. In fact, such an attack was mentioned by Memon and Wong [94], who recommended that C should not generate watermark W due to the possibility of such an attack. This is the main reason that the WCA is used in [94] to generate W. However, Ibrahim *et al.*, in an attempt to prevent the conspiracy problem as previously stated in Section 4.3, removed the WCA without giving a solution as to how to prevent a dishonest C from generating an ill-formed watermark.

Attack 2: Client-in-the-middle Attack on both protocols. The idea behind this attack is for C to modify protocol messages and generate a different watermark

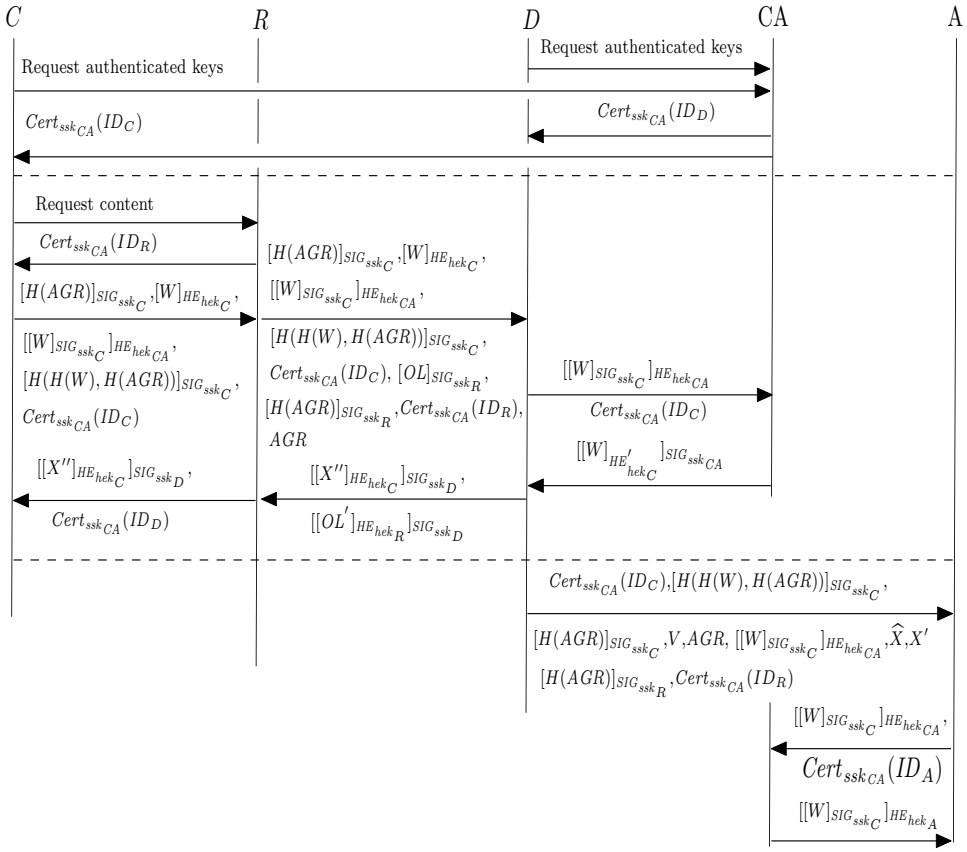

Figure 4.9: IEH-2 – Protocol Flows Diagram for All Three Phases

for a different message, so that D will fail to prove C's act of distributing content illegally, even when C generates a proper watermark W.

In the following we demonstrate how the attack works on Ibrahim *et al.*'s protocols by modifying the protocol steps given in the `Content Watermarking and Distribution` phase in Section 4.3.1, starting from Step 5. We note that the same attack can be deployed against the second protocol. Figure 4.10 further illustrates this attack on both protocols.

5′. Different from the original proposal, C generates three watermarks W_1, W_2 and W_3, instead of one watermark. C then generates two signatures, $[W_1]_{SIG_{ssk_C}}$ and $[W_2]_{SIG_{ssk_C}}$. After that, C encrypts $[W_1]_{SIG_{ssk_C}}$ and $[W_2]_{SIG_{ssk_C}}$, resulting in $[[W_1]_{SIG_{ssk_C}}]_{HE_{hek_{CA}}}$ and $[[W_2]_{SIG_{ssk_C}}]_{HE_{hek_{CA}}}$.

6′. C encrypts W_2 as $[W_2]_{HE_{hek_C}}$.

7′. C generates signature $[H(H(W_3), H(AGR))]_{SIG_{ssk_C}}$. Note that watermark W_3 is hashed and signed instead.

8′. C sends the signatures $[H(AGR)]_{SIG_{ssk_C}}$ and $[H(H(W_3), H(AGR))]_{SIG_{ssk_C}}$, $[W_2]_{HE_{hek_C}}$, $[[W_1]_{SIG_{ssk_C}}]_{HE_{hek_{CA}}}$, together with $Cert_{ssk_{CA}}(ID_C)$ to D (to R for Protocol II).

9′. **When D sends $[[W_1]_{SIG_{ssk_C}}]_{HE_{hek_{CA}}}$ to the CA, C intercepts the message and sends $[[W_2]_{SIG_{ssk_C}}]_{HE_{hek_{CA}}}$ instead**. $[[W_2]_{SIG_{ssk_C}}]_{HE_{hek_{CA}}}$ is decrypted by the CA to obtain the signature $[W_2]_{SIG_{ssk_C}}$, which is then verified to obtain W_2. Next the CA re-encrypts W_2 with C's encryption key as $[W_2]_{HE'_{hek_C}}$ and signs it to obtain $[[W_2]_{HE'_{hek_C}}]_{SIG_{ssk_{CA}}}$. This signature is sent to D. To avoid C from being able to replace the message, D may sign the message before sending it to the CA.

10′. D first verifies $[[W_2]_{HE'_{hek_C}}]_{SIG_{ssk_{CA}}}$ to obtain $[W_2]_{HE'_{hek_C}}$. Next, D compares $H([W_2]_{HE_{hek_C}}) = H([W_2]_{HE'_{hek_C}})$. Since $[W_2]_{HE'_{hek_C}}$ is identical to $[W_2]_{HE_{hek_C}}$ given by C. The comparison will be true. D continues the protocol since both hash values are identical.

12′. Subsequently the client watermark that is embedded into content is W_2, which is different from W_1 in $[[W_1]_{SIG_{ssk_C}}]_{HE_{hek_{CA}}}$ possessed by D.

Following from the above steps, recall from the `Identification and Dispute Resolution` phase that when an illegal content $\widehat{X}$ is found, one of the objects sent by D to the CA is $[[W_1]_{SIG_{ssk_C}}]_{HE_{hek_{CA}}}$. For C to be found guilty, the CA retrieves W_1 from $[[W_1]_{SIG_{ssk_C}}]_{HE_{hek_{CA}}}$, signs it and re-encrypts it as $[[W_1]_{SIG_{ssk_C}}]_{HE_{hek_A}}$ and passes this new encrypted object to A. Upon receiving the encrypted object, A decrypts and retrieves W_1. Next A runs the detection algorithm, expecting to detect W_1 from $\widehat{X}$. However, due to the interception by C in Step 9′ above, the watermark that is embedded in this content is W_2, instead of W_1. Hence A will fail to detect C's watermark and will declare C innocent. We also note that the reason that this attack can be deployed is due to the protocols' introduction of Step 8 and Step 9 in the `Content Watermarking and Distribution` phase in order to avoid the *client's participation in the dispute resolution problem.*

Furthermore, due to Step 5′ and Step 7′, a third watermark W_3 is used by C to generate the signature $[H(H(W_3), H(AGR))]_{SIG_{ssk_C}}$. A will not be able to match

the watermark W_2 extracted from the illegal copy $\widehat{X}$ to the purchase agreement based on this signature, and thus cannot be certain that C bought this content.

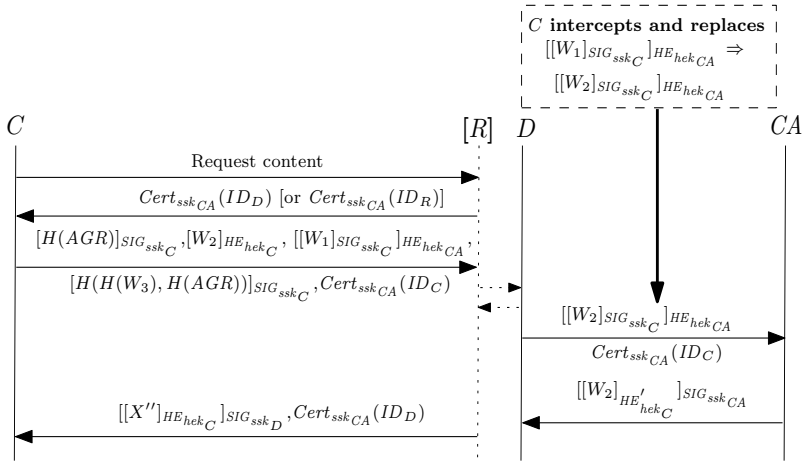

Figure 4.10: IEH Protocols: Attack 2

Attack 3: Distributor-CA Conspiracy Attack on Both Protocols. The main idea behind this attack is based on the fact that the CA *knows* the client watermark W, and essentially has similar responsibility to that of a WCA (which generates the client's watermark) in other protocols such as the protocol in [85].

This can be seen from Step 9 of the `Content Watermarking and Distribution` phase. D sends the CA an encrypted signature $[[W]_{SIG_{ssk_C}}]_{HE_{hek_{CA}}}$ that contains W, and the CA is tasked to retrieve the watermark W, re-encrypt it and sign it with the CA's signing key. Hence the CA can store a copy of W when retrieving it from the signature, and then sends W to D. Thus the claim of avoiding the *conspiracy problem* fails.

4.3.4 Williams-Treharne-Ho Analysis of the Protocols

Independently, Williams, Treharne and Ho [134] discussed another flaw in the IEH protocols. Using formal analysis based on Communicating Sequential Processes (CSP) [63], Williams *et al.* discovered an unbinding attack on the protocols. This refutes Ibrahim *et al.*'s claim that their protocols do not have the *unbinding problem* as discussed in Section 4.3. Following the example given in [134] and without going into the technical details, the attack works as follows.

C requests 1000 different contents from D. Instead of generating distinct watermarks for each of the 1000 contents, C gives one identical watermark for D to embed into all 1000 contents. In other words, all 1000 marked contents that are passed to C have an identical watermark. C then generates many copies of these 1000 marked contents and distributes these copies illegally. At a later stage, D may find any of these illegal copies. However, as stated in [134], *D is only able to prove that one of these copies has been distributed illegally by C*. This means D will not be able to identify the particular content or prove the number of contents that have been copied.

We note that Williams *et al.* also pointed out the attack where C generates two different watermarks W_2 and W_3. These watermarks are used to avoid the claim that C bought the content, where copies of content are distributed illegally (Step $5'$ and Step $7'$ in our **Attack 2** of the protocols).

4.3.5 Deng-Preneel Analysis of the Protocols

Deng and Preneel [32] also pointed out a flaw in the IEH protocols that may occur if the underlying homomorphic encryption scheme is probabilistic. In general, in a probabilistic homomorphic encryption scheme, if a message is encrypted i times, the resulting i encrypted messages are distinct with high probability. The Paillier homomorphic encryption scheme (Figure 2.5) is one such scheme. From the `Content Watermarking and Distribution` phase of IEH-1 (Section 4.3.1):

- In Step 9, the CA decrypts $[[W]_{SIG_{ssk_C}}]_{HE_{hek_{CA}}}$ given by D. The signature $[W]_{SIG_{ssk_C}}$ obtained from the decryption is then verified to obtain the watermark W. After that, *the CA re-encrypts W with C's encryption key as* $[W]_{HE'_{hek_C}}$. It is signed to obtain $[[W]_{HE'_{hek_C}}]_{SIG_{ssk_{CA}}}$. This signature is sent to D.
- In Step 10, D verifies the signature $[[W]_{HE'_{hek_C}}]_{SIG_{ssk_{CA}}}$. After that, *$D$ generates the hash value $H([W]_{HE_{hek_C}})$ and the hash value $H([W]_{HE'_{hek_C}})$. Next D continues to execute the protocol if and only if the two hash values are identical.*

If a probabilistic homomorphic encryption scheme such as Paillier (Figure 2.5) is used, then the resulting encryption in Step 9 by CA, $[W]_{HE'_{hek_C}}$, will most probably

be different from $[W]_{HE_{hek_C}}$ that is given to D by C. The hash values of these encrypted messages are most probably different and D will thus halt the execution of the protocol. To avoid this flaw, the protocols need to use deterministic homomorphic encryption schemes, such as RSA (Figure 2.3). However, as stated in [90, Section 2.2], using a deterministic homomorphic encryption scheme for watermark embedding in the encrypted domain is not secure. Deng and Preneel also independently pointed out **Attack 1** presented in Section 4.3.3.

4.4 A Semi-Fair Content Tracing Protocol

As we have previously discussed, the PS protocol requires C to prove that the watermark is well-formed. But in doing so, additional computations are required. The IEH Protocols eliminate such a requirement by letting C generate any text string to represent the watermark. However, the IEH protocols can be exploited by C and are insecure, as we have demonstrated in the previous section.

In this section we suggest a possible way of allowing C to freely generate a watermark without facing the same issues as the IEH protocols, *if we assume that D is trusted more than C.* In our proposal, C will want to generate a well-formed watermark. If C generates an ill-formed watermark that can easily be removed from a marked content, and A cannot detect this watermark from the found copy of content, then A still assumes that C is guilty based on other evidence (e.g. C's signatures) provided by D. In this sense we say that D is assumed to be *semi-trusted*. By this we mean that when a watermark is ill-formed, A trusts that D will not falsely accuse C of illegal content distribution by releasing a copy of content that can be linked to C, but without C's watermark. This is based on the assumption that D has no knowledge of whether a watermark is ill-formed or well-formed (i.e. the watermark is encrypted), and that D will not coax C into generating an ill-formed watermark.

Fundamentals. The protocol involves C, D, a CA and A. The CA and A are fully trusted. Contrary to the trust assumptions of the PS and IEH protocols, D is semi-trusted. This protocol provides traceability, framing resistance and non-repudiation of redistribution.

Environment. Similar to the PS and IEH protocols, this protocol assumes that

C and D have ample computing resources. We also assume that the execution of the protocol is carried out using a secure communication channel and with public key support. There is no special trusted third party and the main building blocks are digital watermarking schemes, homomorphic encryption schemes and digital signature schemes. Table 4.3 shows the design framework of the protocol. In the following we describe the three phases of the protocol.

Table 4.3: The Design Framework of the Semi-Fair Protocol

Fundamentals	
Parties Involved	C, D, CA, A
Trust Assumptions	CA and A are *fully trusted* D is *semi-trusted*
Security Properties	Traceability (**TR**), Framing resistance (**FR**), Non-repudiation of redistribution (**NR**)
Environment	
Comp. Resources	Assume D and C have ample resources
Sec. comm. Support	Required
Pub. Key Support	Required
TTPs	No special TTP
Building Blocks	Digital watermarking scheme, homomorphic encryption scheme and digital signature scheme

`Initial Setup.` This follows the `Initial Setup` phase of the PS protocol. The main purpose is for C and D to obtain authenticated information from a CA on their public keys. At this point we only show the protocol messages in Figure 4.11, without describing the protocol steps.

① `Initial Setup:`

$$
\begin{aligned}
C\&D \rightarrow \text{CA} &: \{\textit{Request authenticated keys}\}_{AKE} \\
\text{CA} \rightarrow C &: \left\{[hek_C, pvk_C, ID_C]_{SIG_{ssk_{CA}}}\right\}_{AKE} \\
\text{CA} \rightarrow D &: \left\{[hek_D, pvk_D, ID_D]_{SIG_{ssk_{CA}}}\right\}_{AKE}
\end{aligned}
$$

Before content distribution

Figure 4.11: Semi-Fair Protocol – Initial Setup

`Content Watermarking and Distribution.` In this phase C requests content and D embeds a watermark into content in the encrypted domain. The encrypted marked content is then sent to C. Figure 4.12 shows the protocol messages.

The protocol steps are described below:

(I) C requests content, generates a client watermark and approves a content agree-

② Content Watermarking and Distribution:		
$C \rightarrow D$	: $\{hek_C, pvk_C, AGR, [hek_C, pvk_C, ID_C]_{SIG_{ssk_{CA}}}$ $[W]_{HE_{hek_C}}, [[W]_{HE_{hek_C}}, AGR]_{SIG_{ssk_C}}\}_{AKE}$	**Content distribution**
$D \rightarrow C$	: $\left\{[X'']_{HE_{hek_C}}, [[X'']_{HE_{hek_C}}]_{SIG_{ssk_D}}\right\}_{AKE}$	

Figure 4.12: Semi-Fair Protocol – Content Watermarking and Distribution

ment with D.

1. C generates watermark W, encrypts it with a homomorphic encryption scheme using his encryption key hek_C as

$$[W]_{HE_{hek_C}}.$$

C then signs it together with a content agreement AGR using his signing key ssk_C as

$$[[W]_{HE_{hek_C}}, AGR]_{SIG_{ssk_C}}.$$

After that, C sends the encrypted watermark, the signature and C's public keys, together with the CA's signature on these keys, to D.

(II) D produces a marked copy of the requested content and sends it to C.

2. D verifies the signature and generates a watermark V. After that, D embeds V into content X, resulting in

$$X' \leftarrow [X, V]_{EMB_{wmk_V}}.$$

The watermark V is to enable D to trace and identify the owner of found illegal copies of content. D encrypts every element of X' one-by-one with a homomorphic encryption scheme, using C's public encryption key hek_C. Similar to the MW protocol described in Section 3.7, D also permutes every encrypted element of W based on a permutation q. For example, given

$$[W]_{HE_{hek_C}} = ([w_1]_{HE_{hek_C}}, [w_2]_{HE_{hek_C}}, \ldots, [w_n]_{HE_{hek_C}}),$$

D permutes the elements as:

$$q\left([W]_{HE_{hek_C}}\right) = ([w_{q(1)}]_{HE_{hek_C}}, [w_{q(2)}]_{HE_{hek_C}}, \ldots, [w_{q(n)}]_{HE_{hek_C}}).$$

Next, D generates an encrypted marked content as follows:

$$\left.\begin{array}{ll} & [x'_i]_{HE_{hek_C}} \cdot [q(w_i)]_{HE_{hek_C}} \\ = & [x'_i \circ q(w_i)]_{HE_{hek_C}} \\ = & [x''_i]_{HE_{hek_C}} \end{array}\right\} \; 1 \leq i \leq n,$$

where $\circ$ represents either modular addition, modular multiplication or bit-wise XOR depending on the underlying homomorphic encryption used. We denote $[X'']_{HE_{hek_C}} = ([x''_1]_{HE_{hek_C}}, [x''_2]_{HE_{hek_C}}, \ldots, [x''_n]_{HE_{hek_C}})$.

D signs $[X'']_{HE_{hek_C}}$ as:

$$[[X'']_{HE_{hek_C}}]_{SIG_{ssk_D}},$$

and sends $[X'']_{HE_{hek_C}}$, $[[X'']_{HE_{hek_C}}]_{SIG_{ssk_D}}$ to C.

3. C verifies $[[X'']_{HE_{hek_C}}]_{SIG_{ssk_D}}$ and decrypts $[X'']_{HE_{hek_C}}$.

`Identification and Dispute Resolution.` In this phase, D identifies C from a found copy of content and proves to A that C has illegally distributed a copy of the content. Figure 4.13 illustrates the protocol messages and the following describes the protocol steps:

③ Identification and Dispute Resolution:		
D	: $\{\texttt{true}, \texttt{false}\} \leftarrow [\widehat{X}, V, X]_{DET_{wmk}}$	
$D \rightarrow A$	: $\left\{\widehat{X}, X', q, [W]_{HE_{hek_C}}, AGR, [[W]_{HE_{hek_C}}, AGR]_{SIG_{ssk_C}}\right\}_{AKE}$	
$A \rightarrow C$	: $\{\textit{decryption key?}\}_{AKE}$	**After content distribution**
$C \rightarrow A$	: $\{hdk_C\}_{AKE}$	
A	: $\{\texttt{true}, \texttt{false}\} \leftarrow [\widehat{X}, q(W), X']_{DET_{wmk}}$	
	If `false`, check $[[W]_{HE_{hek_C}}, AGR]_{SIG_{ssk_C}}$	

Figure 4.13: Semi-Fair Protocol – Identification and Dispute Resolution

(I) D detects a watermark from the found copy of content in order to identify the client that owns the content.

1. Upon finding an illegal copy $\widehat{X}$, D identifies the client by detecting the presence of V in $\widehat{X}$:

$$\{\texttt{true}, \texttt{false}\} \leftarrow [\widehat{X}, V, X]_{DET_{wmk}}.$$

If the detection algorithm returns `true`, then detection of V is successful.

(II) D proves to A that C illegally distributed copies of content.

2. D provides A with AGR, the signature $[[W]_{HE_{hek_C}}, AGR]_{SIG_{ssk_C}}$, the found copy $\widehat{X}$, the marked copy X', the permutation q and the encrypted watermark $[W]_{HE_{hek_C}}$.

3. A asks C for C's private key hdk_C. A then decrypts $[W]_{HE_{hek_C}}$ and detects whether W is present in $\widehat{X}$. If W is detectable, then C is said to have illegally distributed content $\widehat{X}$.

4. C may generate an ill-formed watermark and remove the watermark from the received marked content. In such a case, A will fail to detect W from $\widehat{X}$, which is the main reason for the PS protocol using a zero-knowledge proof system. It is also one of the reasons that the IEH Protocols are susceptible to **Attack 1** as described in Section 4.3.3. In order to avoid this issue, we place more trust on the distributor in the following sense:

   If A fails to detect W from the illegal copy $\widehat{X}$, given that the client's signature $[[W]_{HE_{hek_C}}, AGR]_{SIG_{ssk_C}}$ is valid, A checks if W is an ill-formed watermark (with the assumption that a list of ill-formed watermarks is defined). If it is, then the client is still guilty of illegal distribution. This is based on the argument that C is assumed to have removed the weak watermark W before distributing copies of it.

In summary, what we suggest is:

- *Stronger assumption.* The protocol is semi-fair. Whenever there is a dispute, D is trusted more than C.

- *Client signs the encrypted watermark.* The signature on the encrypted watermark by C at the beginning of the `Content Watermarking and Distribution` phase plays a key role for D and A to ascertain whether W is indeed generated by C.

- Most importantly, in the `identification and dispute resolution` phase, *if the client watermark W cannot be detected, A proceeds to check whether W is an ill-formed watermark.* This forces C to want to generate a well-formed watermark.

4.5 Analysis

In this section we analyse the security and efficiency of the protocols that we discussed.

4.5.1 Security

In this section we provide a brief security analysis on the protocols that we have described. We begin by analysing the framework and secure communication issues in the IEH protocols.

Design Framework. One of the main reasons that the IEH protocols discussed in Section 4.3 fail to fulfill the claimed security requirements is their lack of a proper framework. This particularly affects the role and trust assumptions on the third party and the definition of the security requirements. We examine them in the following:

- In their protocols, Ibrahim *et al.* state that the CA is fully trusted. This means that the CA will not conspire with D. However, we put forward **Attack 3** (Section 4.3.3), where the CA can reveal the watermark to D, for two reasons:
  - The first is that the CA in the IEH protocols processes the watermark, which means that the CA has access to the watermark and can store the watermark. The CA thus plays a similar role to a WCA, except that the CA does not generate the watermark. Hence Ibrahim *et al.* combined two parties, the CA and WCA, into one entity and denoted them as the CA.
  - Following from the first reason, the claims by Ibrahim *et al.* [65, 66] that protocols that deploy the CA and WCA as separate entities, such as the Memon-Wong protocol discussed in Section 3.7, face conspiracy problems due to the possibility of WCA conspires with D is rather controversial. This is because Ibrahim *et al.* assume that their CA with the role of the WCA is fully trusted but assume otherwise for the WCA in other protocols.

Ultimately, this brings us to conclude that if we properly define the roles and trust assumptions on the trusted third parties (i.e. the CA and WCA), the *conspiracy problem* of Section 4.3 may not be an issue at all.

- As for the additional "problems" (Section 4.3), stated by Ibrahim *et al.*, it can be observed that these are rather disparate in nature. If we examine them more carefully, we see that the *conspiracy problem* and the *unbinding problem* are attacks on the main property known as framing resistance, defined in Section 3.3.3. Similarly, the *man in the middle attack* relates to an attack (rather than property) on the communication channel. However the *client's participation in the dispute resolution problem* and the *practice applicability problem* actually reflect the application considerations of a FaCT protocol. This confusingly merges the security requirements required by such a protocol with the practical considerations and operations in an instantiation of it. Ambiguity in this leads to attacks of the type that we have described.

We further note that other protocols that follow approach II, such as the protocols proposed in [54, 139], also discuss the conspiracy problems. Again, similar to our discussion above, the claims made concerning these protocols are controversial due to the lack of proper definitions and trust assumptions on the third parties deployed.

In summary, the lessons here are to *make explicit the trust assumptions on the TTPs* and to *differentiate between requirements and other "problems"*. This has been provided in our design framework in Section 3.2.

Secure Communication. In **Attack 2** on the IEH protocols we see that C can intercept and replace messages transmitted between D and the CA. This happens mainly because the protocols fail to safeguard the communication between D and the CA, although much care has been taken to ensure secure transmission between C and D. To avoid this pitfall, we think a standard and safer approach should be followed. This is to secure the communications between all involved parties based on well-established protocols in the literatures [8, 16, 73], or based on a standard protocol such as SSL/TLS [36]. This is the *secure communication support* of Section 3.4.2. We can then construct the main part of a FaCT protocol given this support. In summary, a *secure communication channel should be provided using well-established methods.*

Traceability. This security property is assured for the three discussed protocols since D can trace content to the identity of a client based on *seq_no* in the PS protocol. Similarly, D can trace content to the identity of a client based on the watermark V embedded into content in the IEH protocols and the Semi-Fair protocol.

Framing Resistance. The PS protocol and the Semi-Fair protocol provide this property through the embedding of watermark W into content, without D knowing what the watermark is and the final marked copy given to C. Since D cannot determine W and does not know the final marked copy, the issue of framing does not exist.

Similarly, D cannot accuse C of redistributing a different content instead of the found copy by transferring the watermark from the found copy to this other content. This is because a signature ($[[W]_{COM_{hek_C}}, AGR]_{SIG_{ssk_C}}$ or $[[W]_{HE_{hek_C}}, AGR]_{SIG_{ssk_C}}$) that binds the watermark and a content agreement, is produced by C. This content agreement specifies the exact content required by C. In other words, D will need to produce a new signature that binds the extracted watermark to the higher value content for the accusation to be successful. Assuming that the underlying signature scheme is secure, such operation by D is computationally infeasible. The IEH protocols do not provide this property since they are susceptible to the three attacks that we demonstrated in Section 4.3.3.

Non-repudiation of Redistribution. The PS protocol and the Semi-Fair protocol provide this property through the signatures produced by C. These signatures are $[[W]_{COM_{hek_C}}, AGR]_{SIG_{ssk_C}}$ and $[[W]_{HE_{hek_C}}, AGR]_{SIG_{ssk_C}}$, and they allow A to verify that C owns the found copy by checking the agreement AGR. C cannot deny distributing the found copy when A detected the watermark W. The IEH protocols do not provide this property since **Attack 2** presented in Section 4.3.3 is possible.

Summary. It is tempting to design protocols without a special trusted third party using zero-knowledge proof systems, so that a protocol can be more efficient and involve only straightforward communication between C and D. This has been done in the IEH protocols and other protocols proposed in [33, 34, 54, 139]. However, from **Attack 1** presented in Section 4.3.3, we observe that it is possible for a malicious C to defeat these protocols since C can freely choose any text string as the watermark. In this case, framing resistance and non-repudiation of redistribution cannot be assured if the client watermark W can be exploited by the client. As we have

presented in the Semi-Fair protocol in Section 4.4, one alternative is to assume more trust on the distributor. However with such an assumption, the protocol cannot be considered as fair to the client. In summary, *unless new and simpler mechanisms are found, designing protocols that are secure where the watermark is generated by the client is hard.* A potential new and simpler mechanism will be a secure and more efficient zero-knowledge proof system (as attempted by Kuribayashi and Tanaka, discussed in Section 4.2.1).

We summarise the security analysis of the three discussed protocols in Table 5.4. In the table, **TR** denotes traceability, **FR** denotes framing resistance and **NR** denotes non-repudiation of redistribution.

Table 4.4: Summary of the Security Analysis

Protocols	TR	FR	NR	Conditions
PS	✓	✓	✓	Homomorphic bit commitment scheme.
IEH	✓	×	×	Deterministic homomorphic encryption. Susceptible to attacks **I**, **II**, **III** (Section 4.3.3) and William-Treharne-Ho attack.
SF	✓	✓	✓	Stronger assumption: D semi-trusted.

4.5.2 Efficiency

In this section we compare the efficiency of the discussed protocols. A summary is shown in Table 4.5.

Bandwidth. Assuming the content has n elements, all three protocols require $n|m|$ bits to transmit the encrypted marked content from D to C. This is because the size of each encrypted element of content is $|m|$, following the modulus m of the underlying homomorphic encryption scheme. The PS protocol further requires y extra protocol messages being transmitted between C and D. This is assuming that C and D run y rounds of the zero-knowledge proof process so that D is convinced that the watermark generated by C is well-formed.

Trusted Third Parties. Since all protocols do not require a special trusted third party for producing a client watermark, there is no extra communication overhead, compared to the MW protocol discussed in Section 3.7 and other categories of protocols that we will discuss in Chapters 5 and 6.

Computation. For the PS protocol, C needs to perform ny modular exponentiations ($ny\mathbf{E}$). This is assuming that C executes y rounds of protocol messages between D and C, and in each round computes n commitments to prove in zero-knowledge that the watermark is well-formed. C also performs n modular exponentiations ($n\mathbf{E}$) to open the committed marked content. This is assuming that each commitment, or opening the commitment, of the underlying homomorphic bit commitment scheme requires one modular exponentiation. D needs to perform n modular exponentiations ($n\mathbf{E}$) to commit to the content and n modular multiplication ($\mathbf{M}$) to multiply each committed element of content by each committed element of the watermark, so that the watermark is embedded into content in the encrypted domain. Altogether, C performs $n(y+1)\mathbf{E}$ computations and D performs $n(\mathbf{E}+\mathbf{M})$ computations.

The IEH protocols and the Semi-Fair protocol have similar computation requirements for C and D. In this case, D needs to compute n modular exponentiations ($n\mathbf{E}$) and n modular multiplications ($n\mathbf{M}$) to encrypt content and embed the watermark W into content. D also computes n additions ($n\mathbf{A}$) to embed the watermark V into content. C, on the other hand, needs to compute n modular exponentiations ($n\mathbf{E}$) to encrypt the watermark and another n modular exponentiations to decrypt the marked content. So D requires $n(\mathbf{E}+\mathbf{M}+\mathbf{A})$ computations, while C requires $2n\mathbf{E}$ computations.

Storage. For the purpose of producing the encrypted marked content, all protocols require D to store the homomorphic encryption (or commitment) public key of C. The key size is $|m|$. D also requires the encrypted (or committed) watermark with size $n|m|$ bits and the watermark V (or *seq_no*), for which we assume both have $n|Z|$ bits. For C, the required storage is $2|m|$ bits to store the homomorphic encryption and decryption keys. C also stores the watermark W with size $n|Z|$. So D stores $n|m| + |m| + n|Z|$ bits, while C stores $2|m| + n|Z|$ bits.

Summary. All three protocols have similar bandwidth, computation and storage requirements, except that the PS protocol requires more computation due to the zero-knowledge proof of knowledge between C and D.

Table 4.5: Efficiency Comparisons between Protocols without Trusted Third Parties

Pro.	*Bandwidth*	*TTP*	*Computation*[1]	*Storage*[2]
PS	$[X'']_{COM_{hek_C}} = n\|m\|$ y extra pro. msg.	No TTP	C: $n(y+1)\mathbf{E}$ D: $n(\mathbf{E}+\mathbf{M})$	C: $2\|m\| + n\|Z\|$ D: $(n+1)\|m\| + n\|Z\|$
IEH	$[X'']_{HE_{hek_C}} = n\|m\|$	No TTP	C: $2n\mathbf{E}$ D: $n(\mathbf{E}+\mathbf{M}+\mathbf{A})$	C: $2\|m\| + n\|Z\|$ D: $(n+1)\|m\| + n\|Z\|$
SF	$[X'']_{HE_{hek_C}} = n\|m\|$	No TTP	C: $2n\mathbf{E}$ D: $n(\mathbf{E}+\mathbf{M}+\mathbf{A})$	C: $2\|m\| + n\|Z\|$ D: $(n+1)\|m\| + n\|Z\|$

[1] $\mathbf{E}=O(k^3)$, $\mathbf{M}=O(k^2)$, $\mathbf{A}=O(k)$
[2] $|Z| < |m|$

4.6 Summary

In this chapter we have examined FaCT protocols without special trusted third parties. The main characteristic of these protocols is that the client generates the watermark. We discuss two approaches used in existing protocols. The first approach is based on homomorphic bit commitment schemes and zero-knowledge proofs of knowledge. The client in these protocols proves to the distributor that the watermark is well-formed. Due to this, they are relatively computationally expensive. This approach is represented by the PS protocol, which we discussed.

The second approach is based on homomorphic encryption schemes. The client in these protocols can generate any watermark and there is no need to convince the distributor that the watermark is well-formed. Without needing the zero-knowledge proof, these protocols are relatively more efficient. This approach is discussed by examining the recently proposed IEH protocols. We demonstrated flaws, which are caused by the client being able to exploit the watermark generation process, in the IEH protocols. This work was published in [111].

Due to these flaws, which also apply to other protocols in the same category, we argued that it is difficult to design a protocol without trusted third parties that is efficient. We further suggest a new Semi-Fair protocol that is efficient and does not face the issues of the second approach, but requires a stronger assumption in that the distributor is trusted more than the client.

Chapter 5

FaCT Protocols with Online Trusted Third Parties

Contents

This chapter examines FaCT protocols that require special online trusted third parties. We discuss and compare conventional protocols that deploy asymmetric homomorphic encryption schemes and newer protocols that use more efficient approaches. We further show that one recently proposed protocol is flawed due to lack of a proper definition of security properties.

5.1 Overview

FaCT protocols that require online trusted third parties are protocols that deploy a special trusted third party, such as a WCA, to generate client watermarks. The

trusted third party is online, meaning that it is always available during content distribution. There are two approaches to designing such protocols.

The conventional approach uses homomorphic encryption schemes such as Paillier (Figure 2.5) for embedding the watermark in the encrypted domain. We will describe the Lei-Yu-Tsai-Chan protocol [85] in Section 5.2 as an example.

The other approaches seek to replace homomorphic encryption schemes with more efficient alternatives. We describe the protocol proposed by Wu and Pang [137] (Section 5.3) as an example. We then describe a protocol proposed by Ahmed *et al.* [3], and show that the protocol is flawed (Section 5.4). This work was published in [112].

5.2 The Lei-Yu-Tsai-Chan Protocol

Lei, Yu, Tsai and Chan (LYTC) proposed a protocol [85] that is based on the MW protocol described in Section 3.7. It is interesting because it is the first protocol that highlighted the unbinding attack (see Section 4.3), showed that the attack can be mounted successfully on the MW protocol and proposed a solution to prevent this attack. The solution is to bind the client watermark onto the specific content requested by the client based on an agreement that contains a description of the content. It is also one of the earliest protocols that uses an online trusted third party and provides anonymity and unlinkability.

Fundamentals. The LYTC protocol involves five parties. These are C, D, a CA, a WCA and A. The CA, WCA and A are fully trusted. The protocol provides the three standard properties and additionally provides anonymity and unlinkability.

Environment. Although not mentioned in [85], the protocol assumes that C and D have ample computing resources. This is because of the large number of homomorphic encryptions (and decryptions) required to be performed by both C and D. The protocol also assumes public key and secure communication support. It deploys a WCA to generate the client watermarks and the WCA is always online during content distribution. The main building blocks are digital watermarking schemes, homomorphic encryption schemes and digital signature schemes. Table 5.1 shows the design framework of the LYTC protocol.

Table 5.1: The Design Framework of the LYTC Protocol

Fundamentals	
Parties Involved	C, D, CA, WCA, A
Trust Assumptions	CA, WCA, A are *fully trusted*
Security Properties	Traceability (**TR**), Framing resistance (**FR**), Non-repudiation of redistribution (**NR**), Anonymity and unlinkability (**AU**)
Environment	
Comp. Resources	Implicitly assumed D and C have ample resources
Sec. comm. Support	Required
Pub. Key Support	Required
TTPs	online TTP (WCA)
Building Blocks	Digital watermarking scheme, homomorphic encryption scheme and digital signature scheme

In the following we describe the protocol.

`Initial Setup.` The main purpose of this phase is for C and D to obtain certified public keys. Since the protocol also provides anonymity and unlinkability, certified anonymous keys are also provided by the CA to C. Hence there are two parts to the setup. The first part is for the CA to produce certified keys, and the second part is for the CA to produce certified anonymous keys. We note that the second part can be ignored if anonymity and unlinkability are not required. Following the general construction for protocols with anonymity and unlinkability discussed in Section 3.5.5, we describe the steps below. Figure 5.1 shows the protocol messages transmitted between the parties involved.

① `Initial Setup:`

$C\&D \rightarrow \text{CA}$: $\{\textit{Request authenticated keys}\}_{AKE}$ **Before content distribution**

$\text{CA} \rightarrow C$: $\left\{[hek_C, pvk_C, ID_C]_{SIG_{ssk_{CA}}}\right\}_{AKE}$

$\text{CA} \rightarrow D$: $\left\{[hek_D, pvk_D, ID_D]_{SIG_{ssk_{CA}}}\right\}_{AKE}$

$C \rightarrow \text{CA}$: $\left\{pvk_C^*, hek_C^*, [pvk_C^*, hek_C^*]_{SIG_{ssk_C}}, [hek_C, pvk_C, ID_C]_{SIG_{ssk_{CA}}}\right\}_{AKE}$

$\text{CA} \rightarrow C$: $\{Cert_{ssk_{CA}}(pvk_C^*, hek_C^*)\}_{AKE}$

Figure 5.1: LYTC Protocol – Initial Setup

(I) C and D register with the CA to obtain authenticated public keys.

1. C generates a signing key pair (pvk_C, ssk_C) and an encryption key pair (hek_C, hdk_C). Next C sends the public verification key pvk_C and the pub-

lic encryption key hek_C to the CA. The same process is followed by D.

(II) The CA generates and provides C and D with the authenticated key materials.

2. The CA, upon verifying the identity information provided by C and D, provides both parties with certified public keys. This means that signatures are produced on these keys, which can later be used by C and D to prove the validity of their respective public keys. This is shown in Figure 5.1.

(III) Anonymous certification of a new randomly generated key pair by the CA.

3. To create an anonymous certificate, C randomly generates a signature key pair (pvk_C^*, ssk_C^*). C also generates an encryption key pair (hek_C^*, hdk_C^*) and sends pvk_C^*, hek_C^*, a signature $[pvk_C^*, hek_C^*]_{SIG_{ssk_C}}$ and the CA's signature on the permanent public keys to the CA. The CA verifies the signatures and generates a signature $[pvk_C^*, hek_C^*]_{SIG_{ssk_{CA}}}$ as the anonymous certificate. This anonymous certificate is denoted by $Cert_{ssk_{CA}}(pvk_C^*, hek_C^*)$.

(IV) The CA sends the signed keys to C.

4. The anonymous certificate $Cert_{ssk_{CA}}(pvk_C^*, hek_C^*)$ is sent to C.

`Content Watermarking and Distribution.` This is the main phase in which D marks and encrypts content. The marked content is then sent to C. The protocol messages are shown in Figure 5.2 and the protocol steps are described in the following:

(I) C requests content and approves a content agreement with D.

1. C requests content from D by first negotiating with D a purchase agreement AGR. This AGR states the rights and licensing of the specific content.

2. C randomly generates one-time signature and encryption key pairs (pvk^*, ssk^*) and (hek^*, hdk^*). Next, an anonymous certificate is produced on the public

②	Content Watermarking and Distribution:	
$C \rightarrow D$	: $\{pvk^*, hek^*, Cert_{ssk_{CA}}(pvk_C^*, hek_C^*),$ $Cert_{ssk_C^*}(pvk^*, hek^*), AGR, [AGR]_{SIG_{ssk^*}}\}_{AKE}$	**Content distribution**
$D \rightarrow$ WCA	: $\{C$'s message, $X'\}_{AKE}$	
WCA $\rightarrow D$	: $\{[W]_{HE_{hek^*}}, [W]_{HE_{hek_{WCA}}},$ $[[W]_{HE_{hek^*}}, pvk^*, hek^*, [AGR]_{SIG_{ssk^*}}]_{SIG_{ssk_{WCA}}}\}_{AKE}$	
$D \rightarrow C$	: $\{[X'']_{HE_{hek^*}}\}_{AKE}$	

Figure 5.2: LYTC Protocol – Content Watermarking and Distribution

keys, resulting in a certificate $Cert_{ssk_C^*}(pvk^*, hek^*)$. C signs the agreement AGR, resulting in $[AGR]_{SIG_{ssk^*}}$, using the newly generated signing key ssk^* and sends to D pvk^*,hek^*, $Cert_{ssk_{CA}}(pvk_C^*, hek_C^*)$, $Cert_{ssk_C^*}(pvk^*, hek^*)$, AGR and $[AGR]_{SIG_{ssk^*}}$.

(II) D requests a client watermark from the WCA.

3. Upon receiving the message from C, D first checks that the certificate is valid and also verifies the signature. D aborts the protocol if any of them are invalid. If not, D generates a unique watermark V specific to this transaction. D embeds V into content X using any digital watermarking scheme preferred by D:

$$X' \leftarrow [X, V]_{EMB_{wmk_V}}.$$

This watermark V is required so that when a copy of content is found, D can detect V and, based on V, D can identify the client. After that, D sends pvk^*, hek^*, $Cert_{ssk_{CA}}(pvk_C^*, hek_C^*)$, $Cert_{ssk_C^*}(pvk^*, hek^*)$, AGR, $[AGR]_{SIG_{ssk^*}}$ and X' to the WCA.

(III) The WCA sends an encrypted client watermark to D.

4. The WCA verifies the anonymous certificate and the signature. If these are valid then the WCA generates a client watermark W. Next, the WCA encrypts the watermark using hek^* and hek_{WCA}, based on a homomorphic encryption scheme, resulting in $[W]_{HE_{hek^*}}$ and $[W]_{HE_{hek_{WCA}}}$. The WCA also generates a signature $[[W]_{HE_{hek^*}}, pvk^*, hek^*, [AGR]_{SIG_{ssk^*}}]_{SIG_{ssk_{WCA}}}$. The two encrypted watermarks and the signature are sent to D.

(IV) D produces a marked copy of the requested content and sends it to C.

5. Upon receiving the watermarks and signature from the WCA, D verifies the signature and embeds watermark W into content element-by-element as follows:

$$\left.\begin{array}{ll} & [x_i']_{HE_{hek^*}} \cdot [w_i]_{HE_{hek^*}} \\ = & [x_i' \circ w_i]_{HE_{hek^*}} \\ = & [x_i'']_{HE_{hek^*}} \end{array}\right\} \; 1 \leq i \leq n,$$

where n is the number of elements in the watermark and content, and $\circ$ represents either modular addition, modular multiplication or bit-wise XOR depending on the underlying homomorphic encryption used. We denote:

$$[X'']_{HE_{hek^*}} = ([x_1'']_{HE_{hek^*}}, [x_2'']_{HE_{hek^*}}, \ldots, [x_n'']_{HE_{hek^*}}).$$

D then stores in his database: the encrypted watermarks $[W]_{HE_{hek^*}}$ and $[W]_{HE_{hek_{WCA}}}$, the signatures $[[W]_{HE_{hek^*}}, pvk^*, hek^*, [AGR]_{SIG_{ssk^*}}]_{SIG_{ssk_{WCA}}}$ and $[AGR]_{SIG_{ssk^*}}$, the watermark V, the certificates $Cert_{ssk_{CA}}(pvk_C^*, hek_C^*)$ and $Cert_{ssk_C^*}(pvk^*, hek^*)$, and AGR. Next D sends $[X'']_{HE_{hek^*}}$ to C.

6. Finally, C decrypts $[X'']_{HE_{hek^*}}$ to obtain the marked content X''.

`Identification and Dispute Resolution.` When a suspicious copy of content is found, D initiates this phase to prove to A that C has illegally distributed a copy of the content. The protocol steps are described in the following (Figure 5.3):

③	`Identification and Dispute Resolution:`	
D	: $\{\texttt{true}, \texttt{false}\} \leftarrow [\widehat{X}, V, X]_{DET_{wmk}}$	
$D \rightarrow A$	: $\{[[W]_{HE_{hek^*}}, pvk^*, hek^*, [AGR]_{SIG_{ssk^*}}]_{SIG_{ssk_{WCA}}}$, $Cert_{ssk_{CA}}(pvk_C^*, hek_C^*), Cert_{ssk_C^*}(pvk^*, hek^*), [AGR]_{SIG_{ssk^*}}$, $AGR, [W]_{HE_{hek^*}}, [W]_{HE_{hek_{WCA}}}, X', \widehat{X}\}_{AKE}$	
$A \rightarrow$ WCA	: $\{watermark\ info?\}_{AKE}$	
WCA $\rightarrow A$	: $\{watermark\ info\}_{AKE}$	
A	: $\texttt{true} \leftarrow [\widehat{X}, W, X']_{DET_{wmk}}$	
$A \rightarrow$ CA	: $\{pvk_C^*, hek_C^*, Cert_{ssk_{CA}}(pvk_C^*, hek_C^*)\}_{AKE}$	**After content distribution**
CA $\rightarrow A$	: $\{ID_C\}_{AKE}$	

Figure 5.3: LYTC Protocol – Identification and Dispute Resolution

(I) D detects a watermark from the found copy of content in order to identify the client that owns the content.

1. When an illegal copy of content $\widehat{X}$ is found, D detects watermark V from this copy. If V is detected and can be matched to one of the clients in the database, then D is successful in identifying the perpetrator that illegally distributed this copy of content.

(II) D proves to A that C illegally distributed copies of content.

2. D sends $[AGR]_{SIG_{ssk^*}}$, $[[W]_{HE_{hek^*}}, pvk^*, hek^*, [AGR]_{SIG_{ssk^*}}]_{SIG_{ssk_{WCA}}}$, the certificates $Cert_{ssk_{CA}}(pvk^*_C, hek^*_C)$ and $Cert_{ssk^*_C}(pvk^*, hek^*)$, the agreement AGR, the encrypted watermarks $[W]_{HE_{hek^*}}$, $[W]_{HE_{hek_{WCA}}}$ and the marked content X', along with the found copy $\widehat{X}$, to A.

3. Upon receiving these messages, A verifies the certificates and signature. If these are valid, A requests the WCA to decrypt $[W]_{HE_{hek_{WCA}}}$. With W in hand, A performs a correctness check on $[W]_{HE_{hek^*}}$. This is done by encrypting the watermark W obtained from WCA and comparing the result with $[W]_{HE_{hek^*}}$. If both are identical then A proceeds to detect watermark W from the found copy $\widehat{X}$:

$$\texttt{true} \leftarrow [\widehat{X}, W, X']_{DET_{wmk}}.$$

If the detection returns `true`, then A asks the CA to reveal C using pvk^* and hek^*.

5.2.1 Deng-Preneel Analysis of the Protocol

The LYTC protocol suffers from the probabilistic encryption problem, similar to the IEH protocols discussed in Section 4.3.5. This is because in Step 3 of the `Identification and Dispute Resolution` phase, A re-encrypts watermark W obtained from WCA and compares the re-encrypted watermark with the one originally provided by D. If a probabilistic homomorphic encryption scheme such as Paillier (Figure 2.5) or Goldwasser-Micali (Figure 2.4) is deployed, then the re-encryption results in a different encrypted message with high probability. In such a case, the verification based on comparing the re-encrypted watermark with the original $[W]_{HE_{hek^*}}$ will fail.

In summary, the LYTC protocol is an improved and extended version of the MW protocol but requires the use of deterministic homomorphic encryption schemes such as RSA (Figure 2.3).

5.3 The Wu-Pang Protocol

Wu and Pang [137] proposed a FaCT protocol that only requires a digital watermarking scheme for D to produce the marked content, but at the same time the distributor D does not know the embedded watermark and the final marked copy given to C. As a result of this, the WP protocol is more efficient than the LYTC protocol discussed in the previous section, since no homomorphic encryption is involved.

Fundamentals. The WP protocol involves five parties. These are C, D, a CA, a WCA and A. The CA, WCA and A are fully trusted. The protocol provides traceability, framing resistance and non-repudiation of redistribution.

Environment. Since the protocol does not involve computational intensive operations based on homomorphic encryption, it has the flexibility for C to possess only limited resources. D still needs to have ample resources since there may be many clients contacting D to request content. While not mentioned directly, the WP protocol requires public key support and a secure communication channel. The main building blocks are the spread spectrum watermarking scheme [28] described in Figure 2.1, asymmetric encryption schemes and digital signature schemes. Table 5.2 illustrates the design framework of the WP protocol.

We note that the WP protocol is incomplete in the sense that it does not provide detailed protocol messages between the parties involved, except for a brief description of the communication with the WCA. This means, for example, that the requirement for signatures for data origin authentication and non-repudiation of redistribution are only mentioned and assumed in placed. Therefore, in our later description of the protocol, we briefly state the needs for the signatures or other information in the protocol messages between the parties involved.

The main idea of the protocol is to let the WCA generate two independent watermarks, W_1 and W_2, and add these two watermarks together. The resulting water-

Table 5.2: The Design Framework of the WP Protocol

Fundamentals	
Parties Involved	C, D, CA, WCA, A
Trust Assumptions	CA, WCA and A are *fully trusted*
Security Properties	Traceability (**TR**), Framing resistance (**FR**), Non-repudiation of redistribution (**NR**)
Environment	
Comp. Resources	D have ample resources Flexibility: C may have limited resources
Sec. comm. Support	Required
Pub. Key Support	Required
TTPs	online TTP (WCA)
Building Blocks	Digital watermarking scheme, asymmetric encryption scheme and digital signature scheme

mark W is embedded into content by D. This means that a marked content X' that consists of two watermarks is produced. One of the watermarks W_2 is set in such a way that it distorts the content into a copy with much less quality than the original. This lesser quality copy is given to C. To obtain content that is of similar quality to the original, C is given W_2. C subtracts W_2 from the lesser quality marked content X' to produce a copy of content that is marked only with W_1 and is of similar quality to the original. D does not know W_1 and the final marked copy since he only has $W = W_1 + W_2$. Similarly, C does not know W_1 since he is only given W_2. There are other measures to make sure that D and C cannot guess W_1 from the information that they have. In the following we describe the protocol in more detail.

`Initial Setup.` As this was not provided in the original description by Wu and Pang in [137], we assume this is similar to the `Initial Setup` phase in the LYTC protocol described in Section 5.2, except that anonymity and unlinkability are not considered. So D and C register with the CA to obtain the CA's signatures (or certificate) on their respective public keys (i.e. C's and D's signature verification and public encryption keys). We denote this by $[pek_C, pvk_C, ID_C]_{SIG_{ssk_{CA}}}$ and $[pek_D, pvk_D, ID_D]_{SIG_{ssk_{CA}}}$. This allows the parties involved to verify that signatures are authentic. Figure 5.4 shows the protocol messages.

`Content Watermarking and Distribution.` In Wu and Pang's original description in [137], there is no provision of public keys and signatures. We adopt the original description but provide these messages containing public keys and signatures to show completeness of the protocol. Figure 5.5 shows the protocol messages

① Initial Setup:		
$C\&D \rightarrow \text{CA}$	: $\{Request\ authenticated\ keys\}_{AKE}$	**Before content distribution**
$\text{CA} \rightarrow C$	: $\left\{[pek_C, pvk_C, ID_C]_{SIG_{ssk_{CA}}}\right\}_{AKE}$	
$\text{CA} \rightarrow D$	: $\left\{[pek_D, pvk_D, ID_D]_{SIG_{ssk_{CA}}}\right\}_{AKE}$	

Figure 5.4: WP Protocol – Initial Setup

and the protocol steps are described below:

(I) C requests content from D.

1. C sends to D a content request message. The message includes the CA's signature $[pek_C, pvk_C, ID_C]_{SIG_{ssk_{CA}}}$ and C's public verification key pvk_C.

(II) D requests a watermark from the WCA.

2. Upon receiving the content request, D sends C's request and description of the content to the WCA.

(III) The WCA sends a watermark to D.

3. The WCA generates two distinct watermarks W_1 and W_2 based on the features of content and the client information provided by D. After that, the WCA computes $W = W_1 + \beta W_2$, where $\beta > 1$ is a pre-determined parameter that controls the distortion level of the marked content. The WCA also sets a parameter ρ, where ρ is a real number, which is determined by the robustness and content quality requirements of the underlying spread spectrum watermarking scheme (Figure 2.1). As suggested in [137] based on their experiments, $\rho = 20$ and $\beta = 10$. They also mentioned that for the security of the protocol, ρ and β must be kept secret from C. This leads to a security issue, which we discuss in the summary of Section 5.5.1.

4. Then WCA computes $\rho\beta W_2$ and encrypts it with C's public encryption key pek_C as $[\rho\beta W_2]_{PE_{pek_C}}$, using an asymmetric encryption scheme such as RSA-OAEP [82]. The WCA sends $[\rho\beta W_2]_{PE_{pek_C}}$ to C.

5. The WCA also encrypts W with D's public encryption key pek_D as $[W]_{PE_{pek_D}}$ and sends this to D, together with the parameter ρ.

(IV) D produces a marked copy of the requested content and sends it to C.

6. Upon receiving $[W]_{PE_{pek_D}}$ from the WCA, D decrypts the message to retrieve W. Next D embeds W into content X using the parameter ρ. This generates a marked content as:

$$X' = X + \rho W = X + \rho W_1 + \rho\beta W_2.$$

The marked content X' and the description of the content are encrypted using C's public encryption key and sent to C.

7. When X' is received, C generates his marked copy by performing the following:

$$X' - \rho\beta W_2 = X + \rho W_1 + \rho\beta W_2 - \rho\beta W_2 = X + \rho W_1.$$

C also verifies that $X + \rho W_1$ matches the content description provided by D.

② `Content Watermarking and Distribution:` — **Content distribution**

$C \rightarrow D$	: $\{request\ content\}_{AKE}$
$D \rightarrow$ WCA	: $\{C\text{'s request, description of } X\}_{AKE}$
WCA $\rightarrow C$	: $\left\{[\rho\beta W_2]_{PE_{pek_C}}\right\}_{AKE}$
WCA $\rightarrow D$	: $\left\{\rho, [W = W_1 + \beta W_2]_{PE_{pek_D}}\right\}_{AKE}$
$D \rightarrow C$	: $\{X', description\ of\ X\}_{AKE}$

Figure 5.5: WP Protocol – Content Watermarking and Distribution

`Identification and Dispute Resolution.` The protocol steps are described below, and Figure 5.6 shows the protocol messages.

(I) D detects a watermark from the found copy of content in order to identify the client that owns the content.

1. When an illegal copy $\widehat{X} = X + \widehat{\rho W_1}$ is found, D subtracts the original content X from $\widehat{X}$. This results in a watermark $\widehat{\rho W_1}$. Then, based on $W = W_1 + \beta W_2$ and $\widehat{\rho W_1}$, by setting a proper measure, D is able to detect the presence of $\widehat{\rho W_1}$, which is similar to ρW_1. Wu and Pang provide details of the detection process in [137].

(II) D proves to A that C illegally distributed copies of content.

2. Upon successful detection, D sends $\widehat{X}$ to A. Next, A requests the WCA to provide the original watermark W_1 and the parameter ρ. Based on these, A detects W_1 from the illegal copy $\widehat{X}$. This detection is more accurate than the detection by distributor D since there is less noise involved given the exact watermark W_1 and the parameter ρ, instead of detection using W. If the detection returns true, then C is guilty.

③ Identification and Dispute Resolution:		
D	: $\{\texttt{true}, \texttt{false}\} \leftarrow [\widehat{X}, W, X]_{DET_{wmk}}$	
$D \rightarrow A$	: $\left\{\widehat{X}\right\}_{AKE}$	
$A \rightarrow$ WCA	: $\{\textit{Request watermark}\}_{AKE}$	**After content distribution**
WCA $\rightarrow A$	: $\{\rho, W_1\}_{AKE}$	
A	: $\texttt{true} \leftarrow [\widehat{X}, \rho W_1, \widehat{\rho W_1}]_{DET_{wmk}}$	

Figure 5.6: WP Protocol – Identification and Dispute Resolution

In summary, the WP protocol is a simple and efficient FaCT protocol that does not require specific building blocks such as a homomorphic encryption scheme. However, the security of the protocol relies on the two parameters ρ and β, which have small values as suggested in [137]. We will discuss this issue in the summary of Section 5.5.1.

5.4 The Ahmed-Sattar-Siyal-Yu Protocol

The Ahmed-Sattar-Siyal-Yu (ASSY) protocol was proposed in [3]. Similarly to the WP protocol, it is interesting because it does not use homomorphic encryption. However, we will show that there are design flaws in the ASSY protocol. This work was published in [112].

Fundamentals. The ASSY protocol involves five parties. These are C, D, a CA, a WCA and A. The CA, WCA and A are fully trusted. The protocol claimed to provide traceability and non-repudiation of redistribution.

Environment. Similarly to the WP protocol, we assume that D has ample resources, while C may only possess limited resources. The ASSY protocol requires

public key support and secure communication. The main building blocks are digital watermarking schemes and digital signature schemes. We remark that in the proposal two different types of watermarking schemes, based on the spread spectrum scheme described in Section 2.3.1, were proposed. We will not discuss them in detail as our focus is on the roles of the parties involved and how the exchange of protocol messages between the distributor and the client attempts to provide the requirements listed above. Table 5.3 summarises the design framework of the ASSY protocol.

As can be observed, the ASSY protocol does not provide framing resistance, which is the main reason that it is not secure. We show how the provisions of traceability and non-repudiation of redistribution are flawed, unless both the distributor and the client are fully trusted, which in this case defeats the very purpose of constructing the protocol, where the distributor and the client do not trust each other.

The ASSY protocol also claimed to provide *client-distributor identification* and *ownership verification*. The first property allows a client to reveal the identity of the distributor and his own identity from the received marked content so that the client can ascertain that the content is indeed meant for him. It is also used as one of the mechanisms to provide non-repudiation of redistribution. We will show how this is flawed. The second property allows the distributor to claim ownership of content when multiple ownership claims occur. This relates more to the issue of copyright ownership instead of fair content tracing and hence we will not discuss it.

We next describe the three phases of the ASSY protocol.

Table 5.3: The Design Framework of the ASSY Protocol

Fundamentals	
Parties Involved	C, D, CA, WCA, A
Trust Assumptions	CA, WCA, A are *fully trusted*
Security Properties	Traceability (**TR**), Non-repudiation of redistribution (**NR**)
Environment	
Comp. Resources	D have ample resources Flexibility: C may have limited resources
Sec. comm. Support	Required
Pub. Key Support	Required
TTPs	online TTP (WCA)
Building Blocks	Digital watermarking scheme, and digital signature scheme

`Initial Setup.` In this phase the parties involved, such as D and C, generate signature key pairs (pvk_D, ssk_D) and (pvk_C, ssk_C). The public verification keys are signed by a CA, together with C and D's identity to prove the authenticity of the keys of the parties involved. The public keys and the signatures can then be distributed to all parties involved in the protocols. This is similar to the `Initial Setup` phase of the LYTC protocol discussed in Section 5.2, except that it does not involve the generation of an anonymous certificate. Figure 5.7 shows the protocol messages.

① `Initial Setup:`		
$C\&D \rightarrow \text{CA}$:	$\{\textit{Request authenticated keys}\}_{AKE}$	**Before content distribution**
$\text{CA} \rightarrow C$:	$\left\{[pvk_C, ID_C]_{SIG_{ssk_{CA}}}\right\}_{AKE}$	
$\text{CA} \rightarrow D$:	$\left\{[pvk_D, ID_D]_{SIG_{ssk_{CA}}}\right\}_{AKE}$	

Figure 5.7: ASSY Protocol – Initial Setup

`Content Watermarking and Distribution.` Figure 5.8 illustrates the protocol messages. We describe the protocol steps as follows:

(I) C requests content from D.

1. C initiates the protocol by sending a purchase request message to D. This message contains C's public key and the CA's signature on this key.

(II) D requests a client watermark from the WCA.

2. D contacts the WCA to request a distributor's watermark V. The hash value of the original content $H(X)$ to the WCA by D. This hash value can be generated based on a cryptographic hash function such as SHA-2 [70] or RIPEMD-160 [37].

(III) The WCA sends an encrypted client watermark to D.

3. After verifying the request, the WCA generates V and a signature consisting of V, a hash value of X and a timestamp T as $[V, H(X), T]_{SIG_{ssk_{WCA}}}$. The watermark, the timestamp and the signature are sent to D.

(IV) D produces a marked copy of the requested content and sends it to C.

4. After receiving this message and verifying the signature, D embeds V into content X requested by C, resulting in the marked content X':

$$X' \leftarrow [X, V]_{EMB_{wmk_V}}.$$

5. D generates C's watermark as $W = H(pvk_D, pvk_C)$. In other words, the watermark W is the hash value of the public keys of D and C.

6. D embeds the watermark W into content. Due to the specific design of the underlying watermarking algorithm proposed by Ahmed *et al.* [3], a watermark public key KS is also generated during watermark embedding:

$$X'', KS \leftarrow [X', W]_{EMB_{wmk_V}}.$$

   The marked content X'', the key KS and a hash value $H(V)$ are sent to C, together with a signature $[X'', KS, H(V)]_{SIG_{ssk_D}}$ generated by D.

7. Using key KS, C extracts a watermark W from the marked content X'' and compares W with the hash value $H(pvk_D, pvk_C)$. If $W = H(pvk_D, pvk_C)$, then C can be sure that W reflects C's and D's identities.

8. C generates a signature $[H(X'', H(V), pvk_D)]_{SIG_{ssk_C}}$ to acknowledge receiving the marked content and sends this signature to D. This is different from the previous protocols.

9. Finally, D checks the signature $[H(X'', H(V), pvk_D)]_{SIG_{ssk_C}}$.

② `Content Watermarking and Distribution:`

$C \rightarrow D$	: $\{request\ content\}_{AKE}$	**Content distribution**
$D \rightarrow$ WCA	: $\{request\ watermark\ V\}_{AKE}$	
WCA $\rightarrow D$	: $\left\{V, T, [V, H(X), T]_{SIG_{ssk_{WCA}}}\right\}_{AKE}$	
$D \rightarrow C$	: $\left\{X'', KS, H(V), [X'', KS, H(V)]_{SIG_{ssk_D}}\right\}_{AKE}$	
$C \rightarrow D$	: $\left\{[H(X'', H(V), pvk_D)]_{SIG_{ssk_C}}\right\}_{AKE}$	

Figure 5.8: ASSY Protocol – Content Watermarking and Distribution

`Identification and Dispute Resolution.` When an illegal copy of content $\widehat{X}$ is found, D and A initiate the following, with Figure 5.9 showing the protocol messages:

(I) A (or D) detects a watermark from the found copy of content in order to identify the client that owns the content.

1. As was illustrated in Step 7, A can verify that C is the legitimate owner of the found content $\widehat{X}$ based on the extracted watermark W, by comparing the watermark W with the hash value $H(pvk_D, pvk_C)$. This hash value can be easily produced by A since it is based on the public keys of C and D. For the same reason, C is capable of removing W from the original marked content. So if W cannot be found from $\widehat{X}$, A and D move to the next step.

(II) D proves to A that C illegally distributed copies of content.

2. D supplies A with the original content X, the marked content X'', the found copy $\widehat{X}$, the watermark V and the client signature $[X'', H(V), pvk_D]_{SIG_{ssk_C}}$. A extracts V from $\widehat{X}$ based on X and retrieves $H(X'', H(V), pvk_D)$ from $[H(X'', H(V), pvk_D)]_{SIG_{ssk_C}}$. Next, A computes $H^*(X'', H(V), pvk_D)$ based on X'' and V supplied by D. If $H^*(X'', H(V), pvk_D) = H(X'', H(V), pvk_D)$ then A is sure that C bought content X'' from D. Finally, C is considered guilty when the extracted watermark V matches the original V supplied by D and the found content $\widehat{X}$ is similar to X''.

③ Identification and Dispute Resolution:		
D or A	: $\{\texttt{true}, \texttt{false}\} \leftarrow [\widehat{X}, W, KS]_{DET_{wmk}}$	
$D \rightarrow A$	: $\left\{X, X'', \widehat{X}, V, [H(X'', H(V), pvk_D)]_{SIG_{ssk_C}}\right\}_{AKE}$	**After content distribution**
A	: $\texttt{true} \leftarrow [\widehat{X}, V, X'']_{DET_{wmk}}$	

Figure 5.9: ASSY Protocol – Identification and Dispute Resolution

5.4.1 Flaws in ASSY Protocol

In this section we analyse the ASSY protocol against three attacks. The first attack is a common attack on FaCT protocols but has not been explicitly defined previously. The second attack is a new attack that we have defined based on the design weakness

of the ASSY protocol. The third attack is similar to the client-generate-watermark attack described in Section 4.3.3.

Attack 1: Framing and Buyer-denial Attacks. *D knows both the watermarks V and W and can launch a framing attack on C.* As can be observed from the `Content Watermarking and Distribution` phase, when an illegal copy $\widehat{X}$ is found, D identifies C by extracting watermark V and/or W from this found copy. After doing so, D searches the database for the acknowledgment signature $[H(X'', H(V), pvk_D)]_{SIG_{ssk_C}}$ to prove that C indeed bought the content that is similar to the illegal copy. We argue that C can be framed by an unscrupulous D since D knows both V and W and can embed these watermarks into any content. Even worse, D has the client's signature $[H(X'', H(V), pvk_D)]_{SIG_{ssk_C}}$ and can prove that C bought the content, when instead copies of it may have been illegally distributed by D. Conversely, C can launch a buyer-denial attack and claim that it is D who distributed illegal copies of content. This creates a deadlock scenario and can only be solved if we assume that either D or C is honest. Hence, as long as the above scenario of framing or denial exists, D cannot prove to a third party that C is guilty. As Ahmed *et al.* did not consider the possibility of a framing attack, we argue that proving C guilty of illegal content distribution based on the watermarks V, W and C's acknowledgment signature in their protocol is flawed.

Attack 2: Client-runaway Attack. C can choose not to send the acknowledgment signature $[H(X'', H(V), pvk_D)]_{SIG_{ssk_C}}$ to D, since C has already obtained the marked content. Without this signature, there is no way for D to prove to a third party that an illegal copy of bought content belongs to C. While D can always request C to send this signature, a dishonest C can either refuse (or pretend) that it has been sent. The only resolution in the above scenario is for D to blacklist this dishonest C from the client database.

Attack 3: Client-generate-watermark Attack. C can compute the watermark $W = H(pvk_D, pvk_C)$ and trivially subtract W from the marked content (which is also mentioned as possible in [3]). We stress that C can also replace the watermark, since W is the hash value of two public verification keys. In this case C can hash any other client's verification key together with the distributor's verification key, resulting in a new watermark $W^U = H(pvk_D, pvk_U)$, and can then embed this watermark into content, which totally dismisses the credibility of the identification

process in Step (I) in the `Identification and Dispute Resolution` phase.

We further note that in fact anyone authorised to check watermark W can launch a watermark-removal attack since (pvk_D, pvk_C) are in the public domain. So watermark W seems not to serve any purpose at all. Identification of the client still requires the distributor to detect watermark V from any content, rendering the process of embedding and detecting W redundant.

As can be observed from the three attacks presented, the ASSY protocol is not secure. The main problems are the fact that D has full access to the two watermarks and the final marked content, and the requirement that C sends to D an acknowledgment receipt.

5.5 Analysis

In the following we provide security analysis and performance comparisons of the protocols that we have discussed.

5.5.1 Security

In this section we analyse the security of the FaCT protocols we have discussed.

Traceability. In the LYTC protocol, traceability is assured through the embedding and detection of watermark V and W. D can trace C based on watermark V, and A can trace C based on watermark W, with the help of the CA and WCA.

In the WP protocol, traceability is assured through the detection of watermark W_1.

In the ASSY protocol, traceability is provided through the watermark V given to D by the WCA. The other watermark $W = H(pvk_D, pvk_C)$ cannot assure traceability since it can be generated by anyone using the two publicly available public verification keys. This was demonstrated in **Attack 3** in Section 5.4.1.

Framing Resistance. In the LYTC protocol, since the WCA generates watermark W and encrypts the watermark, it is not possible for D to frame C as D does not know the embedded watermark W and the final marked copy given to C. However,

as mentioned in Section 5.2.1, the protocol restricts itself to only deterministic homomorphic encryption schemes. As was pointed out in [90, Section 2.2], using such encryption schemes for watermarking content in the encrypted domain is not secure. This is mainly due to the size of the content space (e.g. images in 16 bits) being small, and a brute force attack may be carried out to guess the encrypted message.

The LYTC protocol also prevents D from extracting the watermark from the found copy and embedding this watermark into other more valuable content (the *unbinding attack*). This is because there is the signature,

$$[[W]_{HE_{hek^*}}, pvk^*, hek^*, [AGR]_{SIG_{ssk^*}}]_{SIG_{ssk_{WCA}}},$$

that explicitly binds the watermark with a particular content through AGR.

In the WP protocol, only the WCA knows the watermark $W = W_1 + \beta W_2$. D is in possession of the watermark $W = W_1 + \beta W_2$ but does not know the final marked copy $X' = X + \rho W_1$ possessed by C. So it is not possible for D to frame C. As for preventing D from extracting and then embedding the watermark into other content, although mentioned, this is not addressed. This is because D can extract $\widehat{\rho W_1}$ by subtracting the original content X from the found content $\widehat{X}$.

In the ASSY protocol, framing resistance is not considered. This resulted in the protocol being susceptible to **Attack 1** discussed in Section 5.4.1 and it thus fails to provide framing resistance.

Non-repudiation of Redistribution. In the LYTC protocol, non-repudiation of redistribution is provided since there is a signature generated by the WCA,

$$[[W]_{HE_{hek^*}}, pvk^*, hek^*, [AGR]_{SIG_{ssk^*}}]_{SIG_{ssk_{WCA}}},$$

that binds the agreement AGR with the watermark W. Furthermore, when the watermark W is detected, with the assistance of the WCA and CA, the identity of C that matches watermark W can be obtained.

In the WP protocol, non-repudiation of redistribution is provided through the successful detection of watermark W_1 by A. With the assistance of the CA, the identity information of C can be obtained.

In the ASSY protocol, non-repudiation of redistribution is not assured, although it is claimed to be provided. The reason is because of the successful framing and client

denial attacks that we demonstrated in **Attack 1** in Section 5.4.1.

Anonymity and Unlinkability. The LYTC protocol further provides this property. This can be observed from the `Initial Setup` phase, where anonymous certificate and one-time key pairs are used by C to communicate with D.

Summary. Assuming the security of the underlying building blocks, traceability, framing resistance and non-repudiation of redistribution are provided in the LYTC protocol and the WP protocol. However, for the LYTC protocol, there is the restriction of using deterministic homomorphic encryption schemes only.

We also note that for the WP protocol, Wu and Pang mentioned that the two predetermined parameters ρ and β are kept secret from C. If these values are revealed, then C knows W_2 from $\rho\beta W_2$ provided by D. Based on the knowledge of these three values, C can produce a new watermark, which, when subtracted from the marked content that contains W_1, causes watermark detection to fail. As suggested by Wu and Pang, $\rho = 20$ and $\beta = 10$. Based on their experiments, they showed that these two values give the optimal results for generating a distorted content and recovering a good quality content for C. However, they are small numbers and hence the security of the protocol can be considered weak, since C can guess (or exhaustively search for) these values.

As for the ASSY protocol, we have shown that due to the design flaws, it is not secure. We summarise the security analysis of these protocols in Table 5.4.

Table 5.4: Summary of the Security Analysis

Protocols	**TR**	**FR**	**NR**	**AU**	**Conditions**
LYTC	✓	✓	✓	✓	Deterministic homomorphic encryption.
ASSY	✓	×	×	×	Susceptible to attacks **I**, **II**, **III** (Section 5.4.1).
WP	✓	✓	✓	×	C must not know ρ and β. Susceptible to unbinding attack.

5.5.2 Efficiency

In this section we examine the performance of the protocols that we have discussed. Table 5.5 shows the efficiency comparison between these protocols.

We will (reasonably) assume that the LYTC protocol presented in Section 5.2 deploys the homomorphic encryption scheme of RSA [117] or Paillier [99], where both of the

schemes have similar computational performance, as stated in [99]. We also use the evaluation of the LYTC protocol to represent existing FaCT protocols with online trusted third parties that deploy homomorphic encryption schemes. We further note that the ASSY protocol contains flaws, as was shown in Section 5.4.1. Hence we will not include the ASSY protocol in our comparisons.

Bandwidth. For the LYTC protocol, the size of the encrypted content that is transmitted from D to C is $n|m|$ following the underlying homomorphic encryption scheme. This is because n elements of content are encrypted under the modulus m. As an example, if $|m| = 1024$ and $n = 10000$, then the size of the encrypted content will be $10000 \cdot 1024$ bits ≈ 1.3 MByte.

In the WP protocol, the final marked content size is $n|Z|$, where Z is the maximum value possible for each element of the marked content as discussed in Section 3.6. This is because the final marked content given to C is the same size as the original, which has n elements and each element has $|Z|$ bits.

Trusted Third parties. All protocols deploy an online trusted third party. This means that the trusted third party is involved in the `Content Watermarking and Distribution` phase, and must be always online so that the distributor can request client watermarks when required. Hence, the communication overhead is higher than the protocols without trusted third parties discussed in the previous chapter.

Computation. The most expensive computation in the LYTC protocol is modular exponentiation under the RSA or Paillier homomorphic encryption scheme for encrypting and decrypting the marked content. As was described in the `Content Watermarking and Distribution` phase in Section 5.2, for D, homomorphic encryption is repeated n times to encrypt each content element. So the computation required is $n\mathbf{E}$. In addition, D multiplies the encrypted elements of the watermark and the encrypted elements of content. This requires $n\mathbf{M}$ computations. Also, D computes n additions ($n\mathbf{A}$) to embed a watermark (i.e. V) into content. Hence the computation for D is $n(\mathbf{E} + \mathbf{M} + \mathbf{A})$. For C, decryption is repeated n times to decrypt the marked content. Therefore the computation is $n\mathbf{E}$ for C.

Compared to the LYTC protocol, the generation of the final marked content in the WP protocol is based only on the underlying watermarking scheme, which is the spread spectrum watermarking scheme described in Figure 2.1. This means that

the computation involves adding the watermark into content (**A**). As discussed in Step 6 of the `Content Watermarking and Distribution` phase in Section 5.3, D embeds one watermark into the content. Therefore the computation is $n\mathbf{A}$. For C the computation is $n\mathbf{A}$ as C needs to subtract a watermark from the marked content to obtain a good quality copy of content, as shown in Step 7 of the same phase.

Storage. In the LYTC protocol, for producing the encrypted marked content, D needs to store the homomorphic encryption key of C. This has size $|m|$ bits, following the size of the modulus m of the underlying homomorphic encryption scheme. D also stores the encrypted watermark obtained from the WCA, which has size $n|m|$ due to n encrypted elements of the watermark each having $|m|$ bits. D further stores the watermark V, which has size $n|Z|$, as V has n elements and each element can be $|Z|$ bits in size, as stated in Section 3.6. Hence, in total, D stores $(n+1)|m| + n|Z|$ bits. C, on the other hand, needs to store only the encryption and decryption keys. This means $2|m|$ bits of storage.

In the WP protocol, D embeds a watermark W into content to generate the final copy of content for C. The watermark W has the size of $n|Z|$. While for C, a watermark W_2 is required to produce a good quality copy of content from the final copy given by D. This was illustrated in Step 7 of the `Content Watermarking and Distribution` phase in Section 5.3. So C stores $n|Z|$ bits for W_2. However, C may choose to purge W_2 once the good quality copy is produced. In this case, C does not require any storage. We again note that this considers only the storage required to obtain the marked content.

Table 5.5: Efficiency Comparisons between Protocols with online Trusted Third Parties

Pro.	*Bandwidth*[1]	*TTP*	*Computation*[2]	*Storage*[1]
LYTC	$[X'']_{HE_{hek_C}} = n\|m\|$	online WCA	C: $n\mathbf{E}$ D: $n(\mathbf{E}+\mathbf{M}+\mathbf{A})$	C: $2\|m\|$ D: $(n+1)\|m\| + n\|Z\|$
WP	$X' = n\|Z\|$	online WCA	C: $n\mathbf{A}$ D: $n\mathbf{A}$	C: $n\|Z\|(0)$ D: $n\|Z\|$

[1] $|Z| < |m|$, $|n| < |L|$
[2] $\mathbf{E}$=$O(k^3)$, $\mathbf{M}$=$O(k^2)$, $\mathbf{A}$=$O(k)$, $\mathbf{S}$=$\mathbf{E}/100$

Summary. From Table 5.5 and the above analysis, we observe that the WP protocol, which uses only a watermarking scheme, is computationally more efficient than the conventional approach represented by the LYTC protocol.

5.6 Summary

In this chapter we investigated FaCT protocols with online trusted third parties. All protocols deploy a special online trusted third party known as a WCA to generate the client's watermark. These protocols use either asymmetric homomorphic encryption schemes such as the Paillier encryption scheme [99] as in the LYTC protocol [85], or use more computationally efficient designs as in the WP [137] and ASSY [3] protocols. We compared the security and efficiency of these protocols and exposed design flaws in the ASSY protocol, which were published in [112].

Chapter 6

FaCT Protocols with Offline Trusted Third Parties

Contents

This chapter examines FaCT protocols that require special offline trusted third parties. Our focus is on the approach where symmetric building blocks are used to design these protocols, instead of the conventional approach where asymmetric homomorphic schemes are deployed. We investigate one recent proposal and compare it with a new design approach that we propose. This new approach uses a symmetric building block known as Chameleon Encryption and is more efficient in terms of communication with the trusted third party.

6.1 Overview

FaCT protocols that require offline trusted third parties are protocols that deploy a special trusted third party, such as a KC, to generate client watermarks. The trusted third party is offline because it is only involved during initial setup and when there is a dispute between C and D. It is not involved during the actual process where marked content is provided to C.

Similar to FaCT protocols with online trusted third parties in Chapter 5, they can be designed either using the conventional approach or new approaches. The MW protocol described in Section 3.7 is an example of the traditional approach, where homomorphic encryption schemes are used for embedding the watermark in the encrypted domain.

Our focus is on new approaches. We discuss a protocol proposed by Kuribayashi and Tanaka [81] (Section 6.2), which uses symmetric encryption. We compare this protocol with a new protocol that we propose (Section 6.3), which is based on a symmetric building block known as Chameleon Encryption [1]. This work was published in [110].

6.2 The Kuribayashi-Tanaka Information Gap Protocol

Kuribayashi and Tanaka proposed a FaCT protocol with offline trusted third parties in [81]. It is interesting in that it only requires a conventional symmetric encryption scheme (e.g. AES [72]) and that any digital watermarking scheme can be used. We denote this protocol as *the KTIG protocol*.

Fundamentals. The KTIG protocol involves three parties. These are C, D and a KC. The KC plays the roles of a CA, a WCA and A. The KC is fully trusted. This protocol provides traceability, framing resistance and non-repudiation of redistribution. We note that it was claimed in [81] that the KTIG protocol can provide anonymity and unlinkability as long as the KC does not reveal the identity of C. However, there is no concrete explanation as to how this can be achieved. Thus we do not include anonymity and unlinkability in our discussion.

Environment. Since the protocol uses symmetric encryption, it seems to be suitable for the scenario where C has limited resources. The protocol also needs public key support as digital signatures are required for the purpose of non-repudiation of redistribution. We also assume that the protocol is conducted using a secure communication channel. The KC is an offline trusted third party, who is only involved during the initial setup and when there is a dispute between C and D. The main building blocks are symmetric encryption schemes, digital watermarking schemes and digital signature schemes. Table 6.1 shows the design framework of the KTIG protocol.

Table 6.1: The Design Framework of the KTIG Protocol

Fundamentals	
Parties Involved	C, D, KC = WCA + CA + A
Trust Assumptions	KC is *fully trusted*
Security Properties	Traceability (**TR**), Framing resistance (**FR**), Non-repudiation of redistribution (**NR**)
Environment	
Comp. Resources	D have ample resources Flexibility: C may have limited resources
Sec. comm. Support	Required
Pub. Key Support	Required
TTPs	offline TTP (KC)
Building Blocks	Digital watermarking scheme, symmetric encryption scheme and digital signature scheme

The main idea of the KTIG protocol is to have the content divided into many smaller packets. For each packet, an exact copy is created. One packet is embedded with watermark "0" and the copy is embedded with watermark "1". The same process is carried out for all the content packets. All these packets are encrypted with a master key by the distributor. The client has a unique key, and based on this key different marked packets are selected and decrypted in a way that a unique watermark is embedded. Since the distributor does not know which packets are selected and decrypted, the distributor cannot guess the embedded watermark. Also, since the client does not have the master key, the client is forced to use the given unique key to select and decrypt content, and hence indirectly embeds a unique watermark into content. Kuribayashi and Tanaka [81] defined this scenario as an *information gap* between C and D. In the following we describe the protocol.

`Initial Setup.` The main purpose of this phase is for C and D to obtain certified

public keys. In addition, D generates a key table as the encryption key, while the KC uses this key table to generate a unique decryption key for C. Figure 6.1 illustrates the protocol messages.

①	Initial Setup:	
	$C\&D \rightarrow$ KC : $\{Request\ authenticated\ keys\}_{AKE}$	**Before content distribution**
	KC $\rightarrow C$: $\{Cert_{ssk_{KC}}(ID_C)\}_{AKE}$	
	KC $\rightarrow D$: $\{Cert_{ssk_{KC}}(ID_D)\}_{AKE}$	
	$D \rightarrow$ KC : $\{key\ index\ table,\ \text{PRNG}\}_{AKE}$	
	KC $\rightarrow C$: $\{DK_C\}_{AKE}$	

Figure 6.1: KTIG Protocol – Initial Setup

We describe the protocol steps as follows:

(I) C and D register with the KC to obtain authenticated key materials.

1. C and D register with the KC to obtain certificates for the purpose of authenticating their identities, and as proof of registration.

(II) The KC generates and provides C and D with the authenticated key materials.

2. The KC certifies the identity information of C and D. For simplicity, we use $Cert_{ssk_{KC}}(ID_C)$ to represent the certificate for C and $Cert_{ssk_{KC}}(ID_D)$ for D.

(III) C requests a decryption key from the KC.

3. When C requests a decryption key, the KC contacts D for the key materials required to generate C's decryption key. Then D generates an index key table and chooses a *pseudo-random number generator (PRNG)*. The index key table contains many keys that are used as seeds for the PRNG. Each of these keys matches to a unique transaction between D and C. After that, D gives to the KC the index key table and the PRNG.

(IV) The KC sends a unique decryption key to C.

4. Given the index key table and the PRNG, the KC generates a key sequence by selecting a key key_C from the table as an input to the PRNG. It is assumed that ID_C contains the index number for the KC to select key_C from the table. Then the KC executes the PRNG to output a key sequence $K = (k_1^0, k_2^1), (k_3^0, k_4^1), (k_5^0, k_6^1), \ldots, (k_{2n-1}^0, k_{2n}^1)$ that contains $2n$ elements.

5. The KC generates client watermark $W \in \{0,1\}^n$ based on the client's ID_C and a random number r_C. The computation is $W = h(ID_C, r_C)$, where h is an invertible function (a practical example of such a function was not provided in [81]). The KC stores W, r_C and $h(.)$ as secrets, and stores ID_C, $Cert_{ssk_{KC}}(ID_C)$ and r_C as the client's record.

6. The KC uses W to select the client's decryption key from the key sequence K, and these selected keys DK_C are given to C. For example, if $W = (0110...0)$ then $DK_C = (k_1^0, k_4^1, k_6^1, k_7^0, \ldots, k_{2n-1}^0)$. Finally, the KC gives DK_C to C.

`Content Watermarking and Distribution.` Figure 6.2 shows the protocol messages, and the protocol steps are as follows:

② Content Watermarking and Distribution:		
$C \rightarrow D$	: $\{Cert_{ssk_{KC}}(ID_C)\}_{AKE}$	**Content distribution**
$D \rightarrow C$	: $\{\sigma[X']_{E_K}\}_{AKE}$	

Figure 6.2: KTIG Protocol – Content Watermarking and Distribution

(I) C requests content and provides D with the identity information ID_C that contains the index number for D to select keys from the index key table.

1. C requests content by sending certificate $Cert_{ssk_{KC}}(ID_C)$ to D.

(II) D produces a marked copy of the requested content and sends it to C.

2. Upon receiving the request from C, D verifies C's certificate. If the certificate is valid, D divides the required content X into n packets, $X = (x_1, x_2, \ldots, x_n)$. D then selects an index key key_C from the pre-generated index key table. This index key key_C is used as a seed for the PRNG. The PRNG generates a unique key sequence based on key_C, $K = (k_1^0, k_2^1), (k_3^0, k_4^1), (k_5^0, k_6^1), \ldots, (k_{2n-1}^0, k_{2n}^1)$.

This is identical to the key sequence generated by the KC in the `Initial Setup` phase.

3. For each packet x_i, $1 \leq i \leq n$, two packets, x_i^0 and x_i^1, are generated. Packet x_i^0 is embedded with watermark "0" and packet x_i^1 is embedded with watermark "1". In other words, the marked content is $X' = (x_1^0, x_1^1, x_2^0, x_2^1, \ldots, x_n^0, x_n^1)$, which is double in size compared to the original.

4. All marked packets are compressed and an error detection string is attached to ensure correct decryption at a later stage.

5. The marked and compressed packets are encrypted using the key sequence K based on a symmetric encryption scheme, as discussed in Section 2.3.2. The encrypted packets are

$$[X']_{E_K} = ([x_1^0]_{E_{k_1^0}}, [x_1^1]_{E_{k_2^1}}), ([x_2^0]_{E_{k_3^0}}, [x_2^1]_{E_{k_4^1}}), \ldots, ([x_n^0]_{E_{k_{2n-1}^0}}, [x_n^1]_{E_{k_{2n}^1}}).$$

6. D permutes the order of each of the encrypted pair based on a permutation σ. This permutation is to prevent C from learning whether a packet is embedded with "0" or "1" based on the location of the encrypted packet.

7. D sends the permuted, encrypted marked content $\sigma[X']_{E_K}$ to C.

8. C decrypts $\sigma[X']_{E_K}$ using the decryption key DK_C. For example, let the first permuted, encrypted marked pair be $([x_1^1]_{E_{k_2^1}}, [x_1^0]_{E_{k_1^0}})$ and the first decryption key element be k_1^0. C runs the decryption algorithm on both packets as follows:

$$\begin{aligned} \perp &\leftarrow [[x_1^1]_{E_{k_2^1}}]_{D_{k_1^0}} \\ x_1^0 &\leftarrow [[x_1^0]_{E_{k_1^0}}]_{D_{k_1^0}}, \end{aligned}$$

where $\perp$ denotes that the decryption fails since the encryption key is different from the decryption key. Recall that a symmetric encryption scheme is used. If we assume $DK_C = (k_1^0, k_4^1, k_6^1, k_7^0, \ldots, k_{2n-1}^0)$, following the decryption process for all the encrypted marked packets results in a decrypted marked content of the form:

$$x_1^0, x_2^1, x_3^1, x_4^0, \ldots, x_n^0,$$

where by combining these decrypted packets a content marked with a unique watermark $W = (0110...0)$ is formed.

`Identification and Dispute Resolution.` Figure 6.3 shows the protocol messages and the protocol steps are as follows:

③ `Identification and Dispute Resolution:`		
D	: $\{\texttt{true}, \texttt{false}\} \leftarrow [\widehat{X}, W, X]_{DET_{wmk}}$	
$D \rightarrow$ KC	: $\{W\}_{AKE}$	
KC $\rightarrow D$	: $\left\{prove\ C\ owns\ \widehat{X}\right\}_{AKE}$	**After content distribution**
$D \rightarrow$ KC	: $\{Cert_{ssk_{KC}}(ID_C)\}_{AKE}$	

Figure 6.3: KTIG Protocol – Identification and Dispute Resolution

(I) D detects a watermark from the found copy of content in order to identify the client that owns the content.

1. When an illegal copy of content $\widehat{X}$ is found, D extracts watermark W from $\widehat{X}$ and sends W to the KC.

(II) D proves to A that C illegally distributed copies of content.

2. The KC determines the identity behind the watermark W by computing $h^{-1}(W) = h^{-1}(h(ID_C, r_C)) = ID_C, r_C$.
3. After determining the identity, the KC asks D for evidence that C has requested the particular content. This is the certificate $Cert_{ssk_{KC}}(ID_C)$ (or proof of registration). If this evidence is provided, then the KC regards C as guilty.

In summary, the KTIG protocol is an efficient protocol that uses only symmetric encryption. However, the bandwidth requirement is high, since the transmitted marked content is double the size of the original content. We will analyse its security and efficiency in Section 6.4.

6.3 A Protocol based on Chameleon Encryption

In this section we describe a FaCT protocol [110] that utilises the Chameleon encryption (CE) scheme [1]. Similarly to the KTIG protocol, it does not deploy asymmetric

homomorphic encryption schemes [99]. Hence it has lower computational overhead and requires less network bandwidth than a protocol using the conventional approach.

6.3.1 Chameleon Encryption

Adelsbach *et al.* proposed the Chameleon encryption scheme in [1]. The intriguing property of this scheme is that it only involves modular addition and watermarking happens simultaneously during decryption, when a client decrypts content using his watermarked key material.

Basic Notion. We define a content space $\mathcal{X} \subseteq \mathbb{R}$. *Content* in a content space ($X \in \mathcal{X}$) is a vector of real numbers $X = (x_1, \ldots, x_n)$. Each vector element of content x_i, $1 \leq i \leq n$, is also a real number in the range $[0, z]$, where z is the maximum value allowed for x_i. We further define a watermark space $\mathcal{W} \subseteq \mathbb{R}$. A *watermark* in a watermark space ($W \in \mathcal{W}$) is a vector of real numbers $W = (w_1, \ldots, w_n)$. The range of values that each watermark element can take depends on the algorithm that is used to generate them. For example, a watermark can be *bipolar*, which means $w_i \in \{-1, 1\}$, or a watermark may be chosen from a normal (Gaussian) distribution, as in a spread spectrum watermarking scheme [28].

Overview. The basic idea behind Chameleon encryption is as follows:

1. Encryption and decryption keys are generated. The encryption key, known as the *master table* MT, contains a vector of real numbers in the same space as the content X. The decryption key, known as the *user table* UT, is a *slightly different* version of MT, consisting of elements of MT reduced by a small real number. This user table UT is generated by $UT = MT - FT$, where FT is a fingerprint table.

2. In order to encrypt content X, elements in the master table MT are selected and added to the elements in X.

3. Decryption is computed by selecting the elements from $UT = MT - FT$ and subtracting these elements from X. Since UT is slightly different from MT,

the decryption will introduce a small "error" (watermark) into the decrypted content, thus making this content unique to the holder of UT.

We adapt the notation of [1] to describe the four phases of the scheme in more detail.

Setup. In this phase, three types of table are generated:

- **Master table** MT: This is a vector of real numbers denoted by $MT = (mt_1, \ldots, mt_L)$, where $L = 2^b$ and b a positive integer. Note that $MT \in \mathcal{X}$ (the content space). Each mt_α, $1 \leq \alpha \leq L$, is randomly selected from $[0, z]$, where z is the maximum value allowed for mt_α.
- **Fingerprint tables**: These are denoted $FT^{(1)}, \ldots, FT^{(N)}$, where N is the number of clients. Each $FT^{(i)} = (ft_1^{(i)}, \ldots, ft_L^{(i)})$, $1 \leq i \leq N$, is a vector of real numbers. It is required that $ft_\alpha^{(i)} = \frac{1}{s} w_\alpha^{(i)}$, $1 \leq \alpha \leq L$, where s is a small positive integer and $w_\alpha^{(i)}$ is an element in watermark $W^{(i)} = (w_1^{(i)}, \ldots, w_L^{(i)})$. The watermark $W^{(i)}$ is generated based on a watermarking scheme such as the SS scheme described in [77]. Elements of a fingerprint table are used to watermark elements of the master table in order to generate a user table for the client.
- **User tables**: Each user table is a vector of integers $UT^{(i)} = (ut_1^{(i)}, \ldots, ut_L^{(i)})$, $1 \leq i \leq N$. It is generated as $UT^{(i)} = MT - FT^{(i)}$, where $ut_\alpha^{(i)} = mt_\alpha - ft_\alpha^{(i)} \bmod p$ for $1 \leq \alpha \leq L$, p a sufficiently large integer. Elements of a user table are used for fingerprinting and decryption of content.

Encryption. Content $X = (x_1, \ldots, x_n)$ is encrypted based on the master table MT. We assume that the value of s has been determined. In order to encrypt content, a random session key K_r is generated using a pseudo-random number generator (PRNG). This session key K_r is then used as input to a *pseudo-random sequence generator* (PRSG), which is a special type of PRNG which outputs a pseudo-random integer sequence R with bit length $n \cdot s \cdot b$ (n is the number of elements representing the content, and b is the positive integer used to determine L). Using R, $n \cdot s$ elements, $(k_1, \ldots, k_{n \cdot s})$, are randomly selected from the master table MT. Finally,

the encrypted content $E = (e_1, \ldots, e_n)$ is:

$$e_\beta = x_\beta + \sum_{j=1}^{s} k_{s(\beta-1)+j} \bmod p \text{ for } 1 \leq \beta \leq n. \tag{6.1}$$

As an example, let $s = 4$, $R = (9, L, 8, 7, 56, \ldots, 10)$, $k_1 = mt_9$, $k_2 = mt_L$, $k_3 = mt_8$ and $k_4 = mt_7$. The encryption of the content's first element x_1 will be $e_1 = (x_1 + mt_9 + mt_L + mt_8 + mt_7) \bmod p$.

Joint Fingerprinting and Decryption. Fingerprinting and decryption of content is carried out by first selecting the watermarked elements in the user table $UT^{(i)}$ based on the identical R used for encryption. We denote elements selected from $UT^{(i)}$ as $(k_1^f, \ldots, k_{n \cdot s}^f)$. Following the previous example, let $s = 4$ and $k_1^f = ut_9^{(i)}$, $k_2^f = ut_L^{(i)}$, and so on. The watermarked content $X^{(i)} = (x_1^{(i)}, \ldots, x_n^{(i)})$ is obtained during decryption of the encrypted content $E = (e_1, \ldots, e_n)$ as follows:

$$x_\beta^{(i)} = e_\beta - \sum_{j=1}^{s} k_{s(\beta-1)+j}^f \bmod p \text{ for } 1 \leq \beta \leq n. \tag{6.2}$$

Continuing from the previous example, decryption of the first element is:

$$\begin{aligned} x_1^{(i)} &= (e_1 - (ut_9^{(i)} + ut_L^{(i)} + ut_8^{(i)} + ut_7^{(i)})) \bmod p \\ &= (x_1 + ft_9^{(i)} + ft_L^{(i)} + ft_8^{(i)} + ft_7^{(i)}) \bmod p. \end{aligned}$$

Remark. Recall that for the generation of the fingerprint table we require that $ft_\alpha^{(i)} = \frac{1}{s} w_\alpha^{(i)}$, for $w_\alpha^{(i)}$ an element of a watermark $W^{(i)} = (w_1^{(i)}, \ldots, w_n^{(i)})$. So for the example shown above, we have $ft_9^{(i)} = \frac{1}{s} f_9^{(i)}$, and so on. This gives us $x_1 + \frac{1}{4} w_9^{(i)} + \frac{1}{4} w_L^{(i)} + \frac{1}{4} w_8^{(i)} + \frac{1}{4} w_7^{(i)}$, where $(\frac{1}{4} w_9^{(i)} + \frac{1}{4} w_L^{(i)} + \frac{1}{4} w_8^{(i)} + \frac{1}{4} w_7^{(i)})$ has similar random distribution to $w_1^{(i)}$, which conforms to the distribution of the underlying watermarking scheme. Due to this, robustness of the watermark embedded into content is equal to that of the watermarking scheme used.

Detection. If we denote $(g_1^{(i)}, \ldots, g_{s \cdot n}^{(i)})$ as a vector where $g_\beta^{(i)} = ft_\beta^{(i)}$, (for example, $g_1^{(i)} = ft_9^{(i)}$, $g_2^{(i)} = ft_L^{(i)}$), then a watermark $\widehat{W}^{(i)} = (\widehat{w}_1^{(i)}, \ldots, \widehat{w}_n^{(i)})$ can be extracted from a found copy of content $\widehat{X}^{(i)} = (\widehat{x}_1^{(i)}, \ldots, \widehat{x}_n^{(i)})$ using the original content $X =$

$(x_1, \ldots, x_n)$ as follows:

$$\widehat{w}_\beta^{(i)} = x_\beta - \widehat{x}_\beta^{(i)} = \sum_{j=1}^{s} g_{s(\beta-1)+j}^{(i)}, \text{ for } 1 \leq \beta \leq n. \tag{6.3}$$

Again, using the example given previously, this can be shown as $\widehat{w}_1^{(i)} = x_1 - \widehat{x}_1^{(i)}$ $= x_1 - (x_1 + ft_9^{(i)} + ft_L^{(i)} + ft_8^{(i)} + ft_7^{(i)})$. This extracted watermark $\widehat{W}^{(i)}$ is then compared with the original watermark $W^{(i)}$ to measure their similarity by:

$$\texttt{Sim}(\widehat{W}^{(i)}, W^{(i)}) = \frac{\widehat{W}^{(i)} W^{(i)}}{\sqrt{\widehat{W}^{(i)} \widehat{W}^{(i)}}}.$$

If $\texttt{Sim}(\widehat{W}^{(i)}, W^{(i)}) > t$, where t is a predetermined threshold, it means that the detection of the watermark is successful.

Remark. As mentioned in [1], the modulo operator of all the above operations only allows computation of integers, but MT, FT and X are all based on real numbers. This issue can be addressed by representing a real value as an integer by scaling. As an illustrative example, 15.687 can be represented as 15687 or 1568700, depending on the requirements of the scheme. Hence we say that there is a one-to-one mapping from $[0, z] \in \mathbb{R}$ to $[0, Z] \in \mathbb{Z}$.

6.3.2 The CE Protocol

In this section we present a FaCT protocol using the Chameleon encryption scheme with new and desirable properties. We call this *the CE protocol*.

Fundamentals. The CE protocol involves four parties. These are C, D, a KC and A. The KC is tasked to generate the Chameleon tables for C and D. It also plays the role of a CA. The KC and A are fully trusted. This protocol provides traceability, framing resistance and non-repudiation of redistribution.

Environment. The CE protocol assumes that D has ample computing resources since many clients may contact D at the same time to request content. However, C can have limited computing resources due to the computational efficiency of the Chameleon encryption scheme. It also requires public key support and a secure channel. The KC is an offline trusted third party and is only involved in the initial

setup phase and when there is a dispute between C and D. The main building blocks are digital watermarking schemes, the Chameleon encryption scheme and digital signature schemes. Table 6.2 shows the design framework.

Table 6.2: The Design Framework of the CE Protocol

Fundamentals	
Parties Involved	C, D, A, KC = CA + WCA
Trust Assumptions	KC and A are *fully trusted*
Security Properties	Traceability (**TR**), Framing resistance (**FR**), Non-repudiation of redistribution (**NR**)
Environment	
Comp. Resources	D have ample resources Flexibility: C may have limited resources
Sec. comm. Support	Required
Pub. Key Support	Required
TTPs	offline TTP (KC)
Building Blocks	Digital watermarking scheme, Chameleon encryption scheme and digital signature scheme

`Initial Setup.` In this phase, D and C register their details with the KC, including registering their public keys for digital signature schemes. More importantly, the KC provides D and C with Chameleon encryption and decryption key materials that can be used for many protocol sessions. Figure 6.4 shows the protocol messages and in the following we describe the protocol steps:

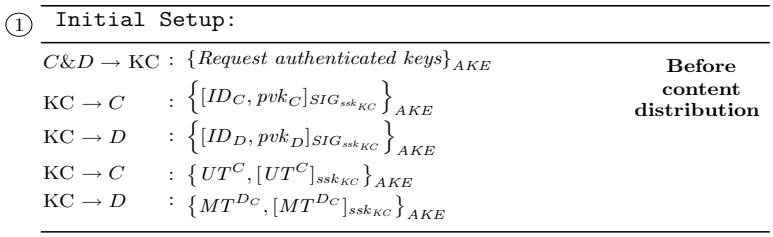

Figure 6.4: CE Protocol – Initial Setup

(I) C and D register with the KC to obtain authenticated key materials.

1. C and D send their respective public verification keys pvk_C and pvk_D, together with their identity information to the KC.

(II) The KC generates and provides C and D with the authenticated keys.

2. Upon receiving the registration requests from C and D, the KC signs the public verification keys together with C's and D's identity information. The resulting signatures, as shown in Figure 6.4, are passed to C and D.

(III) C and D request key materials from the KC.

3. Upon receiving the registration requests from C and D, the KC runs the **Setup** phase of the Chameleon encryption scheme to produce a master table MT^{D_C}, a client fingerprint table FT^C and a client user table $UT^C = MT^{D_C} - FT^C$. The master table MT^{D_C} and the user table UT^C are signed by the KC, resulting in two signatures $[MT^{D_C}]_{ssk_{KC}}$ and $[UT^C]_{ssk_{KC}}$.

(IV) The KC sends the key materials to C and D.

4. The master table MT^{D_C} is passed to D. This master table is intended for D to encrypt content specifically for C. The user table UT^C is passed to C. From here onwards C and D can then use their respective user table and master table to conduct many `Content Watermarking and Distribution` sessions without further contacting the KC. This means key distribution is a one-time process.

`Content Watermarking and Distribution.` In this phase C requests content from D, and D watermarks content and sends the marked content to C. Figure 6.5 shows the protocol messages and we describe the protocol steps as follows:

②	`Content Watermarking and Distribution:`	
$C \rightarrow D$	$: \left\{ pvk_C, AGR, [ID_C, pvk_C]_{SIG_{ssk_{KC}}}, [AGR]_{SIG_{ssk_C}} \right\}_{AKE}$	
$D \rightarrow C$	$: \left\{ K_r, E^v, [K_r, E^v, [AGR]_{SIG_{ssk_C}}, AGR]_{SIG_{ssk_D}} \right\}_{AKE}$	**Content distribution**

Figure 6.5: CE Protocol – Content Watermarking and Distribution

(I) C requests content and approves a content agreement with D.

1. C sends a content request to D. This contains C's public key and the signature generated by the KC. At the same time C signs an agreement AGR that

contains the description of content and the licensing terms. The resulting signature is $[AGR]_{SIG_{ssk_C}}$. This agreement may be put up on the website owned by D. This signature is also sent to D.

(II) D produces a marked copy of the requested content and sends it to C.

2. D verifies $[AGR]_{SIG_{ssk_C}}$ and generates a client's watermark V^C. This watermark V^C is embedded into content X. We denote content with V^C embedded as $X^v = (x_1^v, \ldots, x_n^v)$.

$$X^v \leftarrow [X, V^C]_{EMB_{wmk_V}}.$$

The reason for embedding V^C is so that D is able to trace and match C with the client's record from copies of content that D has found. In this case D can generate and embed V^C using any watermarking scheme. After that, D runs the **Encryption** phase of the Chameleon encryption scheme. Recall from Section 6.3.1 that this means D generates K_r using a PRNG, and K_r is then used as an input to a PRSG to generate a pseudo-random sequence R. Elements of master table MT^{D_C} are chosen based on R to encrypt X^v, as shown in (6.1). We denote the encrypted marked content as E^v. This is then signed together with K_r, the agreement AGR and C's signature $[AGR]_{SIG_{ssk_C}}$. The signature is represented as:

$$[K_r, E^v, [AGR]_{SIG_{ssk_C}}, AGR]_{SIG_{ssk_D}}.$$

This signature is then sent together with K_r and E^v to C.

3. C verifies $[K_r, E^v, [AGR]_{SIG_{ssk_C}}, AGR]_{SIG_{ssk_D}}$ and runs the **Joint Fingerprinting and Decryption** phase of the Chameleon encryption scheme. This means C uses K_r as input to the PRSG to generate R, and elements of his user table UT^C are selected based on R to decrypt E^v. At the same time as decryption, a second watermark W^C is embedded into this content, as shown in (6.2). We denote the final decrypted content with two embedded watermarks as X^{vw}.

`Identification and Dispute Resolution.` Figure 6.6 shows the protocol messages and the protocol steps are presented as follows:

③	Identification and Dispute Resolution:	
D	: $\{\texttt{true}, \texttt{false}\} \leftarrow [\widehat{X}, V^C, X]_{DET_{wmk}}$	
$D \rightarrow A$	: $\{X^v, K_r, E^v, \widehat{X}, AGR, [K_r, E^v, [AGR]_{SIG_{ssk_C}}, AGR]_{SIG_{ssk_D}}, [AGR]_{SIG_{ssk_C}}\}_{AKE}$	
$A \rightarrow$ KC	: $\{W\}_{AKE}$	**After content distribution**
KC	: $\texttt{Sim}(W, W^C)$	
KC $\rightarrow A$	: $\{\textit{similar/not similar}\}_{AKE}$	

Figure 6.6: CE Protocol – Identification and Dispute Resolution

(I) D detects a watermark from the found copy of content in order to identify the client that owns the content.

1. When a suspicious copy of content $\widehat{X}$ is found, D tries to detect watermark V^C based on the watermarking scheme deployed. If the watermark is detected, then D proceeds to match V^C to the client's identity from their records.

(II) D proves to A that C illegally distributed copies of content.

2. After identifying C based on the watermark V^C, D proves to a third party that C has illegally distributed copies of content. In this case, D sends X^v, K_r, E^v, $\widehat{X}$, AGR, $[K_r, E^v, [AGR]_{SIG_{ssk_C}}, AGR]_{SIG_{ssk_D}}$ and $[AGR]_{SIG_{ssk_C}}$ to A. After A verifies the signatures to confirm the agreement between C and D on the content, A extracts W from $\widehat{X}$ based on (6.3). After that, A gives W to the KC and requests the KC to run $\texttt{Sim}(W, W^C)$. If the watermark is detected, the KC retrieves the associated identity of C and sends the information to A for A to decide whether C is guilty or not.

6.3.3 Alternative Approaches

We note that there are alternative ways in which this Chameleon encryption approach to fair content tracing could be deployed:

- *CE protocol with always online KC.* If the KC is always online, it is possible to design the protocol so that there is no need for D and C to store the master table MT^{D_C} and the user table UT^C, respectively. This is achieved by the

KC holding all three tables. When C wishes to obtain content, D contacts the KC with authenticated information of C. Then the KC generates the pseudo-random sequence R and uses the sequence to select the elements from MT^{D_C} and UT^C. Every s elements are added together. These newly selected and added elements from MT^{D_C} (and UT^C) become the encryption key (and joint fingerprinting and decryption key) for D (and C).

- *CE protocol with reduced client-KC contact.* If contacting the KC by C is an issue during key distribution, the protocol can be modified so that the user table UT^C is delivered to C through D. This can be done by the KC encrypting and signing the user table UT^C based on conventional cryptographic primitives, and giving UT^C to D, who then forwards the table to C. In this case, C only needs to communicate with D for the entire execution of the protocol.

The CE protocol would thus seem to offer some interesting trade-offs between storage restrictions and operational contact with a trusted third party, allowing it to be tailored to different application environments. In summary, the CE protocol mitigates the requirement of a homomorphic encryption scheme. At the same time, alternative designs can be used for different application requirements. By doing so it provides several advantages compared to conventional protocols using homomorphic encryption schemes such as the MW and LYTC protocols (see Sections 3.7 and 5.2). It also holds a few advantages when compared to the WP and KTIG protocols (see Sections 5.3 and 6.2). However, it does involve certain trade-offs, especially on storage size. We will discuss these in the next section.

6.4 Analysis

In the following we give security analysis and performance comparisons of the protocols that we have discussed.

6.4.1 Security

In this section we analyse the security of the FaCT protocols we have discussed.

Traceability. In the KTIG protocol, traceability is assured by the binary watermark W embedded during the selection of content packets and the decryption of the encrypted packets by C. D can extract this watermark W when an illegal copy is found and ask the KC to reveal C's identity.

In the CE protocol, traceability is assured, since D can trace its content to C through the watermark V^C, and also that the KC can detect watermark W^C to identify C. In addition, C has no knowledge of W^C and thus cannot simply remove the watermark from content. C may attempt to remove W^C based on knowledge of the user table UT^C of the Chameleon encryption scheme. Such an attempt is identical to removing a watermark W from marked content, which subsequently means defeating the robustness and collusion resistance of the underlying digital watermarking scheme.

The reason is that the user table UT^C is generated by subtracting the fingerprint table $FT^{(i)}$ from the master table MT^{D_C}, and the master table MT^{D_C} is derived from the same space as content X, that is $MT^{D_C} \in \mathcal{X}$. Similarly, the elements in the fingerprint table $FT^{(i)}$ have the same statistical properties of a watermark generated from a watermarking scheme. Although the value of the elements in $FT^{(i)}$ have been fractioned by s, as mentioned in [1], this can be compensated by generating a higher number of elements for $FT^{(i)}$, which means having a larger value for L. So, given $MT^{D_C} \in \mathcal{X}$ and $FT^{(i)} \in \mathcal{W}$, the user table UT^C is statistically identical to a marked content, and our earlier statement holds.

We also note that Chameleon encryption has been shown to be *semantically secure*, which means that an attacker, who is not any of the parties involved in the protocol, learns nothing from the encrypted content [1]. In addition, traceability may be improved if a more effective watermark coding method is devised, such as the recent improvement in watermark coding for Chameleon encryption proposed in [76].

Framing Resistance. In the KTIG protocol, only the KC knows watermark W. D has no knowledge of W since D does not have the unique key given to C. Without this key D cannot know which content packets are selected and decrypted by C and thus has no way to determine the watermark W embedded into content. It also relies on the secrecy of the invertible function $h(.)$. Kuribayashi and Tanaka did not consider and discuss the scenario where D extracts watermark from a found copy of content and then embeds this watermark into other contents to frame C, and hence

the KTIG protocol is susceptible to the *unbinding attack*.

In the CE protocol, D may try to learn the watermark W^C based on his master table MT^{D_C}. We note that this is equivalent to learning a watermark W generated using a watermarking scheme based on only the original content X. Recall that $X \in \mathcal{X}$ and $MT^{D_C} \in \mathcal{X}$ (Section 6.3.1). This means that the master table MT^{D_C} has identical statistical distribution to content X. Similarly, W^C has the same distribution as a watermark W generated using a watermarking scheme. Hence guessing W^C given MT^{D_C} is akin to guessing W with only possession of the original content X. This brings us to the identical framing resistant settings of the conventional FaCT protocols such as the LYTC protocol, where D has the original content X, but the watermark W is generated by other parties.

The CE protocol also addresses the unbinding attack, since there is a binding statement that links the marked copy X^{vw} received by the client (and all similar copies where W^C can be detected) to the agreement AGR. This is achieved through the signature $[K_r, E^v, [AGR]_{SIG_{ssk_C}}, AGR]_{SIG_{ssk_D}}$ generated on the marked copy, the client's signature and the agreement. Thus framing resistance is assured as long as the marked copy X^{vw} and the user table UT^C are kept secret by C.

Non-repudiation of Redistribution. In the KTIG protocol, non-repudiation of redistribution is provided through the successful detection of watermark W by the KC. Furthermore, given the detected W, the KC can reveal the identity of C based on C's registration information and the records stored by the KC.

In the CE protocol, C cannot repudiate that he illegally distributed content because:

- there exists a signature $[AGR]_{SIG_{ssk_C}}$, which binds C to the agreement AGR that specifies the content X^{vw}.
- when the watermark W, which can be matched to C, is detected, the identity of C is revealed by the KC.

Based on this information, A demonstrates that C is guilty. Note that the agreement AGR plays an important role as it contains the description of content that will eventually be used to confirm that the illegal content found $\widehat{X}$ is indeed a copy of the original X (which is similar to X^{vw}). Thus, C cannot deny hav-

ing illegally distributed copies of content if A, with information gathered from D and the KC, demonstrates that this is the case based on the verified signature $[K_r, E^v, [AGR]_{SIG_{ssk_C}}, AGR]_{SIG_{ssk_D}}$ and the detected watermark W.

Summary. Assuming the security of the underlying building blocks, traceability, framing resistance and non-repudiation of redistribution are provided in the KTIG and CE protocols. However, for the KTIG protocol, in addition to the security of the underlying building blocks, security relies on keeping the invertible function $h(.)$ secret. Unfortunately, Kuribayashi and Tanaka did not provide details as to how to construct the invertible function $h(.)$.

Table 6.3: Summary of the Security Analysis

Protocols	TR	FR	NR	Conditions
KTIG	✓	✓	✓	Relies on $h(.)$ being a secret function. Susceptible to unbinding attack.
CE	✓	✓	✓	Relies on the security of Chameleon encryption.

6.4.2 Efficiency

In this section we examine the performance of the protocols that we have discussed and the efficiency comparison between them is shown in Table 6.4. For the purpose of our discussion, as stated in Section 6.3.1, L is the number of entries in MT.

Bandwidth. For the CE protocol, due to the mechanism in Chameleon encryption as presented in Section 6.3.1, each encrypted element of content has similar size to the original element. If we assume the value for $|Z|$ is 16 or 32 bits (e.g. images or video frame), and $n = 10000$, then the marked content size is $10000 \cdot 16$ bits ≈ 20 KByte or $10000 \cdot 32$ bits ≈ 40 MByte, which is much smaller compared to the content size in the MW protocol previously described in Section 3.7 and other protocols that deploy asymmetric homomorphic encryption schemes to embed watermarks in the encrypted domain.

For the KTIG protocol, the size of the encrypted marked content is $2n|Z|$. This is assuming that there are n packets of content and the size of each packet is $|Z|$ bits. The size of the encrypted marked content is double the original content size (i.e. $n|Z|$) because for each packet of content, an exact copy is produced, as illustrated in the `Content Watermarking and Distribution` phase in Section 6.3.

Trusted Third Parties. Both the KTIG and CE protocols deploy an offline trusted third party. This means that the trusted third party can be offline after necessary information (i.e. key materials) are provided to the distributor and/or the client in the `Initial Setup` phase.

Furthermore, the adoption of Chameleon encryption in the CE protocol means that once the key materials are acquired, C can subsequently request content from D many times, without further involving the trusted third party. This is beneficial compared to the KTIG protocol, where for each content request by C, new registration information and a new key must be obtained by C from the trusted third party.

Computation. In the KTIG protocol, the computational requirements are the embedding of the watermark and the symmetric encryption of the content packets. We denote symmetric encryption by **S**, as described in Section 3.6. We assume that there are $2n$ content packets, with n original content packets and n duplicates. This is required, as described in Section 6.3, where one packet is embedded with watermark "0", and another identical packet is embedded with watermark "1". So for D, computations involve $2n\mathbf{A}$ to embed the watermarks into the $2n$ packets and $2n\mathbf{S}$ to encrypt them. While for C, the computation involves decryption of content, where the worst case is to decrypt all $2n$ packets to get the correct packets of content. Hence C needs $2n\mathbf{S}$ computations.

For the CE protocol, D requires $ns\mathbf{A}$ computation to encrypt the content, where n is the number of elements in the watermark and content, while s is the number of elements selected from the master table. As described in Section 6.3.1, for each element of content, s elements from the master table are added, so all together there are ns additions. Assuming subtraction has the same computational requirement to addition, then C also requires $ns\mathbf{A}$ computation to decrypt, while simultaneously embedding a watermark into content.

Storage. For the KTIG protocol, D needs to store the index key table and the watermark used to mark each of the content packets. Since Kuribayashi and Tanaka did not provide much detail on the generation of this table, we hypothetically assume that there are L elements in the table, and each element has bit size $|Z|$, following the same notation as the CE protocol. Hence D stores $L|Z|$ bits of key table and $n|Z|$ bits of watermark. C only requires the key DK_C containing n elements to

decrypt the encrypted marked content. For simplicity, we assume that each element in this key has bit size $|Z|$. So C requires $n|Z|$ bits to store the key. However, C can optionally purge this key after the encrypted marked content is decrypted. If this is the case then C does not require any storage in terms of obtaining the marked content. This is shown as (0) in Table 6.4.

For the CE protocol, D stores the master table MT^{D_C} and C stores the user table UT^C. Since each of these tables has L elements, and each element has value Z, the storage requirement for D and C is $L|Z|$. As an illustration, in the following we borrow a simple but practical example from [1]. We assume a content image with $n = 10000$ significant coefficients, where each coefficients is of length 16 bits. Then $Z = 2^{16}$. Suppose $L = 8Z$ then $L = 8(2^{16}) = 2^{19}$, where $L = 8Z$ achieves the statistical quality required for the master table MT^{D_C} as mentioned in [1]. Hence the size of the tables MT^{D_C} and UT^C for D and C is $L|Z| = 8(16)(2^{16}) = 2^{23}$ bits = 1 MByte, which can be acceptable in terms of current storage capacity. However, as we have also discussed in the alternative approaches in Section 6.3.3, if the KC is always online, then the key size will be $n|Z| = 10000(16)$ bits = 20KBytes, which is much smaller. This is because the KC stores all tables, including MT^{D_C} and UT^C, and only passes the selected and added key streams from MT^{D_C} to D, and the key streams from UT^C to C.

Table 6.4: Efficiency Comparisons between Protocols with offline Trusted Third Parties

Pro.	*Bandwidth*[1]	*TTP*	*Computation*[2]	*Storage*[1]
KTIG	$\sigma[X']_{E_K} = 2n\|Z\|$	offline KC	C: $2n\mathbf{S}$ D: $2n(\mathbf{A} + \mathbf{S})$	C: $n\|Z\|(0)$ D: $(L+n)\|Z\|$
CE	$E^v = n\|Z\|$	offline KC[3]	C: $ns\mathbf{A}$ D: $ns\mathbf{A}$	C: $L\|Z\|$ $(n\|Z\|)$ D: $L\|Z\|$ $(n\|Z\|)$

[1] $|Z| < |m|$, $|n| < |L|$
[2] $\mathbf{E}$=$O(k^3)$, $\mathbf{M}$=$O(k^2)$, $\mathbf{A}$=$O(k)$, $\mathbf{S}$=$\mathbf{E}/100$
[3] *One-off process*

Summary. From Table 6.4 and the above analysis, we observe that the KTIG protocol is efficient as it only involves a symmetric encryption scheme, but requires the content size to be doubled. For the CE protocol, its major advantage is the one-off process by the KC in providing the key materials to D and C. This means that after this process, the KC stays offline since C can request many contents in many interactions with D using the key materials provided. This is in contrast to the KTIG protocol where, for each content request, C needs to contact the KC (or the WCA). In addition, it has more assurance in term of security since the underlying

Chameleon encryption scheme has been shown to be semantically secure. We do note that both the KTIG and CE protocols come at the expense of greater key storage requirements, as shown in Table 6.4.

6.5 Summary

In this chapter we examined FaCT protocols with offline trusted third parties, where a KC is tasked to generate client watermarks or key materials. We examined the KTIG protocol [81] and compared its security and efficiency with a protocol based on Chameleon encryption [110] that we proposed. We showed that our proposal, as compared to the KTIG protocol, is advantageous in the sense that the initial setup is a one-time process. This means that once C obtains the key material, C can request many contents from D without needing to communicate with the KC. Our proposal is also more computationally efficient compared to protocols that deploy asymmetric homomorphic encryption.

Chapter 7

FaCT Protocols with Trusted Hardware

Contents

This chapter investigates FaCT protocols that deploy trusted hardware. We discuss one existing proposal and propose two protocols based on the Trusted Platform Module (TPM). The existing proposal assumes general trusted hardware without discussion on the realisation of such hardware. Hence our proposals provide more explicit constructions. Protocols with trusted hardware are advantageous for distributed computing environments as there is no central trusted third party to be contacted during content distribution.

7.1 Overview

FaCT protocols with trusted hardware deploy hardware that is normally tamper-proof and is either embedded in the distributor's computing device or in the client's computing device (or both).

Tomsich and Katzenbeisser [129] proposed one of the earliest approaches to protecting content distribution using trusted hardware. A general infrastructure that can be deployed for copyright protection of content is suggested. It was mentioned that the infrastructure can also be used for content tracing. The basic idea of the infrastructure is for either the client or the distributor to deploy trusted hardware in their computing platform. The watermark is generated by this trusted hardware in such a way that it is not possible that it can be manipulated by the client or the distributor.

A recent FaCT protocol proposed by Fan *et al.* [46] uses a similar idea by letting the trusted hardware generate the watermark. However, it deals with the trusted hardware only at an abstract level, without providing a practical example of trusted hardware that can be deployed. Hence it can be seen as incomplete. We examine this protocol in Section 7.2.

In view of this incompleteness, We then propose two new protocols based on the Trusted Computing Platform [6, 95]. One of these protocols, which is based on Direct Anonymous Attestation, was jointly devised by Adrian Leung and published in [86]. Our proposal deploys a Trusted Platform Module (TPM), which is the trusted hardware defined under the Trusted Computing Platform. The technology has been included as an integral component in many computing devices such as laptops [126]. In addition, all the protocols we describe provide anonymity and unlinkability.

7.2 The Fan-Chen-Sun Protocol

Fan, Chen and Sun (FCS) proposed a protocol that deploys trusted hardware in [46].

Fundamentals. It involves five parties and trusted hardware T_D that resides in the computing platform of D. The five parties are C, D, a CA, a WCA and A. The

CA, WCA and A are fully trusted. The FCS protocol provides traceability, framing resistance, non-repudiation of redistribution and anonymity and unlinkability.

Environment. As C and D are required to perform homomorphic encryption for watermarking in the encrypted domain, we assume that both of them have ample resources. It also requires public key support, since in the protocol C and D generate and verify digital signatures for the purpose of data origin authentication and non-repudiation of redistribution. A secure communication channel is implicitly assumed to be in place. Trusted hardware T_D is responsible for generating client watermarks, and possesses the WCA's public and private keys. The main building blocks are digital watermarking schemes, homomorphic encryption schemes and digital signature schemes. Table 7.1 shows the design framework of the FCS protocol.

In general, the FCS protocol follows the design of the LYTC protocol described in Section 5.2, but replaces the communication with the watermark certification authority WCA by communication with the trusted hardware T_D.

Table 7.1: The Design Framework of the FCS Protocol

Fundamentals	
Parties Involved	C, D, A, CA, WCA
Trust Assumptions	CA, WCA and A are *fully trusted*
Security Properties	Traceability (**TR**), Framing resistance (**FR**), Non-repudiation of redistribution (**NR**), Anonymity and unlinkability (**AU**)
Environment	
Comp. Resources	Assume D and C have ample resources
Sec. comm. Support	Required
Pub. Key Support	Required
TTPs	Trusted hardware T_D
Building Blocks	Digital watermarking scheme, homomorphic encryption scheme and digital signature scheme

`Initial Setup.` This phase is identical to the `initial setup` phase of the LYTC protocol (see Section 5.2). In other words, C and D obtain certified public keys from the CA, as shown in Figure 7.1. In addition, C obtains anonymous keys so that he can request content from D without revealing his identity. We note that C has long term signature key pairs (ssk_C, pvk_C) and encryption key pairs (hdk_C, hek_C), anonymous signature key pairs (ssk_C^*, pvk_C^*) and encryption key pairs (hdk_C^*, hek_C^*), and an anonymous certificate $Cert_{ssk_{CA}}(pvk_C^*, hek_C^*)$.

It is also assumed that the trusted hardware T_D holds the WCA signing and encryption key pairs, (ssk_{WCA}, pvk_{WCA}) and (hdk_{WCA}, hek_{WCA}). These key pairs may be negotiated between the WCA and the CA even before this phase, so that the keys can be embedded securely in the trusted hardware T_D. Figure 7.1 illustrates the protocol messages.

① Initial Setup:		
$C\&D \rightarrow$ CA	: $\{Request\ authenticated\ keys\}_{AKE}$	**Before content distribution**
CA $\rightarrow C$	: $\left\{[hek_C, pvk_C, ID_C]_{SIG_{ssk_{CA}}}\right\}_{AKE}$	
CA $\rightarrow D$	: $\left\{[hek_D, pvk_D, ID_D]_{SIG_{ssk_{CA}}}\right\}_{AKE}$	
$C \rightarrow$ CA	: $\left\{pvk_C^*, hek_C^*, [pvk_C^*, hek_C^*]_{SIG_{ssk_C}}, [hek_C, pvk_C, ID_C]_{SIG_{ssk_{CA}}}\right\}_{AKE}$	
CA $\rightarrow C$	: $\{Cert_{ssk_{CA}}(pvk_C^*, hek_C^*)\}_{AKE}$	

Figure 7.1: FCS Protocol – Initial Setup

`Content Watermarking and Distribution.` This is assisted by the trusted hardware T_D in the computing device of D. Figure 7.2 shows the protocol messages and in the following we present the protocol steps:

② Content Watermarking and Distribution:		
$C \rightarrow D$	: $\{[Cert_{ssk_{CA}}(pvk_C^*, hek_C^*), [pvk^*, hek^*, ID_C]_{SIG_{ssk^*}}]_{HE_{hek_{WCA}}}$, $AGR, pvk^*, hek^*, [pvk^*, hek^*, AGR]_{SIG_{ssk^*}}\}_{AKE}$	
$D \rightarrow T_D$	: $[Cert_{ssk_{CA}}(pvk_C^*, hek_C^*), [pvk^*, hek^*, ID_C]_{SIG_{ssk^*}}]_{HE_{hek_{WCA}}}$	
$T_D \rightarrow D$	: $[W]_{HE_{hek^*}}, [W]_{HE_{hek_{WCA}}}$, $[[W]_{HE_{hek^*}}, pvk^*, hek^*, [W]_{HE_{hek_{WCA}}}]_{SIG_{ssk_{WCA}}}$	
$D \rightarrow C$	: $\{[X'']_{HE_{hek^*}}\}_{AKE}$	**Content distribution**

Figure 7.2: FCS Protocol – Content Watermarking and Distribution

(I) C requests content and approves a content agreement with D.

1. C requests content from D by first checking the purchase agreement AGR displayed on D's website. This AGR states the rights and licensing terms of the specific content.

2. C randomly generates one-time signature and encryption key pairs (pvk^*, ssk^*) and (hek^*, hdk^*). C signs the agreement AGR and the randomly generated public keys as $[pvk^*, hek^*, AGR]_{SIG_{ssk^*}}$. C also signs C's identity and the anonymous public keys as $[pvk^*, hek^*, ID_C]_{SIG_{ssk^*}}$ and encrypts this signature

and $Cert_{ssk_{CA}}(pvk^*_C, hek^*_C)$ using the WCA's public encryption key:

$$[Cert_{ssk_{CA}}(pvk^*_C, hek^*_C), [pvk^*, hek^*, ID_C]_{SIG_{ssk^*}}]_{HE_{hek_{WCA}}}.$$

C sends this encrypted message, AGR, pvk^*, hek^* and $[pvk^*, hek^*, AGR]_{SIG_{ssk^*}}$ to D.

(II) D and the trusted hardware generate the watermark. The trusted hardware verifies the watermark to be well-formed.

3. Upon receiving the message from C, D first verifies the signature and checks that the certificate is valid. If this is the case then D inputs the encrypted message,

$$[Cert_{ssk_{CA}}(pvk^*_C, hek^*_C), [pvk^*, hek^*, ID_C]_{SIG_{ssk^*}}]_{HE_{hek_{WCA}}},$$

into the trusted hardware T_D.

4. The trusted hardware T_D decrypts the encrypted message and verifies the signature $[pvk^*, hek^*, ID_C]_{SIG_{ssk^*}}$. If the signature is valid, T_D generates a client watermark W. Then T_D encrypts the watermark using hek^* and hek_{WCA} based on a homomorphic encryption scheme, resulting in $[W]_{HE_{hek^*}}$ and $[W]_{HE_{hek_{WCA}}}$. The trusted hardware T_D also generates a signature:

$$[[W]_{HE_{hek^*}}, pvk^*, hek^*, [W]_{HE_{hek_{WCA}}}]_{SIG_{ssk_{WCA}}}.$$

The two encrypted watermarks and the signature are returned to D.

(III) D produces a marked copy of the requested content and sends it to C.

5. Upon receiving the encrypted watermarks and the signature from the trusted hardware T_D, D generates a unique watermark V specific to this transaction. D embeds V into content X using any preferred digital watermarking scheme:

$$X' \leftarrow [X, V]_{EMB_{wmk_V}}.$$

This watermark V is required so that when a copy of content is found, D can detect V to identify C. Next D embeds watermark W into the content

element-by-element using the encrypted watermark provided by the trusted hardware as follows:

$$\left.\begin{array}{ll} & [x_i']_{HE_{hek^*}} \cdot [w_i]_{HE_{hek^*}} \\ = & [x_i' \circ w_i]_{HE_{hek^*}} \\ = & [x_i^{''}]_{HE_{hek^*}} \end{array}\right\} \; 1 \leq i \leq n,$$

where n represents the number of elements in the watermark and content, and $\circ$ represents either modular addition, modular multiplication or bit-wise XOR depending on the underlying homomorphic encryption scheme. The encrypted marked content is denoted by:

$$[X'']_{HE_{hek^*}} = ([x_1^{''}]_{HE_{hek^*}}, [x_2^{''}]_{HE_{hek^*}}, \ldots, [x_n^{''}]_{HE_{hek^*}}).$$

D stores in his database entry the watermark V, the agreement AGR, the client's anonymous keys, all the signatures and encrypted watermarks. Finally, D sends $[X'']_{HE_{hek^*}}$ to C.

6. C decrypts $[X'']_{HE_{hek^*}}$ to obtain the marked content X''.

`Identification and Dispute Resolution.`

(I) D detects a watermark from the found copy of content in order to identify the client that owns the content.

1. When an illegal copy of content $\widehat{X}$ is found, D detects watermark V from this copy. If V is detected, then D can identify the perpetrator that distributed $\widehat{X}$ illegally.

(II) D proves to A that C illegally distributed copies of content.

2. D sends the encrypted watermarks, the agreement, client certificate and the signatures stored in the database, along with the found copy $\widehat{X}$ to A. This is shown in Figure 7.3.

3. A verifies the certificate and signatures. If these are valid, A asks the WCA to decrypt $[W]_{HE_{hek_{WCA}}}$ (the protocol assumes that the trusted hardware T_D and the WCA shares the same encryption key pairs). Next A performs a

correctness check on $[W]_{HE_{hek^*}}$. This is done by encrypting the watermark W obtained from the WCA and comparing the result with $[W]_{HE_{hek^*}}$ received from D. If both are identical then A proceeds to detect watermark W from the found copy $\widehat{X}$:

$$\texttt{true} \leftarrow [\widehat{X}, W, X']_{DET_{wmk}}.$$

If the detection returns `true`, A asks the WCA again to decrypt

$$[Cert_{ssk_{CA}}(pvk_C, hek_C), [pvk^*, hek^*, ID_C]_{SIG_{ssk^*}}]_{HE_{hek_{WCA}}}$$

to reveal the identity of C.

③ `Identification and Dispute Resolution:`

D	: $\{\texttt{true}, \texttt{false}\} \leftarrow [\widehat{X}, V, X]_{DET_{wmk}}$	
$D \rightarrow A$	: $\{[Cert_{ssk_{CA}}(pvk_C^*, hek_C^*), [pvk^*, hek^*, ID_C]_{SIG_{ssk^*}}]_{HE_{hek_{WCA}}},$ $AGR, pvk^*, hek^*, [pvk^*, hek^*, AGR]_{SIG_{ssk^*}},$ $[[W]_{HE_{hek^*}}, pvk^*, hek^*, [W]_{HE_{hek_{WCA}}}]_{SIG_{ssk_{WCA}}},$ $[W]_{HE_{hek^*}}, [W]_{HE_{hek_{WCA}}}, X', \widehat{X}\}_{AKE}$	**After content distribution**
$A \rightarrow$ WCA	: $\{watermark\ info?\}_{AKE}$	
WCA $\rightarrow A$	: $\{watermark\ info\}_{AKE}$	
A	: $\texttt{true} \leftarrow [\widehat{X}, W, X']_{DET_{wmk}}$	
$A \rightarrow$ WCA	: $\left\{[Cert_{ssk_{CA}}(pvk_C^*, hek_C^*), [pvk^*, hek^*, ID_C]_{SIG_{ssk^*}}]_{HE_{hek_{WCA}}}\right\}_{AKE}$	
WCA $\rightarrow A$	: $\{ID_C\}_{AKE}$	

Figure 7.3: FCS Protocol – Identification and Dispute Resolution

In summary, the main idea of the FCS protocol is to shift the responsibility of the WCA in the LYTC protocol to trusted hardware embedded in the computing device of D. Although this seems to avoid the requirement of a centralised trusted third party (i.e. the WCA), it actually only moves the requirement to centralised trusted hardware. This is because now the trusted hardware T_D is responsible for processing many client requests and hence has similar responsibility of that of a WCA.

7.3 Protocols based on TPM

In this section, we offer a practical approach to designing protocols with trusted hardware based on the Trusted Platform Module (TPM) defined in the trusted computing platform initiative [95, 126]. A trusted computing platform is a computing device that has a TPM embedded and activated. The trusted computing

platform initiative has gained major support from the key players in industry and is also gaining increased interest for practical applications [126]. Therefore, designing FaCT protocols based on a TPM can take advantage of the many well-defined properties in this trusted computing platform compared to the general infrastructure of Tomsich-Katzenbeisser mentioned in Section 7.1 and the abstract trusted hardware of the FCS proposals described in Section 7.2.

7.3.1 Trusted Platform Modules

A TPM serves as the foundation of trust, which is termed the *root of trust* [6]. It means that a computing platform (e.g. a mobile device or laptop) that has a TPM embedded in it can use the TPM to convince others of its trustworthiness.

The TPM is also used to measure the integrity of all the processes and software in the computing platform [6, 95]. In other words, the TPM also functions as a checkpoint to validate whether a particular software or process adheres to the security requirements of the computing platform. It is only when this software or process is validated that it is allowed to run. This validation (or measurement) is performed based on the *integrity measurement, storage and reporting (IMSR)* mechanisms within the TPM [86]. The measurement information is known as the *integrity metrics* and is stored in the TPM. The core components of the IMSR are the three roots of trust known as *root of trust for measurement (RTM)*, *root of trust for storage (RTS)* and *root of trust for reporting (RTS)*. Together they play the crucial role of ensuring that the operations (e.g. the execution of the software) in the computing platform can be trusted. Through these roots of trust, any malicious attempts to exploit the execution of the software will be detected. We now describe in more detail the TPM and the roots of trust:

TPM Endorsement Key and Identities. The TPM can be viewed as an enhanced smart card and contains an encryption key pair known as the *endorsement key* (EK). Together with a digital certificate, this key pair serves as proof that the TPM contained in a computing platform is genuine. The public half of the key is used for encrypting messages. The private half of the key never leaves the TPM and is used for decrypting encrypted messages. The EK is generated and inserted into the TPM by a CA, which in this case is normally the TPM manufacturer.

This EK does not serve as the identity of the TPM. To create an identity, the TPM generates another key pair, known as the *attestation identity key* (AIK). This is associated with the public half of the endorsement key in such a way that an attestation authority (i.e. a direct anonymous attestation issuer [18] or a privacy CA [6]) is convinced of the TPM's identity. An attestation authority has a similar role to a certificate authority who issues anonymous certificates. This attestation authority then *attests* the TPM's identity by generating a certificate so that other parties can verify the validity of the TPM that they are communicating with. We will examine in more detail the creation of the TPM's identities, in particular for privacy protection based on direct anonymous attestation and a privacy CA, later in this section.

Root of Trust for Measurement (RTM). Measuring the integrity of the processes in the computing platform is the first IMSR step. This is performed using the RTM. This is a computing engine that has the ability to carry out mathematical calculations or logical functions, similar to the ability of a micro-processor. On the initial start up of the computing platform, the RTM measures the processes and software on the platform. The measured values (the *integrity metrics*) demonstrate the platform's current state, which can then be used to decide whether this platform can be trusted or not.

Root of Trust for Storage (RTS). The main responsibility of the RTS is to store the *integrity metrics* measured by the RTM. These metrics are stored in a log termed the *Stored Measurement Log* (SML). The SML may be large considering the various processes that need to be measured in today's computing platforms. Hence it is not feasible to store the log in the TPM itself. Instead, it is stored in the conventional storage of a computing platform. The SML does not need to be protected as any modification of it can be detected. This is so because a summary of the integrity metrics stored by the SML is computed and used to check the validity of the SML. This summary is normally the hash value of the integrity metrics. The hash value, which can be computed based on a cryptographic hash algorithm such as SHA-2 [70], is smaller in size than the SML. So it can be stored in the limited storage of the TPM known as the *Platform Configuration Registers* (PCRs).

Root of Trust for Reporting (RTR). The main objective of the RTR is to report the integrity metrics and the authenticity of these metrics when requested by other

parties. This is performed by:

- retrieving the integrity metrics from the SML and the corresponding PCR values, and
- signing the PCR values using the TPM's identities, the *Attestation Identity Key* (AIK).

In summary, the TPM, together with the IMSR, are the fundamental components of the trusted computing platform. They allow a computing platform to validate that another computing platform can be trusted.

Direct Anonymous Attestation. Direct Anonymous Attestation (DAA) [18] is a special type of signature scheme that can be used to anonymously authenticate a trusted computing platform. Briefly, the DAA scheme has the following properties:

- It allows other parties to validate that a TPM in a client's platform is genuine, and that the TPM possesses an identity based on an AIK.
- These validations are performed without revealing the identity of the TPM to these other parties.
- The transactions involving different AIKs from an identical client's platform cannot be linked together even if these other parties collude.

Due to the three properties stated above, DAA provides full anonymity and unlinkability. The DAA scheme consists of two phases. These are the `DAA Join` phase and `DAA Sign` phase. It involves three parties. These are the DAA Issuer, with a similar role to a certificate authority who generates anonymous certificates, the TPM in C's computing platform, and the verifier (e.g. the distributor D). We describe them in the following:

`DAA Join.` The main goal of this phase is to allow the client's computing platform to obtain a DAA certificate from the DAA Issuer. The DAA certificate is used to prove that the client's computing platform can be trusted by other parties. For simplicity, we will represent the client's computing platform as the client C.

1. The first step is for a client to authenticate himself to the DAA Issuer based on the TPM's *endorsement key* EK. As mentioned in Section 7.3.1, this EK is usually generated and inserted into the TPM of the client by its manufacturer. In this case, the manufacturer acts as the CA. A certificate on the public half of the EK is also generated by the CA as $Cert_{ssk_{CA}}(EK)$. Other certificates provided by the CA to prove the authenticity of the TPM are also passed to the DAA Issuer. We denote these certificates as C's TPM *credentials* CRE_{TPM_C}.

2. Next the DAA Issuer issues a DAA certificate to the TPM. In the following we describe how the DAA certificate is issued:

   - Let (m, S, Y, R) be the public key of the DAA Issuer, in which m is an RSA modulus (e.g. $m = pq$ where p and q are distinct large primes), and $S, Y, R \in \mathbb{Z}_m$ are randomly generated.
   - The client generates a value f and computes $U = R^f S^{v'} \bmod m$, where v' is a value generated randomly to "blind" f. The value f is the secret key of the client. The client also computes $N_I = \zeta_I^f \bmod \Gamma$, where ζ_I is derived from the identity of the DAA Issuer, and Γ is a large prime. The client then sends (U, N_I) to the DAA Issuer, and convinces the DAA Issuer that U and N_I are correctly formed (using a zero-knowledge proof of knowledge of a discrete logarithm [122]).
   - If the DAA Issuer is convinced, it signs the message U by computing $G = (\frac{Y}{US^{v''}})^{1/e} \bmod n$, where v'' is a random integer and e is a random prime. The DAA Issuer then sends (G, e, v'') to the client and proves that G was computed correctly. The DAA Certificate for the TPM is (G, e, v), where $v = v' + v''$.

`DAA Sign.` The main goal of this phase is to allow the client to authenticate a message to other parties based on a signature. This is performed by using the DAA certificate obtained from the previous phase. It involves the client and one other party, which in our context will be the distributor. In the following we describe the process:

1. The client signs a message M using the secret key f, the DAA Certificate, and other required information. This message M can be an Attestation Identity

Key (AIK) generated by the TPM of the client. The client also computes $N_V = \zeta^f \bmod \Gamma$ as part of the signature computation, where ζ is an integer that affects the degree of the client's anonymity. We denote the generated signature by σ.

2. The distributor verifies the signature σ. If the signature is valid then the distributor checks that the client has a valid DAA Certificate (G, e, v) and thus has a genuine TPM embedded. This is accomplished by a proof of knowledge of a discrete logarithm [122] based on a set of values f, G, e and v such that $G^e R^f S^v \equiv Y \bmod n$. With this, the distributor is also convinced that the message M was signed by the client using the secret key f, without knowing what f is.

In summary, using the `DAA Join` and `DAA Sign` phases, the client obtains a DAA Certificate (which only needs to be performed once) and can generate and sign as many AIKs as the client wishes. The client is also able to prove to other parties that these AIKs are valid without involving the DAA Issuer.

Full anonymity and unlinkability due to DAA. The capability of generating many distinct but valid AIKs, together with the parameter ζ, provides full anonymity and unlinkability. If for each transaction with the distributor the client uses a distinct ζ, then the distributor will not be able to *link* these transactions to the same client. Similarly, if a different AIK is used by the client to communicate with different distributors then these distributors, even when they collude, will not be able to link these transactions to a single client. More importantly, the DAA certificate does not contain any information that can be linked to the EK of a particular TPM of the client. This means that no parties, not even the trusted DAA issuer, can determine the identity of the TPM that communicated with other parties. This is in contrast to other provisions of anonymity and unlinkability, such as in the LYTC protocol, where the CA has knowledge of the identity of the client.

Privacy CA. According to the definition by the Trusted Computing Group [126], a *Privacy CA* is a trusted third party (typically well known and recognised), trusted by both the owner (i.e. client C) and the verifier (i.e. distributor D), that will issue AIK certificates. Recall from our discussion of the TPM endorsement key and identities, that the AIKs are a key pair generated by a TPM as the identity of

the TPM. The role of the Privacy CA is to attest these AIKs. This means that it generates a certificate on the public half of the AIKs. However, it does not include the information on the EK in this certificate. In other words, when a client uses this certificate to communicate with the distributor, the distributor is convinced that the client is legitimate but will not be able to determine which TPM the communication is originating from. Hence, a Privacy CA plays a similar role to a certificate authority who issues anonymous certificates, such as the CA in the FCS protocol.

The difference between the Privacy CA and the DAA Issuer is that the Privacy CA knows the EK of a TPM, while the DAA Issuer does not. This means that the Privacy CA can link the EK to the AIKs and hence is able to determine the client who deploys a TPM with this EK.

7.3.2 A Protocol Based on DAA

In this section, a FaCT protocol based on the trusted computing platform proposed in [86] is discussed. This protocol also provides anonymity and unlinkability based on DAA. We denote this protocol as *the DAA protocol*.

Fundamentals. It involves five parties and the TPM. The five parties are C, D, CA, A and DAA Issuer. The CA and A are fully trusted. The DAA Issuer is expected to correctly generate AIK certificates, but is allowed to collude with other parties. The DAA protocol provides traceability, framing resistance, and full anonymity and unlinkability. With the provision of the DAA scheme, the design of the protocol is such that it is not possible for any party to identify C even when the DAA Issuer and D collude. Hence C enjoys full anonymity and unlinkability compared to other protocols where the trusted third party has the identity information of C. So for D to prove illegal distribution of content, evidence from other means outside the protocol execution is needed to identify C. Hence, it is a weak FaCT protocol as defined in Section 3.3.3.

Environment. The DAA protocol assumes that both D and C have ample computing resources due to the deployment of homomorphic encryption to perform watermarking in the encrypted domain. It also assumes a secure communication channel is in place and public key support is available. The DAA Issuer is an offline TTP since it is only required during the initial setup. The main building blocks are

digital watermarking schemes, homomorphic encryption schemes, the DAA scheme and digital signature schemes. Table 7.2 summarises the design framework of the DAA protocol.

Table 7.2: The Design Framework of the DAA Protocol

Fundamentals	
Parties Involved	C, D, CA, A, DAA Issuer
Trust Assumptions	CA and A are *fully trusted* DAA Issuer is trusted to correctly generate certificates
Security Properties	Traceability (**TR**), Framing resistance (**FR**), full anonymity and unlinkability (**AU**)
Environment	
Comp. Resources	Assume D and C have ample resources
Sec. comm. Support	Required
Pub. Key Support	Required
TTPs	Trusted Computing Platforms [126], offline DAA Issuer
Building Blocks	Digital watermarking scheme, homomorphic encryption scheme, the DAA scheme and digital signature scheme

We now describe the three phases of the protocol.

`Initial Setup.` In this phase C obtains a DAA Certificate from the DAA Issuer. This operation follows the `DAA Join` phase. We note that the `DAA Join` phase can be performed before the computing platform is given to C. If this is the case then the `DAA Join` phase is a one-off process. Figure 7.4 shows the protocol messages.

① `Initial Setup (DAA Join):`

$C \rightarrow$ DAA : $\{CRE_{TPM_C}, Cert_{ssk_{CA}}(EK)\}_{AKE}$

DAA $\rightarrow C$: $\{DAA\ Cert\}_{AKE}$

Before content distribution

Figure 7.4: DAA Protocol – Initial Setup

`Content Watermarking and Distribution.` In this phase C requests content from D and D sends a marked content to C. Through the use of TPM and IMSR, C generates the watermark and is forced to generate a well-formed one. Figure 7.5 shows the protocol messages. The protocol steps are as follows:

(I) C (or the trusted hardware) generates the watermark and the trusted hardware verifies the watermark to be well-formed.

② Content Watermarking and Distribution:		
$C \rightleftharpoons$ TPM	: SML, PCR	
$C \rightarrow D$	: $\{[W]_{HE_{hek^*}}, VAIK_C, pvk^*, hek^*, \sigma, \text{AGR}, [pvk^*]_{SIG_{SAIK_C}}, \text{SML}, [[W]_{HE_{hek^*}}, hek^*, AGR]_{SIG_{ssk^*}}, [\text{PCR}]_{SIG_{SAIK_C}}\}_{AKE}$	
$D \rightarrow C$	: $\{[X'']_{HE_{hek^*}}\}_{AKE}$	**Content distribution**

Figure 7.5: DAA Protocol – Content Watermarking and Distribution

1. C generates a watermark W based on a digital watermarking scheme (e.g. the spread spectrum watermarking scheme presented in Figure 2.1). C also generates an AIK key pair, ($VAIK_C$,$SAIK_C$).

2. C retrieves the Stored Measurement Log (SML) and the corresponding TPM's Platform Configuration Register (PCR) values. C then signs the PCR values as $[\text{PCR}]_{SIG_{SAIK_C}}$. The SML and PCR values provide the evidence that a particular watermarking scheme was used by C to generate the watermark. Also, the SML and PCR values provide the evidence that the watermark is well-formed, based on the execution of an algorithm that tests the randomness of the watermark.

(II) C requests content, provides D with the encrypted client watermark and approves a content agreement with D.

3. C initiates the content request by negotiating with D a content agreement AGR, without revealing C's real identity. This agreement contains the description of content. For example, the agreement might have been pre-set on a website hosted by D, and C just needs to download the agreement.

4. C generates an encryption key pair (hek^*, hdk^*) and a signature key pair (pvk^*, ssk^*).

5. C encrypts the watermark W as $[W]_{HE_{hek^*}}$ using a homomorphic encryption scheme and signs the encrypted watermark $[W]_{HE_{hek^*}}$, the encryption key hek^* and the agreement AGR to obtain:

$$[[W]_{HE_{hek^*}}, hek^*, AGR]_{SIG_{ssk^*}}.$$

6. C computes $\zeta = \mathrm{H}(ID_C)$, where ID_C denotes the client's identity information. C then creates a pseudonym, $N_v = \zeta^f$, where f is the secret key generated during the `DAA join` phase.

7. To prove to D that the AIK key pair ($VAIK_C$,$SAIK_C$) originates from a genuine TPM, C signs $VAIK_C$ using f, the *DAA Certificate*, and the other required information. The output is the DAA signature σ (which also includes ζ and N_v). Next, C signs pvk^* using $SAIK_C$ as $[pvk^*]_{SIG_{SAIK_C}}$ to prove that pvk^* is a valid verification key originating from the client's TPM.

8. C sends to D $[W]_{HE_{hek^*}}$, $VAIK_C$, pvk^*, hek^*, σ, AGR, $[pvk^*]_{SIG_{SAIK_C}}$, SML, $[[W]_{HE_{hek^*}}, hek^*, AGR]_{SIG_{ssk^*}}$, $[\mathrm{PCR}]_{SIG_{SAIK_C}}$.

(III) D produces a marked copy of the requested content and sends it to C.

9. D verifies the DAA signature σ. If the signature is valid then D is convinced that:

   - C is in possession of a legitimate DAA Certificate, which implies that a genuine TPM is contained in C's computing platform.
   - $VAIK_C$ was signed using C's secret key f. Even though the value of f is never revealed, D knows that the value is linked to the value in the DAA certificate.

10. D examines the integrity measurements of C's platform. This is achieved by recursively hashing the values in the SML, and then comparing them with the corresponding PCR values. If the outcome is satisfactory, D is convinced that the watermark W is well-formed.

11. D verifies $[pvk^*]_{SIG_{SAIK_C}}$ and $[[W]_{HE_{hek^*}}, hek^*, AGR]_{SIG_{ssk^*}}$. Next, D generates a watermark V and embeds V into content X to create:

$$X' \leftarrow [X, V]_{EMB_{wmk_V}}.$$

12. Similar to the watermarking in the encrypted domain process of the Memon-Wong protocol (Section 3.7), D encrypts every element of X' one-by-one using C's public encryption key hek^*. After that, D permutes every encrypted element of W, which is represented as

$$q\left([W]_{HE_{hek^*}}\right) = ([w_{q(1)}]_{HE_{hek^*}}, [w_{q(2)}]_{HE_{hek^*}}, \ldots, [w_{q(n)}]_{HE_{hek^*}}).$$

Next, D generates an encrypted marked content as follows, using the watermarking in the encrypted domain scheme (Section 2.3.3):

$$\left.\begin{array}{rl} & [x'_i]_{HE_{hek^*}} \cdot [q(w_i)]_{HE_{hek^*}} \\ = & [x'_i \circ q(w_i)]_{HE_{hek^*}} \\ = & [x''_i]_{HE_{hek^*}} \end{array}\right\} \; 1 \leq i \leq n,$$

where $\circ$ represents either modular addition, modular multiplication or bit-wise XOR depending on the underlying homomorphic encryption used. We denote the resulting encrypted marked content by:

$$[X'']_{HE_{hek^*}} = ([x''_1]_{HE_{hek^*}}, [x''_2]_{HE_{hek^*}}, \ldots, [x''_n]_{HE_{hek^*}}).$$

The encrypted marked content $[X'']_{HE_{hek^*}}$ is sent to C.

13. When C receives the encrypted marked content $[X'']_{HE_{hek^*}}$, he decrypts it using hdk^* to retrieve the marked content $X'' = X' \circ q(W)$.

`Identification and Dispute Resolution.` In this phase D identifies C who illegally distributed content, based on a found copy of content and the watermark embedded in it.

③	`Identification and Dispute Resolution:`	
D	: $\{\texttt{true}, \texttt{false}\} \leftarrow [\widehat{X}, V, X]_{DET_{wmk}}$	
$D \rightarrow A$	: $\{[W]_{HE_{hek^*}}, VAIK_C, pvk^*, hek^*, \sigma, \text{AGR}, [pvk^*]_{SIG_{SAIK_C}}, \text{SML}, [[W]_{HE_{hek^*}}, hek^*, AGR]_{SIG_{ssk^*}}, [\text{PCR}]_{SIG_{SAIK_C}}, X', \widehat{X}, q\}_{AKE}$	
A	: *collates other evidence to identify C*	
$A \rightarrow C$	: $\{watermark\ info?\}_{AKE}$	**After content distribution**
$C \rightarrow A$	: $\{watermark\ info\}_{AKE}$	
A	: $\texttt{true} \leftarrow [\widehat{X}, q(W), X']_{DET_{wmk}}$	

Figure 7.6: DAA Protocol – Identification and Dispute Resolution

(I) D detects a watermark from the found copy of content in order to identify the client that owns the content.

1. When an illegal copy $\widehat{X}$ is found, D detects the presence of V in $\widehat{X}$:

$$\{\texttt{true}, \texttt{false}\} \leftarrow [\widehat{X}, V, X]_{DET_{wmk}}.$$

If the detection algorithm returns `true`, then detection of V is successful and D can blacklist the pseudonyms, that is, the N_v values used by the client. It is not possible to blacklist the client since D has no knowledge of the real identity of C.

(II) D proves to A that C illegally distributed copies of content, with evidence from other sources.

2. Firstly, C cannot dispute the blacklisting by D if the reason is that D found an illegally distributed content attributed to C. This is because only C possesses the marked copy of content.

3. Secondly, to prove to A that C has illegally distributed content, D sends the pseudonyms N_v and marked content to A. However, since no party can determine the real identity of C except for C himself, due to the `initial setup` phase using the `DAA Join`, other means of identifying the perpetrator, such as disclosure of network activities by the Internet Service Provider (ISP) are required.

Therefore, the dispute resolution phase is different from other protocols in that D cannot prove to A the illegal act of C based on the information D has and the assistance of the trusted third party (i.e. DAA Issuer). The DAA Issuer will not be able to identify the client, since the DAA Issuer cannot determine the client. This is due to the *full anonymity and unlinkability* provided by the DAA scheme, as discussed earlier in the `DAA Join` phase.

In summary, the main idea of the protocol is:

- to deploy a trusted computing platform that consists of a TPM and IMSR, which allows the client to generate the watermark, but at the same time forces the client to generate a well-formed one based on the integrity measurement of the IMSR.

- to give full privacy protection (*anonymity and unlinkability*) to the client based on DAA, such that no parties will be able to determine the real identity of the client and link any of the content requests.

However, the distributor might want to know the real identity of a client who illegally redistributed copies of content. This is performed directly in other protocols by trusted third parties (i.e. the CA or WCA) who have such information. In the next section we present a protocol based on a trusted computing platform that allows the distributor to determine the real identity of the client, while providing anonymity and unlinkability.

7.3.3 A Protocol Based on a Privacy CA

This is a protocol that is based on a TPM and replaces the DAA Issuer with a Privacy CA. As opposed to the DAA protocol, the Privacy CA is tasked with holding the client's identity and is fully trusted. We denote this protocol as *the PCA protocol*.

Fundamentals. As shown in Table 7.3, the PCA protocol involves five parties, with the Privacy CA replacing the DAA Issuer. The CA, A, and Privacy CA are fully trusted. The protocol provides traceability, framing resistance, non-repudiation of redistribution and anonymity and unlinkability.

Environment. With similar reason to the DAA protocol, we assume in this protocol that both C and D have ample resources. We also assume that public key and secure communication support is in place. The Privacy CA is an offline TTP because it is only involved in the initial setup phase and when there is a dispute between C and D. The main building blocks are digital watermarking schemes, homomorphic encryption schemes and digital signature schemes. In the following we describe the three phases of the protocol.

`Initial Setup.` In this phase, C generates an AIK key pair $(VAIK_C, SAIK_C)$ and requests the Privacy CA to certify $VAIK_C$, the public half of the AIK. It is assumed that the Privacy CA knows the *endorsement key* EK of the TPM in the client's computing platform. Figure 7.7 shows the protocol messages and the following describes the protocol steps:

(I) C requests the Privacy CA to certify C's AIK.

1. C chooses an identity ID_C.
2. C signs $VAIK_C$ and the identity ID_C as $[VAIK_C, ID_C]_{SIG_{SAIK_C}}$. This signature

Table 7.3: The Design Framework of the PCA Protocol

Fundamentals	
Parties Involved	C, D, A, CA, Privacy CA
Trust Assumptions	CA, Privacy CA and A are *fully trusted*
Security Properties	Traceability (**TR**), Framing resistance (**FR**) Non-repudiation of redistribution (**NR**) Anonymity and unlinkability (**AU**)
Environment	
Comp. Resources	Assume D and C have ample resources
Sec. comm. Support	Required
Pub. Key Support	Required
TTPs	Trusted Computing Platforms [126], offline Privacy CA
Building Blocks	Digital watermarking scheme, homomorphic encryption scheme and digital signature scheme

① `Initial Setup:`

$C \rightarrow$ P CA : $\{VAIK_C, ID_C, [VAIK_C, ID_C]_{SIG_{SAIK_C}}$,
$CRE_{TPM_C}, Cert_{ssk_{CA}}(EK)\}_{AKE}$

P CA $\rightarrow C$: $\{Cert_{ID_C}\}_{AKE}$

Before content distribution

Figure 7.7: Privacy CA Protocol – Initial Setup

is then sent, together with the various certificates provided by the CA (e.g. the manufacturer), to the Privacy CA. We denote these certificates as C's TPM *credentials* CRE_{TPM_C}. These credentials are used to convince the Privacy CA that the TPM is genuine.

(II) Privacy CA provides C a certified AIK.

3. The Privacy CA verifies the signature and C's TPM credentials.

4. The Privacy CA creates an identity certificate that normally contains the $VAIK_C$ and encrypts the identity certificate with C's TPM public endorsement key EK. This is to ensure that only the genuine C's TPM will be able to decrypt and use the identity certificate. We denote the identity certificate by $Cert_{ID_C}$. (In practical terms, the certificate can be encrypted using a symmetric encryption scheme, and the symmetric key and a hash value of $VAIK_C$ are encrypted using the public half of EK).

5. Based on $Cert_{ID_C}$, C can generate other signing and encryption key pairs and certifies these key pairs with the certified AIK. If C wishes different content requests to be unlinkable, then C will request from the Privacy CA a distinct certified AIK for each different content request.

Similar to the DAA Issuer, the Privacy CA's only task is to register the client and distributor (similar to the role of a normal CA), except that the Privacy CA knows the client's real identity. Since C can use the certified AIK key pair to generate and certify many new randomly generated key pairs, if C does not mind his patterns of content requests being linked by the distributor (i.e. unlinkability), then the initial setup is a one-time process.

`Content Watermarking and Distribution.` The main goal of this phase is to provide marked content to C, while at the same time ensuring that neither the distributor nor the client will be discriminated due to tracing of content. The main idea is to let C generate the watermark. The TPM and the IMSR in C's computing platform will then check that the generated watermark is well-formed. With the certified pseudonym key pair $(VAIK_C, SAIK_C)$ provided by the Privacy CA, C can generate a signing key pair and an encryption key pair to proceed with the protocol. All the steps are identical to the `Content Watermarking and Distribution` phase of the DAA protocol. The only difference is the replacement of the *DAA Certificate* with the *identity certificate* $Cert_{ID_C}$ provided by the Privacy CA. Therefore, we will not describe the steps here. The protocol messages are shown in Figure 7.8.

② `Content Watermarking and Distribution:`		
$C \rightleftharpoons$ TPM	: SML, PCR	
$C \rightarrow D$	: $\{[W]_{HE_{hek^*}}, VAIK_C, pvk^*, hek^*, Cert_{ID_C}, [pvk^*]_{SIG_{SAIK_C}}$, AGR, SML, $[[W]_{HE_{hek^*}}, hek^*, AGR]_{SIG_{ssk^*}}$, $[\text{PCR}]_{SIG_{SAIK_C}}\}_{AKE}$	
$D \rightarrow C$	: $\{[X'']_{HE_{hek^*}}\}_{AKE}$	**Content distribution**

Figure 7.8: Privacy CA Protocol – Content Watermarking and Distribution

`Identification and Dispute Resolution.` Just as in the protocol based on DAA discussed in the previous section, the identification of a particular found content can be done by detecting the watermark V. The difference from the previous protocol is that D and A can request the Privacy CA to reveal the real identity of C. Figure 7.9 shows the protocol messages and in the following we present the communication

between D, A and the Privacy CA:

③ Identification and Dispute Resolution:		
D	: $\{\texttt{true}, \texttt{false}\} \leftarrow [\widehat{X}, V, X]_{DET_{wmk}}$	
$D \rightarrow A$	: $\{[W]_{HE_{hek^*}}, VAIK_C, pvk^*, hek^*, Cert_{ID_C}, [pvk^*]_{SIG_{SAIK_C}}$, AGR, SML, $[[W]_{HE_{hek^*}}, hek^*, AGR]_{SIG_{ssk^*}}$, $[\text{PCR}]_{SIG_{SAIK_C}}$, $X', \widehat{X}, q\}_{AKE}$	
$A \rightarrow$ P CA	: $\{Cert_{ID_C}\}_{AKE}$	
P CA $\rightarrow A$	: $\{C\text{'s identity}\}_{AKE}$	
$A \rightarrow C$	: $\{\textit{watermark info?}\}_{AKE}$	
$C \rightarrow A$	: $\{\textit{watermark info}\}_{AKE}$	**After content distribution**
A	: $\texttt{true} \leftarrow [\widehat{X}, q(W), X']_{DET_{wmk}}$	

Figure 7.9: Privacy CA Protocol – Identification and Dispute Resolution

(I) D detects a watermark from the found copy of content in order to identify the client that owns the content.

1. When an illegal copy of content $\widehat{X}$ is found, D detects watermark V from this copy. If the watermark is detected, D proceeds to communicate with A to prove illegal distribution by C.

(II) D proves to A that C illegally distributed copies of content.

2. D sends $[W]_{HE_{hek^*}}$, $VAIK_C$, pvk^*, hek^*, $Cert_{ID_C}$, $[pvk^*]_{SIG_{SAIK_C}}$, AGR,SML, $[[W]_{HE_{hek^*}}, hek^*, AGR]_{SIG_{ssk^*}}$, $[\text{PCR}]_{SIG_{SAIK_C}}$, X', $\widehat{X}$ and q to A.
3. A then matches C's identity certificate $Cert_{ID_C}$ with the records stored by the Privacy CA, which includes the public endorsement key EK of C's TPM. By knowing C's identity, A ascertains whether C illegally redistributed content by checking the signature $[[W]_{HE_{hek^*}}, hek^*, AGR]_{SIG_{ssk^*}}$ and matching the watermark W retrieved from the found copy to the original watermark.

In summary, the PCA protocol is beneficial for the distributor compared to the DAA protocol, since it allows the distributor to determine the identity of a client with the help of the Privacy CA. At the same time, although there is not full anonymity and unlinkability, as long as the Privacy CA is fully trusted, a client can be assured that he can communicate with the distributor anonymously. Hence, depending on the application scenarios, if full anonymity is required then the DAA protocol can

be used, while if a trusted third party that holds client's information is required, then the PCA protocol can be deployed.

7.4 Analysis

In this section we analyse the protocols with trusted hardware that we have just discussed.

7.4.1 Security

Traceability. In the FCS protocol, traceability is provided through the watermark V and watermark W. When an illegal copy of content is found, D traces the client based on the detection of watermark V. A can be convinced that an illegal copy belongs to C based on the detection of watermark W.

Similar to the FCS protocol, traceability in the PCA protocol is provided through the watermark V and watermark W. Traceability in the DAA protocol is provided through the watermark V. However, due to the property of full anonymity and unlinkability, D can only trace to the pseudonyms instead of the client's real identity.

Framing Resistance. The FCS protocol provides this property since D has no knowledge of the watermark W, as W is produced and encrypted by the trusted hardware T_D. Also, the watermark W is embedded into content without D being able to determine the watermark. Therefore, it is not possible for D to frame C. For the case of embedding a watermark W retrieved from a found copy into a more valuable content, A will be able to spot this discrepancy. This is done by checking C's signature on the agreement $[pvk^*, hek^*, AGR]_{SIG_{ssk^*}}$ and the WCA's signature $[[W]_{HE_{hek^*}}, pvk^*, hek^*, [W]_{HE_{hek}{}_{WCA}}]_{SIG_{ssk}{}_{WCA}}$ generated by the trusted hardware T_D in Step 2 and Step 4 in the `Content Watermarking and Distribution` phase.

Similar to the FCS protocol, in the DAA and the PCA protocols, framing of C by D is not possible since D has no knowledge of the embedded watermark W in the final copy possessed by C. This can be observed from the `Content Watermarking and Distribution` phase, in which W is embedded into content in the encrypted form. Also, transplanting the watermark W retrieved from an illegal copy to

another more valuable copy to frame C can be spotted based on the signature $[[W]_{HE_{hek^*}}, hek^*, AGR]_{SIG_{ssk^*}}$, which links the agreement AGR with other information, as provided in Step 5 of the `Content Watermarking and Distribution` phase in Section 7.3.2.

Non-repudiation of Redistribution. This property is provided in the FCS protocol through C's signature on the agreement $[pvk^*, hek^*, AGR]_{SIG_{ssk^*}}$ and signature $[[W]_{HE_{hek^*}}, pvk^*, hek^*, [W]_{HE_{hek_{WCA}}}]_{SIG_{ssk_{WCA}}}$ generated by the trusted hardware T_D. Based on these signatures, the watermark W retrieved from the found copy and the description of the agreement AGR, A can confirm that C requested and owned the copy of content found by D. This is performed by first matching the retrieved watermark W with the original generated by T_D and then verifying the two signatures. After that, the identity of the client can be determined with the help of the CA.

Similarly, the PCA protocol provides non-repudiation of redistribution through signature $[[W]_{HE_{hek^*}}, hek^*, AGR]_{SIG_{ssk^*}}$ generated by C and the detection of watermark W. Since the watermark W is known only to C, A obtains the identity of C from the Privacy CA and proceeds to request C to reveal the watermark.

As for the DAA protocol, since the main aim is to provide full anonymity and unlinkability in such a way that no parties know the client's identity except for the client himself, non-repudiation of redistribution as provided in the previous two protocols with the help of a trusted third party (e.g. the CA, Privacy CA) is not possible. Therefore, to prove that C has illegally distributed content based on the signature $[[W]_{HE_{hek^*}}, hek^*, AGR]_{SIG_{ssk^*}}$ and the retrieved watermark W, other evidence to determine the identity of C is required, such as through collections of network activities from the Internet Service Provider. Thus the DAA protocol is a weak FaCT protocol.

Anonymity and Unlinkability. All three protocols provide this property. For the FCS and the PCA protocol, this is provided through pseudonyms certified by the CA or Privacy CA. In the FCS protocol, these pseudonyms are (pvk^*_C, ssk^*_C), and (hek^*_C, hdk^*_C) and the randomly generated one-time key pairs (pvk^*, ssk^*) and (hek^*, hdk^*). In the PCA protocol, these are ($VAIK_C$,$SAIK_C$), and the signing and encryption key pairs (pvk^*, ssk^*) and (hek^*, hdk^*). D cannot determine the identity of C based solely on these pseudonyms and the randomly generated one-time keys,

as there is no identity information attached to them. If C wishes that each content request is unique and cannot be linked, then C generates unique (pvk^*_C, ssk^*_C) and (hek^*_C, hdk^*_C) or ($VAIK_C$,$SAIK_C$) and asks the CA or Privacy CA to certify these pseudonyms. So D will not be able to link the many content requests to C. However, the CA and Privacy CA must be fully trusted since these parties know the identity of C.

This brings us to the *full* anonymity and unlinkability provided by the DAA protocol. In this protocol, C interacts with D using $AIKs$, which act as pseudonyms (just as in the PCA protocol). For unlinkability, C uses different AIK keys and N_v values to communicate with D. The difference between this protocol and the previous ones is that even though the DAA Issuer knows which TPMs with EKs possess which valid DAA Certificates, the DAA Issuer cannot link these EKs with the corresponding $AIKs$. This is because, in order to be able to make this link, the DAA Issuer would require knowledge of the TPM's DAA secret value, f. This is computationally infeasible because of the way that a DAA Certificate is created, and since f never leaves the TPM. Hence the DAA Issuer cannot determine the real identity of C when presented with the $AIKs$.

Summary. Assuming that the underlying building blocks are secure, all three protocols provide traceability and framing resistance. The FCS and the PCA protocols also provide non-repudiation of redistribution. The DAA protocol, however, does not provide non-repudiation of redistribution directly. In addition, all protocols provide the additional property of anonymity and unlinkability. The DAA protocol further provides *full* anonymity and unlinkability, which means that not even the trusted third party DAA Issuer is able to determine the identity of C.

We summarise the security analysis of these protocols in Table 7.4.

Table 7.4: Summary of the Security Analysis

Protocols	**TR**	**FR**	**NR**	**AU**	**Conditions**
FCS	✓	✓	✓	✓	Abstract trusted hardware.
DAA	✓	✓	×	✓[1]	Security of TPM and DAA. A weak FaCT protocol.
P CA	✓	✓	✓	✓	Security of TPM.

[1] *full anonymity and unlinkability*

7.4.2 Efficiency

We now examine the performance of the three protocols, the results of which are summarised in Table 7.5. We assume that all three protocols deploy the homomorphic encryption scheme of Paillier [99]. In addition, since the DAA and the PCA protocols have similar processes with regards to their bandwidth, computation and storage requirements, we will treat them as one entity.

Bandwidth. All three protocols produce the encrypted marked content based on a homomorphic encryption scheme with modulus m. Given that there are n elements in the watermark and content, the size of the encrypted marked content transmitted between D and C is $n|m|$.

Trusted Third Parties. All three protocols require special trusted hardware for producing and authenticating the watermark. In the FCS protocol, this trusted hardware resides in D's computing platform, while in the DAA/PCA protocols, the trusted hardware resides in C's computing platform. As there is no explanation on how to realise the trusted hardware, we reason that the implementation cost for the trusted hardware in the FCS protocol is higher since it is assumed that the trusted hardware performs watermarking in the encrypted domain and digital signatures. In contrast, in the DAA/PCA protocols, these computations are performed by C's computing platform, while the TPM only measures and ensures the integrity of these processes. Also, since these protocols provide anonymity and unlinkability, the CA, DAA Issuer or Privacy CA, in addition to the common responsibility of public key support, needs to provide pseudonyms (anonymous keys) for C.

Computation. Similar to the LYTC protocol presented in Section 5.2, in all three protocols, D needs to encrypt n elements of content and multiplies them with the n encrypted elements of the watermark. Therefore, D performs n modular exponentiations ($n\mathbf{E}$) and n modular multiplication ($n\mathbf{M}$). D also adds a watermark V into the content. This amounts to n additions ($n\mathbf{A}$). As for C, in the FCS protocol, only n modular exponentiations ($n\mathbf{E}$) to decrypt the encrypted marked content is required, while in the DAA/PCA protocols, an extra n modular exponentiations ($n\mathbf{E}$) is required during the encryption of the watermark W.

Storage. In all three protocols, for producing the encrypted marked content, D needs to store the homomorphic encryption key of C. This has size $|m|$ based on the

modulus m of the underlying homomorphic encryption scheme. D also stores the encrypted watermark $[W]_{HE_{hek^*}}$ and the watermark V. The encrypted watermark has size $n|m|$, since there are n encrypted elements and each of them has $|m|$ bits. The watermark V has size $n|Z|$, where Z denotes the highest possible value for each element. C, on the other hand, needs to store the public and private encryption keys for the encryption and decryption of the marked content. This amounts to $2|m|$ bits of storage. The DAA/PCA protocols further require the storage of the watermark W, since this is generated and only known by C. This amounts to $n|Z|$ bits.

Summary. From the above analysis, we observe that all three protocols have similar efficiency, except that C in the FCS protocol requires less computation and storage. However, the implementation cost of the trusted hardware in the FCS protocol is higher.

Table 7.5: Efficiency Comparisons between Protocols with Trusted Hardware

Pro.	*Bandwidth*	*TTP*	*Computation*[1]	*Storage*[2]
FCS	$[X'']_{HE_{hek^*}} = n\|m\|$	D's TH[3]	C: $n\mathbf{E}$ D: $n(\mathbf{E}+\mathbf{M}+\mathbf{A})$	C: $2\|m\|$ D: $(n+1)\|m\| + n\|Z\|$
DAA/ PCA	$[X'']_{HE_{hek^*}} = n\|m\|$	C's TH	C: $2n\mathbf{E}$ D: $n(\mathbf{E}+\mathbf{M}+\mathbf{A})$	C: $2\|m\| + n\|Z\|$ D: $(n+1)\|m\| + n\|Z\|$

[1] $\mathbf{E}$=$O(k^3)$, $\mathbf{M}$=$O(k^2)$, $\mathbf{A}$=$O(k)$
[2] $|Z| < |m|$
[3] Higher implementation cost compared to the DAA and PCA protocols

7.5 Summary

In this chapter we discussed three FaCT protocols that deploy trusted hardware. All three protocols provide anonymity and unlinkability as an additional property. The FCS protocol assumes the existence of trusted hardware that resides in D's computing platform. The trusted hardware generates and encrypts the watermark W. The proposal does not explain how such trusted hardware can be realised.

Two other protocols, which we proposed, adopt the TPM hardware module. The first protocol, which was published in [86], uses DAA to provide full anonymity and unlinkability to the client, which means that no parties (not even the third party known as the DAA Issuer) knows the real identity of a client. Since this might not be desirable for D, an alternative approach, based on a trusted third party known as a Privacy CA, can be used. This is demonstrated in our second protocol.

Chapter 8

FaCT Protocols with Payment and Fair Exchange

Contents

In this chapter we investigate FaCT protocols with payment and fair exchange. The addition of payment, which in turns motivates the requirement of fair exchange, has not been discussed before in the context of fair content tracing. We discuss why this is an important issue and examine how payment and fair exchange can be added into the main categories discussed in previous chapters. We further provide an example protocol.

8.1 Overview

Many existing FaCT protocols assume that the content involved is being exchanged for monetary payment, without demonstrating how payment can be included. Such addition of payment was briefly discussed in Section 3.5.6. While the addition seems straightforward, except for the requirement of a payment agent (see Figure 3.11), it brings out the issue of how content can be traded *fairly*. This means that D is assured of receiving the correct payment while C receives correct content. This issue is relevant since D and C do not necessarily trust one another, which is one of the main assumptions behind the design of FaCT protocols.

Therefore, the main goal of this chapter is to investigate the addition of payment and how C and D can trade content fairly. Fair trading of this type is known as *fair exchange* [5, 92]. We discuss the required changes to the four categories of FaCT protocols when payment and fair exchange are added. We also construct a fair exchange FaCT protocol with online trusted third parties as an example.

8.2 Adding Payment and Fair Exchange

In order for a FaCT protocol to include payment, parties known as *payment agents* must be involved. While to ensure that C and D trade fairly, we examine and use the existing *fair exchange protocols*.

Payment Agents PA. As discussed in Section 3.5.6, a client C and a distributor D enter into contractual relationships with their respective banks and agree on a payment mechanism. Thus, when C buys content from D and provides D with the payment information based on the agreed mechanism, D forwards this information to his bank (or a payment gateway such as paypal [101], VISA [131] or Mastercard [93]) to obtain the correct payment. We denote the bank or the payment gateway as the *payment agent* PA.

In a FaCT protocol, this means that when C buys content from D during the `Content Watermarking and Distribution` phase, D needs to contact the PA to verify the payment. As a result, both C and D will be provided with information from the PA showing that payment has been processed.

Fair Exchange (FE) Protocols. These are protocols commonly applied to electronic payment and certified mail exchange for fair trading between two parties. As observed in [5], a straightforward method of ensuring fair exchange is to deploy a trusted third party (i.e. the arbiter A). For example, D gives the marked content to A and C provides his payment information to A. After A is satisfied that both the received objects are correct, he forwards the marked content to C and process the payment for D. By exchanging objects through A it is ensured that no parties are discriminated. The main drawback of this approach is that A is always involved in the process, even when C and D are honest. Nevertheless, it has been proved by Even and Yacobi in [45] that fair exchange is impossible without any involvement of a trusted third party.

Thus the main goal of a FE protocol is to reduce the reliance on this trusted third party. For example, Franklin and Reiter [49] proposed a FE protocol with a semi-trusted third party, in which this third party may misbehave on its own but will not collude with other parties. Asokan *et al.* [5] proposed an *optimistic* FE protocol, where optimistic means that a trusted third party is involved only when there is a dispute between the two communicating parties, but not if the protocol is run honestly.

More recently, Ray *et al.* [116] and, Zhang and Markantonakis [140] proposed fair e-payment protocols that also protect client privacy. The proposal in [116] is an optimistic FE protocol following Asokan *et al.*'s approach. Protection of client privacy is provided based on the anonymous cash system of Chaum [20]. The proposal in [140] focuses on purchase of physical goods. Fair payment is assured by C's and D's banks, who have the capability of releasing payment to D if C refuses to do so after retrieving the goods from a storage facility. Similarly, the protocol is designed in such a way that D can only receive the payment after C retrieves the goods. Protection of client privacy is provided based on anonymous credit card protocols [89]. Pagnia *et al.* [98], on the other hand, proposed a modular approach that summarised various frameworks of fair exchange, while Markowitch *et al.* [92] investigated various properties of fairness.

Following the definition provided by Asokan *et al.* [5], fair exchange can further be divided into *strong fairness* and *weak fairness*. A FE protocol achieves *strong fairness* for a party $\mathcal{I}_1$ if $\mathcal{I}_1$ never releases the object he promised to unless the

object he required has been received. On the other hand, a FE protocol achieves *weak fairness* for a party $\mathcal{I}_2$ if whenever $\mathcal{I}_2$ releases the object he promised to but never receives the object required, then there exists a proof that $\mathcal{I}_2$ can provide to A that either forces the release of the object or leads to the recovery of any losses. We note that these are simplified versions of the original definitions, which are sufficient to cover the required scope of fair exchange in the context of FaCT protocols.

We choose to design FaCT protocols with payment and fair exchange such that strong fairness is provided to D and weak fairness is provided to C. The design also follows that of Asokan *et al.*'s optimistic protocol, in which A is involved only if there is a dispute between C and D. Such a choice reflects the actual scenario of buying and selling of goods using the Internet. For example, it is always the case that the buyer provides the payment information to the seller, and the seller checks the validity of this information with the PA, before the goods are sent to the buyer. If there is any discrepancy, such as the goods received by the buyer are faulty, then it is possible for the buyer to contact the seller (or a consumer advice organisation) to resolve the issue.

In the following sections we examine the required changes in each of the main categories when payment and fair exchange are added. We assume that both C and D have entered into contractual relationships with their respective banks and have previously agreed on a payment mechanism.

8.2.1 Protocols without Trusted Third Parties

In this category, the addition of payment requires the involvement of a PA that must be online during the content distribution phase. This is so that D can verify the payment, while C receives a receipt noting that a payment had been made to D.

Following from the general construction described in Section 3.5.6, we illustrate the protocol messages in Figure 8.1. As can be observed, C includes the payment information PAY in the message sent to D. Upon receiving the message, D sends PAY to PA so that PA can verify the validity of the payment. If this is the case, and D receives the payment, then D proceeds to generate marked content and sends the resulting content to C. The PA, after paying D, also produces a payment receipt and sends it to C. This receipt serves as evidence that C paid for content and is

important to ensure weak fairness for C, which we next discuss.

② Content Watermarking and Distribution:		
$C \rightarrow D$	: $\{PAY,\ info,\ f(W),\ \texttt{SIG}\}_{AKE}$	
$D \rightarrow \text{PA}$	: $\{PAY, info\}_{AKE}$	**content distribution**
$\text{PA} \rightarrow D$	: $\{Paid, info\}_{AKE}$	
$\text{PA} \rightarrow C$	: $\{receipt, info\}_{AKE}$	
$D \rightarrow C$	: $\left\{info,\ f(\tilde{X})\right\}_{AKE}$	

Figure 8.1: Adding PA and FE: Protocols without TTPs

D is assured of strong fairness since he verifies PAY before sending C the content. To provide weak fairness for C, a new phase is required. This is the `Dispute Resolution for Fair Exchange` phase. The general construction was previously shown in Figure 3.12. We note that this phase is identical across all categories of FaCT protocols. In the following we demonstrate an example construction with payment and fair exchange based on the Semi-Fair protocol, which was discussed in Section 4.4.

Adding Payment and Fair Exchange to the Semi-Fair Protocol. In order to add payment and provide fair exchange, new protocol messages are added to the `Content Watermarking and Distribution` phase. As shown in Figure 8.2, in the first message sent from C to D, the payment information PAY and the identity of the PA ID_{PA} are included.

② Content Watermarking and Distribution:		
$C \rightarrow D$	: $\{PAY, ID_{PA}, hek_C, pvk_C, [hek_C, pvk_C, ID_C]_{SIG_{ssk_{KC}}}, AGR,$ $[W]_{HE_{hek_C}}, [PAY, ID_{PA}]_{SIG_{ssk_C}}, [[W]_{HE_{hek_C}}, AGR]_{SIG_{ssk_C}}\}_{AKE}$	
$D \rightarrow \text{PA}$	: $\{PAY, [PAY, ID_{PA}]_{SIG_{ssk_C}},$ $hek_C, pvk_C, [hek_C, pvk_C, ID_C]_{SIG_{ssk_{KC}}}\}_{AKE}$	**Content distribution**
$\text{PA} \rightarrow D$	: $\left\{Paid, [Paid]_{SIG_{ssk_{PA}}}\right\}_{AKE}$	
$\text{PA} \rightarrow C$	: $\left\{Receipt, [Receipt]_{SIG_{ssk_{PA}}}\right\}_{AKE}$	
$D \rightarrow C$	: $\left\{[X'']_{HE_{hek_C}}, [[X'']_{HE_{hek_C}}]_{SIG_{ssk_D}}\right\}_{AKE}$	

Figure 8.2: Adding PA and FE: The Semi-Fair Protocol

Instead of producing the marked content as in the original protocol, upon receiving C's message, D sends PAY to PA so that PA can verify the validity of the payment information and pay the exact amount to D. Only after D receives the payment

does D generate and send the marked content to C. Thus D is provided with strong fairness. Also, when the payment is made to D, the PA sends C a receipt. In practical terms, this can be a bank statement stating that C has paid D a certain amount of money.

When there is a dispute in which C claims that he did not receive the said content or that the content is not correct, and C is not able to resolve the dispute with D, then C initiates the `Dispute Resolution for Fair Exchange` phase (Figure 8.3).

④ `Dispute Resolution for Fair Exchange:`

$C \rightarrow A$: $\left\{ Receipt, [Receipt]_{SIG_{ssk_{PA}}} \right\}_{AKE}$

$A \rightarrow D$: $\{resend\}_{AKE}$

$D \rightarrow A$: $\left\{ [X'']_{HE_{hek_C}} \right\}_{AKE}$

$A \rightarrow C$: $\left\{ [X'']_{HE_{hek_C}} \right\}_{AKE}$

Figure 8.3: Dispute Resolution for FE: The Semi-Fair Protocol

C can request new content by sending the receipt as proof to A. If the receipt is valid, A obtains a new copy from D and forwards this copy to C. It is possible that C has already received the correct content but uses the receipt to obtain a new copy. We remark that we do not see this as a threat since C can produce as many copies of content as he likes with the content already owns. There is no motivation for C to want to go through the process of requesting new content by contacting A.

8.2.2 Protocols with Online Trusted Third Parties

Similarly to the previous category, adding payment requires the PA to be online during content distribution. Since in the original design a WCA must be online to generate client watermarks, the PA and the WCA may be combined as one entity (although in logical terms they play different roles). If this is the case then adding payment does not require additional communication between D and the trusted third party, but an extra message where C receives a payment receipt from the PA. This is shown in the general construction in Figure 8.4. Provision of fair exchange is also based on the payment receipt and is identical to Figure 8.3. An example protocol with payment and fair exchange is provided in Section 8.3.

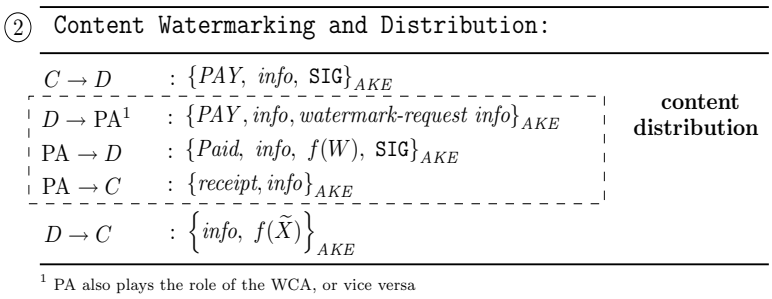

Figure 8.4: Adding PA and FE: Protocols with Online TTPs

8.2.3 Protocols with Offline Trusted Third Parties

The addition of payment and provision of fair exchange in this category is identical to that of protocols without trusted third parties (see Figures 8.1 and 8.3). This is because only the `Content Watermarking and Distribution` phase needs to be modified to include payment, and both these categories have an identical content distribution phase, except for the generation of client watermarks, which does not affect the addition of payment and fair exchange.

8.2.4 Protocols with Trusted Hardware

The main design for protocols in this category is to use trusted hardware in replacement of the online (or offline) trusted third party that is responsible for generating client watermarks. Also, if clients generate the watermarks, then the trusted hardware can be used to validate these watermarks. Thus by using trusted hardware, the issue of a central trusted third party can be alleviated through embedding their functionality into the clients' computing platforms. We have seen three protocols of this type in Chapter 7.

However, in order to add payment, the PA must be involved in the content distribution phase, as it seems impossible to have the entire payment mechanism embedded in trusted hardware or processed by the trusted hardware. Hence, the PA becomes a central point of communication during content distribution, which may be undesirable in a distributed computing environment.

In Figure 8.5 we show the protocol messages between D, C and the PA in the `Content Watermarking and Distribution` phase. As can be observed, these are

similar to other categories of protocols since the addition of payment requires the same communication flows between D, C and the PA. We will examine the adding of payment and fair exchange in the DAA protocol in the next section, where we also discuss the effects of adding them to FaCT protocols with anonymity and unlinkability.

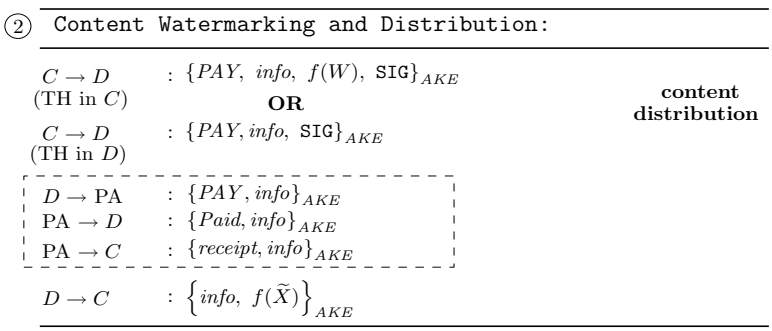

Figure 8.5: Adding PA and FE: Protocols with TH

8.2.5 Protocols with Anonymity and Unlinkability

There are two scenarios for including payment in a FaCT protocol that additionally provides anonymity and unlinkability:

- *The PA is allowed to know the identity of C.* This is similar to the CA in the original design of FaCT protocols, such as the LYTC protocol discussed in Section 5.2. If this is the case then adding payment and providing fair exchange can be done in a similar way to the other categories, except that the payment details contained in PAY are encrypted. This is required since D should not be able to determine C based on C's account information contained in PAY. In other words, C encrypts the payment information using PA's public encryption key so that only the PA can decrypt it and processes the payment.

- *The PA is not allowed to know the identity of C.* If this is the case then anonymous payment mechanisms must be used. An example of using such a mechanism for fair exchange is provided in [116], where digital coins are used. Before purchasing content from D, C requests digital coins from the PA. Due to the underlying digital cash mechanism, C obtains the digital coins without the PA being able to link these coins to C (see [116]). This is akin to

withdrawing cash from an ATM machine. Then C uses these coins as PAY to buy content from D.

However, since the PA has no way of determining C, it is not possible for the PA to provide a payment receipt to C. In such a case C relies on D to provide a receipt, or the PA stores a receipt linked to the digital coins and other information, such as a signature generated using C's one-time signing key, received from D. Thus ensuring weak fairness for C in this scenario depends on D willingly providing a payment receipt to C. We suggest that D would normally provide such a receipt since C can choose to purchase content from other distributors if D refuses to do so. If in the worst case scenario C does not receive content and the receipt after paying, then C needs to show to A other forms of evidence, and A contacts PA to verify whether such a transaction was performed.

In the following we provide an example construction of the second scenario using the DAA protocol discussed in Section 7.3.2.

Adding Payment and Fair Exchange to the DAA Protocol. As can be observed from Figure 8.6, the first message sent by C to D contains payment information, the encrypted watermarks and the information on the trusted hardware. As opposed to the original DAA protocol, where upon verifying the validity of this message D produces the marked content and sends it to C, first D passes PAY to the PA for validation. Only after the PA verifies PAY does D generate the marked content. After that, D sends the encrypted marked content, together with a payment receipt, to C. This payment receipt is a signature $[[ID_{PA}, AGR]_{SIG_{ssk^*}}]_{SIG_{ssk_D}}$ generated by D. In contrast to other protocols, in this scenario D issues the payment receipt instead of the PA. In addition, the PA stores in his database all the objects, such as C's AIKs and signature, received from D.

When there is a dispute concerning receipt of the correct content, C sends A the payment receipt and information that allows A to authenticate the anonymous identity of C. This normally contains C's TPM credentials. Upon verifying the validity of the information provided, A asks for a new copy of content from D and forwards this new copy to C. The protocol messages are shown in Figure 8.7.

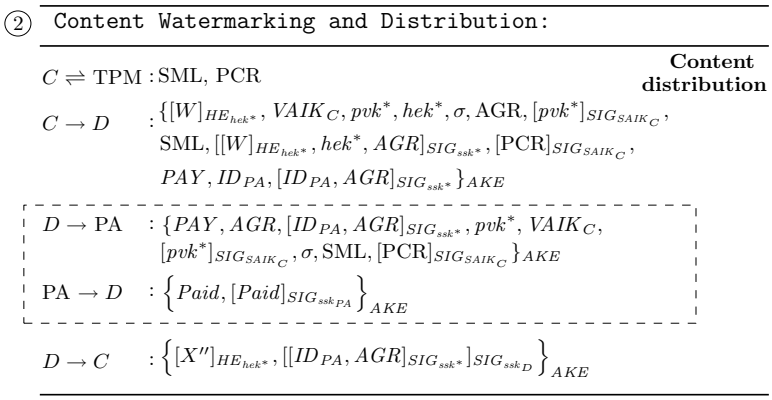

Figure 8.6: Adding PA and FE: The DAA Protocol

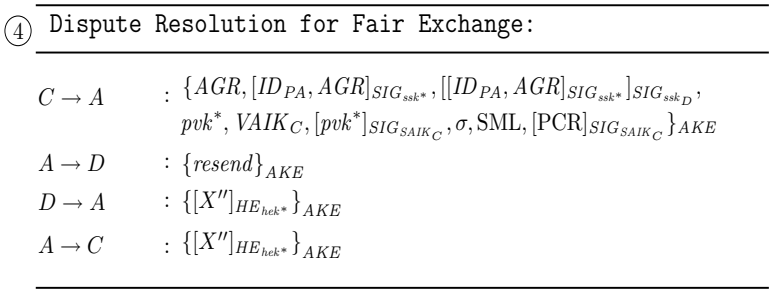

Figure 8.7: Dispute Resolution for FE: The DAA Protocol

As mentioned earlier, if during the content distribution phase D fails to provide content and a payment receipt to C, then C sends to A:

$$C \rightarrow A: \ \{AGR, [ID_{PA}, AGR]_{SIG_{ssk^*}}, pvk^*, VAIK_C, [pvk^*]_{SIG_{SAIK_C}}, \sigma, \text{SML}, [\text{PCR}]_{SIG_{SAIK_C}}\}_{AKE} \ ,$$

without a payment receipt. This message is validated by A based on the information provided by the PA to check whether C has paid for the said content as claimed. If this is the case then A asks D for a copy of content.

In the next section we present a FaCT protocol that includes payment and provides fair exchange.

8.3 A Protocol with Payment and Fair Exchange

We provide an example FaCT protocol that includes payment and provides fair exchange. We denote this protocol as *the FE protocol*. In this protocol, C does not receive the content until the payment is transferred to D. On the other hand, if after

making the payment the content is not received, or does not fit the description, then C requests via A that D re-issues the content.

Fundamentals. Table 8.1 shows the design framework of the FE protocol. In brief, the protocol involves C, D, a CA, a PA and A. The CA, PA and A are fully trusted. The PA also acts as a WCA to generate client watermarks. The FE protocol provides traceability, framing resistance, non-repudiation of redistribution, anonymity and unlinkability with respects to D, strong fairness for D and weak fairness for C.

Environment. This is a protocol with an online trusted third party. Its environment is similar to the LYTC protocol, on which it is based. It provides a `Dispute Resolution for Fair Exchange` phase to solve the fair exchange issue.

Table 8.1: The Design Framework of the FE Protocol

Fundamentals	
Parties Involved	C, D, A, CA, PA
Trust Assumptions	CA, PA and A are *fully trusted*
Security Properties	Traceability (**TR**), Framing resistance (**FR**), Non-repudiation of redistribution (**NR**), Anonymity and unlinkability (**AU**), Fair exchange (**FE**)
Environment	
Comp. Resources	Assume D and C have ample resources
Sec. comm. Support	Required
Pub. Key Support	Required
TTPs	Online TTP (PA)
Building Blocks	Digital watermarking scheme, homomorphic encryption scheme and digital signature scheme

`Initial Setup.` The main purpose of this phase is for C and D to register with the CA. It creates a pseudonym for C, so that C can later purchase content from D anonymously. It is identical to the `Initial Setup` phase of the LYTC protocol presented in Section 5.2. This means that after the completion of the initial setup, C possesses long term key pairs (hek_C, hdk_C) and (pvk_C, ssk_C), and a signature $[hek_C, pvk_C, ID_C]_{SIG_{ssk_{CA}}}$ generated by the CA. In addition, C possesses a pseudonym containing anonymous key pairs (hek^*_C, hdk^*_C) and (pvk^*_C, ssk^*_C), and an anonymous certificate $Cert_{ssk_{CA}}(pvk^*_C, hek^*_C)$.

`Content Watermarking and Distribution.` The main purpose of this phase is for C to receive correct content and D to receive correct payment. It involves the agreement between C and D on the price and description of the content that C wishes to purchase. In addition, C and D also agree on a PA for payment purposes.

We assume that the payment token PAY contains sufficient details for D to receive payment. Alternatively, PAY could denote a payment sub-protocol between C, D and the PA. Ways in which this can be achieved are surveyed in [119]. Figure 8.8 shows the protocol messages and the communication between the three parties is described below:

② Content Watermarking and Distribution:		
$C \rightarrow D$	: $\{pvk^*, hek^*, Cert_{ssk_{CA}}(pvk_C^*, hek_C^*), Cert_{ssk_C^*}(pvk^*, hek^*), PAY, ID_{PA}, AGR, [PAY, ID_{PA}, AGR]_{SIG_{ssk^*}}\}_{AKE}$	
$D \rightarrow$ PA	: $\{pvk^*, hek^*, Cert_{ssk_{CA}}(pvk_C^*, hek_C^*), Cert_{ssk_C^*}(pvk^*, hek^*), PAY, AGR, [PAY, ID_{PA}, AGR]_{SIG_{ssk^*}}\}_{AKE}$	
PA $\rightarrow D$	: $\{Paid, [Paid, [[W]_{HE_{hek^*}}]_{PE_{pek_D}}, AGR, [W]_{PE_{pek_{PA}}}]_{SIG_{ssk_{PA}}}, [[W]_{HE_{hek^*}}]_{PE_{pek_D}}, [W]_{PE_{pek_{PA}}}\}_{AKE}$	
PA $\rightarrow C$	: $\left\{[PAY, AGR]_{SIG_{ssk_{PA}}}\right\}_{AKE}$	**Content distribution**
$D \rightarrow C$	: $\{[X'']_{HE_{hek^*}}\}_{AKE}$	

Figure 8.8: FE Protocol – Content Watermarking and Distribution

(I) C requests content and approves a content agreement with D.

1. C selects a PA and checks AGR for later purchasing content from D.
2. C randomly generates a one-time encryption key pair (hek^*, hdk^*) and signature key pair (pvk^*, ssk^*). The public keys are signed to produce an anonymous certificate $Cert_{ssk_C^*}(pvk^*, hek^*)$.
3. C prepares payment PAY and generates a signature on (PAY, ID_{PA}, AGR). The signature $[PAY, ID_{PA}, AGR]_{SIG_{ssk^*}}$, produced using the one-time signing key ssk^*, is sent together with PAY, ID_{PA} and AGR to D. The one-time public keys pvk^* and hek^*, the new certificate $Cert_{ssk_C^*}(pvk^*, hek^*)$ and the anonymous certificate $Cert_{ssk_{CA}}(pvk_C^*, hek_C^*)$ generated by the CA are also sent to D.

(II) D sends PAY and requests a client watermark from the PA.

4. D verifies the signature, and sends $[PAY, ID_{PA}, AGR]_{SIG_{ssk^*}}$, PAY and AGR to the PA. He also sends C's anonymous certificate $Cert_{ssk_{CA}}(pvk^*_C, hek^*_C)$, hek^* and pvk^* to PA so that PA can verify the signature generated by C.

(III) The PA sends to D an encrypted client watermark. D also receives payment while C receives a payment receipt.

5. The PA verifies the certificate using the CA's public verification key pvk_{CA}. Upon successful verification, the PA verifies the signature using C's public verification key ssk^*. Next, the PA checks PAY, AGR and deposits PAY to D's account.

6. The PA chooses a unique client watermark W. Using C's encryption key, hek^*, PA encrypts W with a homomorphic encryption scheme, resulting in $[W]_{HE_{hek^*}}$. This will later be used by D to form the encrypted and watermarked content. The encrypted watermark is further encrypted under D's encryption key, resulting in $[[W]_{HE_{hek^*}}]_{PE_{pek_D}}$. Watermark W is also encrypted under the PA's encryption key as $[W]_{PE_{pek_{PA}}}$. Both these encryptions are to ensure confidentiality. This will be used, if necessary, during dispute resolution.

7. Next the PA generates a signature:

$$[Paid, [[W]_{HE_{hek^*}}]_{PE_{pek_D}}, AGR, [W]_{PE_{pek_{PA}}}]_{SIG_{ssk_{PA}}},$$

where $Paid$ denotes the payment transfer statement. The PA sends this signature, $Paid$, together with $[[W]_{HE_{hek^*}}]_{PE_{pek_D}}$ and $[W]_{PE_{pek_{PA}}}$ to D. The PA also generates a signature

$$[PAY, AGR]_{SIG_{ssk_{PA}}}$$

and sends this signature to C. For the PA, sending $[W]_{PE_{pek_{PA}}}$ to D means that the PA does not need to store W, while $[PAY, AGR]_{SIG_{ssk_{PA}}}$ serves as a payment receipt for C.

(IV) D produces a marked copy of the requested content and sends it to C.

8. D verifies the PA's signature, checks *Paid* and decrypts $[[W]_{HE_{hek^*}}]_{PE_{pek_D}}$ to retrieve the encrypted watermark $[W]_{HE_{hek^*}}$.

9. Next, D generates a watermark V and embeds V into the content X which C wishes to purchase, resulting in:

$$X' \leftarrow [X, V]_{EMB_{wmk_V}}.$$

With V embedded, D can check any content found at a later stage in order to verify D's ownership and confirm that the content belongs to C, by matching V to D's database.

10. D then uses the homomorphic encryption scheme to generate the encrypted marked content, in the same way that it is generated in the LYTC protocol (Section 5.2):

$$\left.\begin{array}{rl} & [x'_i]_{HE_{hek^*}} \cdot [w_i]_{HE_{hek^*}} \\ = & [x'_i \circ w_i]_{HE_{hek^*}} \\ = & [x''_i]_{HE_{hek^*}} \end{array}\right\} \; 1 \leq i \leq n,$$

where n is the number of elements in the watermark and content, and where $\circ$ represents either modular addition, modular multiplication or bit-wise XOR depending on the underlying homomorphic encryption used. The resulting encrypted marked content is denoted as $[X'']_{HE_{hek^*}}$. This encrypted marked content is sent to C.

11. C verifies the payment receipt $[PAY, AGR]_{SIG_{ssk_{PA}}}$ and decrypts the encrypted marked content $[X'']_{HE_{hek^*}}$ to get the watermarked content X''. C further checks if the content X'' is correct. If it is, then the purchase is completed. If not, C proceeds to the fair exchange phase.

`Identification and Dispute Resolution.` This protocol is executed if D discovers a copy of its sold contents and suspects C of having illegally distributed it. It allows for D to identify C responsible for redistributing the bought content and further to obtain a proof of this. The following explains how the identification and arbitration is carried out between D, A, the PA and the CA. Note that D does not need the cooperation of C to obtain a proof of the fact that C illegally distributed copies of content. Figure 8.9 shows the protocol messages.

(I) D detects a watermark from the found copy of content in order to identify the client that owns the content.

③ Identification and Dispute Resolution:		
D	: $\{\texttt{true}, \texttt{false}\} \leftarrow [\widehat{X}, V, X]_{DET_{wmk}}$	
$D \rightarrow A$	: $\{hek^*, pvk^*, Cert_{ssk_C^*}(pvk^*, hek^*), Cert_{ssk_{CA}}(pvk_C^*, hek_C^*), X', \widehat{X}, PAY, ID_{PA}, AGR, [PAY, ID_{PA}, AGR]_{SIG_{ssk^*}}\}_{AKE}$	
$A \rightarrow$ PA	: $\{watermark\ info?\}_{AKE}$	
PA $\rightarrow A$	: $\{watermark\ info\}_{AKE}$	**After content distribution**
A	: $\texttt{true} \leftarrow [\widehat{X}, W, X']_{DET_{wmk}}$	

Figure 8.9: FE Protocol – Identification and Dispute Resolution

1. D initiates by checking if found content $\widehat{X}$ is similar to content in D's database.
2. If so, D detects whether V is embedded in the content. If it is, D sends the following information to A to prove D's claim: the marked content X', the found copy $\widehat{X}$, the one-time public key hek^*, pvk^*, the certificate of them $Cert_{ssk_{ssk_C^*}}(pvk^*, hek^*)$, the pseudonym $Cert_{ssk_{CA}}(pvk_C^*, hek_C^*)$ of C, the agreement AGR, C's signature $[PAY, ID_{PA}, AGR]_{SIG_{ssk^*}}$ and the payment PAY.

(II) D proves to A that C illegally distributed copies of content.

3. A verifies the signature on the one-time key and the pseudonym to confirm that C's keys are valid, and uses pvk^* to verify C's payment offer. A also verifies C's signature on the purchase agreement. If the verifications are correct, A requests the client's watermark W by sending its encrypted version $[W]_{PE_{pek_{PA}}}$ to the PA.
4. The PA decrypts $[W]_{PE_{pek_{PA}}}$ and sends W back to A.
5. A detects whether the watermark W is embedded in the found content $\widehat{X}$. If W is detected, A decides that C is guilty and asks the CA to reveal C's real identity by sending C's pseudonym $Cert_{ssk_{CA}}(pvk_C^*, hek_C^*)$ to the CA.
6. The CA verifies the pseudonym and reveals the real identity of C to A.

`Dispute Resolution for Fair Exchange.` A content dispute occurs when the content received by C does not match the description, or if C does not receive anything at all, and C is unable to resolve the matter directly with D. Figure 8.10 shows the interactions between the parties involved in order to resolve the dispute.

④ `Dispute Resolution for Fair Exchange:`

$C \rightarrow A$: $\{hek^*, pvk^*, Cert_{ssk_C^*}(pvk^*, hek^*), Cert_{ssk_{CA}}(pvk_C^*, hek_C^*), PAY, ID_{PA}, AGR, [PAY, ID_{PA}, AGR]_{SIG_{ssk^*}}, [PAY, AGR]_{SIG_{ssk_{PA}}}\}_{AKE}$

$A \rightarrow D$: $\{resend\}_{AKE}$

$D \rightarrow A$: $\{[X'']_{HE_{hek^*}}\}_{AKE}$

$A \rightarrow C$: $\{[X'']_{HE_{hek^*}}\}_{AKE}$

Figure 8.10: FE Protocol – Dispute Resolution for Fair Exchange

Here C sends a request to A for D to resend the content, as described below:

1. C initiates by sending to A the objects needed by A to verify C's identity and C's claim. These include C's one-time public keys hek^*, pvk^*, the anonymous certificates $Cert_{ssk_{ssk_C^*}}(pvk^*, hek^*)$ and $Cert_{ssk_{CA}}(pvk_C^*, hek_C^*)$, the purchase agreement AGR, the signature $[PAY, ID_{PA}, AGR]_{SIG_{ssk^*}}$ and C's payment receipt $[PAY, AGR]_{SIG_{ssk_{PA}}}$.

2. A checks the certificates and verifies the signature. If they are valid, A sends a *resend* request to D. The payment receipt allows A to ascertain that D has agreed to sell the content and that payment has been made to D.

3. D prepares new encrypted marked content and sends this to A, which is then forwarded to C.

4. C decrypts the newly encrypted marked content $[X'']_{HE_{hek^*}}$ to get content X'', and checks X''.

8.3.1 Security

The security of the protocol with fair exchange is based on the security of the underlying building blocks and the fair exchange protocol in [98]. In the following we discuss how the security requirements stated in Table 8.1 are fulfilled.

Traceability. Traceability of content is preserved since D can trace content through the watermark V, which is embedded in the content in the `Content Watermarking and Distribution` phase.

Framing Resistance. Framing of C by D is not possible since both C and D have no knowledge of the resultant watermark embedded in the final copy possessed by C. This can be observed from the execution of the `Content Watermarking and Distribution` phase, in which C's watermark is generated and encrypted by the PA, and later embedded in the encrypted domain into the content by D using homomorphic encryption.

Non-repudiation of Redistribution. An undeniable proof of redistribution is in D's possession, namely C's signature $[PAY, ID_{PA}, AGR]_{SIG_{ssk^*}}$ binding C to the purchase agreement AGR. This information can be presented to A, and C cannot deny distributing bought content illegally when C's signature and AGR, which contains the description of the content, have been verified, and furthermore, C's watermark W is found embedded in the distributed content. The real identity of C can then be identified with the assistance of the CA.

Anonymity and unlinkability. The pseudonym utilised by C during a purchase consists of the one-time key pairs (hek^*, hdk^*) and (pvk^*, ssk^*), the signature $[hek^*, pvk^*]_{SIG_{ssk^*_C}}$ and the pseudonym $Cert_{ssk_{CA}}(pvk^*_C, hek^*_C)$. Anonymity of C is preserved with respect to D since there is no way for D to link the one-time key pairs to C's real identity. Unlinkability of C can be preserved by using unique pseudonym and one-time key pairs each time C buys content from D.

Fair Exchange. In the case where the protocol completes successfully after the execution of the `Content Watermarking and Distribution` phase, from C's perspective there is a guarantee that if C does not receive the content (or the content bought does not fit the description), C has a way to demand the correct content from D. From D's perspective, C will not get the content until D receives the payment. From this, we say that this protocol with fair exchange achieves *strong fairness* for D, since D can always be assured of receiving the payment before sending C the content. If not, D simply does not send the content. As for C, in the case where D receives the payment but C does not receive the content (or the content is corrupted), C can prove this to A and A will ask D to resend the content. In the scenario where C actually received the content but claims they have not, D can resend the content since there is no additional advantage for C to hold many identical digital copies.

8.3.2 Efficiency

In this section we discuss the performance of the protocol with fair exchange. The main objective here is to show that while the protocol includes the additional property of fair exchange, its efficiency is only marginally poorer than the LYTC protocol (Section 5.2) on which it is built. Table 8.2 summarises the performance.

Bandwidth. The size of the encrypted marked content $[X'']_{HE_{hek_C}}$ transmitted from C to D is $n|m|$. This is due to the homomorphic encryption with modulus m on the n elements of content. This is identical to the LYTC protocol. However, the FE protocol further requires extra objects to be included in messages to ensure D receives payment, and C receives content and a receipt. In addition, there is one extra message where the PA gives C a payment receipt.

Trusted Third Parties. The main extra requirement of the FE protocol is the PA. It is tasked with generating the client watermark, like the WCA in the LYTC protocol, but it also needs to ensure payment is received by D, and C receives a receipt. Hence the PA in the protocol has extra responsibility.

Computation. The computational requirements are identical to that of the LYTC protocol. This is because the FE protocol uses the same mechanism, homomorphic encryption, to produce the encrypted marked content. Hence, given that the number of elements in content and a watermark is n, D needs to compute n modular exponentiations ($n\mathbf{E}$) to encrypt content, and then n modular multiplications ($n\mathbf{M}$) to embed the watermark into content in encrypted form. Also, before encrypting content, D adds n elements of watermark V into content ($n\mathbf{A}$). As for C, n modular exponentiations ($n\mathbf{E}$) are required to decrypt the encrypted marked content.

Storage. Similarly, the storage requirement is identical to that of the LYTC protocol. This means that D stores the watermark V ($n|Z|$), the encrypted watermark ($n|m|$) and the encryption key of C ($|m|$), while C stores his encryption and decryption keys ($2|m|$).

Summary. Except for the extra objects in the protocol messages required between D, C and the PA, and one extra message from the PA to C, the performance of the FE protocol is comparable to that of the LYTC protocol.

Table 8.2: Efficiency Comparisons between LYTC Protocol and FE Protocol

Pro.	*Bandwidth*	*TTP*	*Computation*[1]	*Storage*[2]
LYTC	$[X'']_{HE_{hek_C}} = n\|m\|$	online WCA	C: $n\mathbf{E}$ D: $n(\mathbf{E}+\mathbf{M}+\mathbf{A})$	C: $2\|m\|$ D: $(n+1)\|m\| + n\|Z\|$
FE	$[X'']_{HE_{hek_C}} = n\|m\|$ 3 extra obj. & msg.	online WCA	C: $n\mathbf{E}$ D: $n(\mathbf{E}+\mathbf{M}+\mathbf{A})$	C: $2\|m\|$ D: $(n+1)\|m\| + n\|Z\|$

[1] $\mathbf{E}$=$O(k^3)$, $\mathbf{M}$=$O(k^2)$, $\mathbf{A}$=$O(k)$
[2] $|Z| < |m|$

8.4 Summary

In this chapter we have discussed the addition of payment into the main categories of FaCT protocol, and the requirement of fair exchange, which has not been discussed before in the context of fair content tracing. We reasoned that fair exchange is required, especially when payment is involved, since C and D do not trust each other. Hence we want to make sure not just that the content can be traced fairly, but also that C receives the correct content while D receives the correct payment.

We showed how this can be performed by adapting payment and fair exchange into FaCT protocols in all four main categories that we discussed in the previous chapters. We further presented a new protocol as an example.

Chapter 9

Conclusion

Contents

This chapter summarises the thesis. We discuss the main achievements and future research directions.

9.1 Main Achievements

FaCT protocols were proposed to address the concern of illegal content distribution by allowing a distributor to trace a client from a found copy of content and prove this fact to others. At the same time, these protocols ensure that the distributor cannot use such tracing ability to falsely accuse an innocent client of illegal content distribution.

The overall goal of the thesis is to analyse existing FaCT protocols and suggest alternative and better approaches to designing them. We believe that the thesis has contributed to this in the following ways:

- We proposed a design framework, since existing protocols in the literature are diverse and difficult to analyse. Based on this framework we were able to classify existing FaCT protocols into four categories, which facilitates more systematic analysis and better approaches to designing them. More importantly,

we were able to pinpoint design issues in some of the existing protocols.

- We examined existing protocols in the first category, which we termed *Protocols without Trusted Third Parties*, where the main characteristic is that the client is responsible of generating their own watermark. We analysed the IEH protocols and demonstrated design flaws, which we summarise in Table 9.1. We further proposed a Semi-Fair protocol that alleviates issues in the existing protocols. Our proposal is based on a stronger assumption, namely that the distributor is assumed to be more trustworthy than the client, which in turn forces the client to generate a well-formed watermark without requiring zero-knowledge proof systems.

- We examined *Protocols with Online Trusted Third Parties*, the second category in our classification. The main characteristic of these protocols is that an online trusted WCA generates the client watermarks. We cryptanalysed the ASSY protocol based on our framework by demonstrating design flaws in this protocol (Table 9.1). In particular, we showed that the protocol does not provide non-repudiation of redistribution due to the capability of D framing C and C successful denial of illegal content distribution.

- We examined *Protocols with Offline Trusted Third Parties*, the third category in our classification. We proposed a CE protocol based on Chameleon encryption [110]. As compared with the recently proposed KTIG protocol, which we discussed, the main advantage of our proposal is that the involvement of the trusted third party is a one-off process. This means that after the client and the distributor obtain key materials for the underlying Chameleon encryption scheme from the trusted third party, the client can conduct many content requests. In addition, as compared to conventional protocols that use asymmetric homomorphic encryption, our proposal is more computationally efficient as shown in the performance summary in Table 9.2.

- We examined *Protocols with Trusted Hardware*, the fourth category in our classification. We proposed two protocols based on trusted computing platforms [126]. Our first protocol is based on the Trusted Platform Module (TPM) and Direct Anonymous Attestation (DAA) [86]. The second protocol is based on the TPM and a Privacy CA. Both protocols also provide anonymity and unlinkability.

- We examined the addition of payment and, with this, the issue of fair exchange, which has not been discussed before for FaCT protocols. We demonstrated how payment and fair exchange can be incorporated in the four main categories of FaCT protocols. We also proposed a FE protocol with payment and fair exchange. This protocol provides the additional property of fair exchange to ensure D and C trade content fairly, but with an increase of communication cost as can be observed from Table 9.2.

Tables 9.1 and 9.2 summarise the security and performance of the FaCT protocols in the four categories that we have discussed (Section 3.5), where *C1* denotes **Category 1**, protocols without trusted third parties, *C2* denotes **Category 2**, protocols with online trusted third parties, *C3* denotes **Category 3**, protocols with offline trusted third parties and *C4* denotes **Category 4**, protocols with trusted hardware.

Table 9.1: Security Analysis of the FaCT protocols in the Four Categories

Pro./Sec.	**TR**	**FR**	**NR**	**AU**	**Conditions**
C1					
PS/4.2	✓	✓	✓	×	Homomorphic bit commitment scheme.
IEH/4.3	✓	×	×	×	Deterministic homomorphic encryption. Susceptible to attacks **I**, **II**, **III** (Section 4.3.3) and William-Treharne-Ho attack.
SF/4.4	✓	✓	✓	×	Stronger assumption: D semi-trusted.
C2					
LYTC/ 5.2	✓	✓	✓	✓	Deterministic homomorphic encryption.
WP/5.3	✓	✓	✓	×	C must not know ρ and β. Susceptible to unbinding attack.
ASSY/ 5.4	✓	×	×	×	Susceptible to attacks **I**, **II**, **III** (Section 5.4.1).
C3					
MW/3.7	✓	✓	✓	×	Susceptible to unbinding attack.
KTIG/ 6.2	✓	✓	✓	×	Relies on $h(.)$ being a secret function. Susceptible to unbinding attack.
CE/6.3	✓	✓	✓	×	Relies on the security of Chameleon encryption.
C4					
FCS/7.2	✓	✓	✓	✓	Abstract trusted hardware.
DAA/ 7.3.2	✓	✓	×	✓[1]	Security of TPM and DAA. A weak FaCT protocol.
P CA/ 7.3.3	✓	✓	✓	✓	Security of TPM.
AU+FE/ 8.2	✓	✓	✓	✓	Further provides fair exchange.

[1] *full anonymity and unlinkability*

Table 9.2: Performance of the FaCT Protocols in the Four Categories

Pro./ Sec.	**Bandwidth**[1]	**TTP**	**Computation**[2]	**Storage**[1]
C1				
PS/ 4.2	$[X'']_{COM_{hek_C}} = n\|m\|$ y extra pro. msg.	No TTP	C: $n(y+1)\mathbf{E}$ D: $n(\mathbf{E}+\mathbf{M})$	C: $2\|m\| + n\|Z\|$ D: $(n+1)\|m\| + n\|Z\|$
IEH/ 4.3	$[X'']_{HE_{hek_C}} = n\|m\|$	No TTP	C: $2n\mathbf{E}$ D: $n(\mathbf{E}+\mathbf{M}+\mathbf{A})$	C: $2\|m\| + n\|Z\|$ D: $(n+1)\|m\| + n\|Z\|$
SF/ 4.4	$[X'']_{HE_{hek_C}} = n\|m\|$	No TTP	C: $2n\mathbf{E}$ D: $n(\mathbf{E}+\mathbf{M}+\mathbf{A})$	C: $2\|m\| + n\|Z\|$ D: $(n+1)\|m\| + n\|Z\|$
C2				
LYTC/ 5.2	$[X'']_{HE_{hek_C}} = n\|m\|$	online WCA	C: $n\mathbf{E}$ D: $n(\mathbf{E}+\mathbf{M}+\mathbf{A})$	C: $2\|m\|$ D: $(n+1)\|m\| + n\|Z\|$
WP/ 5.3	$X' = n\|Z\|$	online WCA	C: $n\mathbf{A}$ D: $n\mathbf{A}$	C: $n\|Z\|(0)$ D: $n\|Z\|$
FE/ 8.2	$[X'']_{HE_{hek_C}} = n\|m\|$ 3 extra obj. & msg.	online WCA	C: $n\mathbf{E}$ D: $n(\mathbf{E}+\mathbf{M}+\mathbf{A})$	C: $2\|m\|$ D: $(n+1)\|m\| + n\|Z\|$
C3				
MW/ 3.7	$[X'']_{HE_{hek_C}} = n\|m\|$	offline WCA	C: $n\mathbf{E}$ D: $n(\mathbf{E}+\mathbf{M}+\mathbf{A})$	C: $2\|m\| + n\|Z\|$ D: $(n+1)\|m\| + n\|Z\|$
KTIG/ 6.2	$\sigma[X']_{E_K} = 2n\|Z\|$	offline KC	C: $2n\mathbf{S}$ D: $2n(\mathbf{A}+\mathbf{S})$	C: $n\|Z\|(0)$ D: $(L+n)\|Z\|$
CE/ 6.3	$E^v = n\|Z\|$	offline KC[3]	C: $ns\mathbf{A}$ D: $ns\mathbf{A}$	C: $L\|Z\|$ $(n\|Z\|)$ D: $L\|Z\|$ $(n\|Z\|)$
C4				
FCS 7.2	$[X'']_{HE_{hek^*}} = n\|m\|$	D's TH[3]	C: $n\mathbf{E}$ D: $n(\mathbf{E}+\mathbf{M}+\mathbf{A})$	C: $2\|m\|$ D: $(n+1)\|m\| + n\|Z\|$
DAA/ **PCA**/ 7.3.2/ 7.3.3	$[X'']_{HE_{hek^*}} = n\|m\|$	C's TH	C: $2n\mathbf{E}$ D: $n(\mathbf{E}+\mathbf{M}+\mathbf{A})$	C: $2\|m\| + n\|Z\|$ D: $(n+1)\|m\| + n\|Z\|$

[1] $|Z| < |m|$, $|n| < |L|$
[2] $\mathbf{E}$=$O(k^3)$, $\mathbf{M}$=$O(k^2)$, $\mathbf{A}$=$O(k)$, $\mathbf{S}$=$\mathbf{E}/100$
[3] Higher implementation cost compared to the DAA and PCA protocols

9.2 Research Directions

In the following we describe some potential research directions:

- In our study on FaCT protocols, we have not considered the performance of these protocols in constrained computing environments. In particular we have not considered FaCT protocols that execute on a client's computing platform that has only small memory size and computing power. In such cases, using watermarking in the encrypted domain based on asymmetric homomorphic

encryption to provide fair content tracing is likely to be too expensive in terms of computations and bandwidth. Alternatives such as the protocols described in Chapters 5 and 6 that deploy symmetric building blocks could be further studied for this purpose.

- Also, only a heuristic approach to security analysis has been taken. We have not considered the use of formal approaches to access the security of the protocols. This is because formal analysis for many of the underlying building blocks, particularly digital watermarking schemes, is still very much in its infancy, as can be observed in [2, 64]. Developing formal security will also require the development of a standard and stable notion of security for the underlying digital watermarking schemes, which is out of the scope of our study. We note that a formal modeling approach has been initiated by Williams *et al.* [134, 135].

- We have only considered the distribution of content as a whole. This means, for example, the distributor sends a pre-recorded movie or song to the client. We have not considered streaming of content. It would be interesting to investigate how the existing FaCT protocols can be adapted to a scenario where the distributor *streams* content to the client, while still providing the required security properties. The key issue, again, is efficiency, since small blocks of content are sent in real time. Encrypting and watermarking them based on asymmetric homomorphic encryption will be too computationally expensive.

Bibliography

[1] A. Adelsbach, U. Huber, and A.-R. Sadeghi. Fingercasting-Joint Fingerprinting and Decryption of Broadcast Messages. In L. M. Batten and R. Safavi-Naini, editors, *Information Security and Privacy, 11th Australasian Conference - ACISP 2006*, volume 4058 of *Lecture Notes in Computer Science*, pages 136–147. Springer-Verlag, 2006. Also, Technical Report detailing the ACISP 2006 paper (Private communication with U. Huber).

[2] A. Adelsbach, S. Katzenbeisser, and A.-R. Sadeghi. A Computational Model for Watermark Robustness. In Jan Camenisch, Christian S. Collberg, Neil F. Johnson, and Phil Sallee, editors, *8th International Workshop on Information Hiding - IH 2006*, volume 4437 of *Lecture Notes in Computer Science*, pages 145–160. Springer-Verlag, 2006.

[3] F. Ahmed, F. Sattar, M. Y. Siyal, and D. Yu. A Secure Watermarking Scheme for Buyer-Seller Identification and Copyright Protection. *EURASIP Journal on Applied Signal Processing*, 2006:56904, 15 pages, 2006. doi:10.1155/ASP/2006/56904.

[4] R. Anderson, C. Manifavas, and C. Sutherland. Netcard - a practical electronic cash system. In T. Mark A. Lomas, editor, *Security Protocols Workshop 1996*, volume 1189 of *Lecture Notes in Computer Science*, pages 49–57. Springer-Verlag, 1997.

[5] N. Asokan, M. Schunter, and M. Waidner. Optimistic Protocols for Fair Exchange. In *Proceedings of 4th ACM Conference on Computer and Communications Security*, pages 7–17, 1997.

[6] Boris Balacheff, Liqun Chen, Siani Pearson, David Plaquin, and Graeme Proudler. *Trusted Computing Platforms: TCPA Technology in Context*. Prentice Hall PTR, Upper Saddle River, New Jersey, 2003.

[7] British Broadcasting Corporation (BBC). BBC iPlayer - An Internet TV and Radio Broadcasting Services, accessed February 2009. Available at: `www.bbc.co.uk/iplayer`.

[8] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated Key Exchange Secure against Dictionary Attacks. In B. Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 139–155. Springer-Verlag, 2000.

[9] M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. In D. R. Stinson, editor, *Advances in Cryptology - CRYPTO 1993*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249. Springer-Verlag, 1994.

[10] M. Bellare and P. Rogaway. Optimal Asymmetric Encryption. In Alfredo De Santis, editor, *Advances in Cryptology - EUROCRYPT 1994*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer-Verlag, 1994.

[11] P. Biddle, P. England, M. Peinado, and B. Willman. The Darknet and the Future of Content Protection. In Joan Feigenbaum, editor, *Proceedings of the 2002 ACM Workshop on Digital Rights Management*, volume 2696 of *Lecture Notes in Computer Science*, pages 155–176. Springer-Verlag, 2003.

[12] I. Biehl and B. Meyer. Protocols for Collusion-Secure Asymmetric Fingerprinting. In *14th Symposium on Theoretical Aspects of Computer Science (STACS)*, 1997.

[13] Simon Blake-Wilson and Alfred Menezes. Entity Authentication and Authenticated Key Transport Protocols Employing Asymmetric Techniques. In B. Christianson, B. Crispo, T. Mark A. Lomas, and M. Roe, editors, *Security Protocols*, volume 1361 of *Lecture Notes in Computer Science*, pages 137–158. Springer-Verlag, 1997.

[14] G. R. Blakley, C. Meadows, and G. B. Purdy. Fingerprinting Long Forgiving Messages. In H. C. Williams, editor, *Advances in Cryptology - CRYPTO 1985*, volume 218 of *Lecture Notes in Computer Science*, pages 180–189. Springer-Verlag, 1985.

[15] D. Boneh and J. Shaw. Collusion-Secure Fingerprinting for Digital Data. In D. Coppersmith, editor, *Advances in Cryptology - CRYPTO 1995*, volume 963 of *Lecture Notes in Computer Science*, pages 452–465. Springer-Verlag, 1995.

[16] C. Boyd and A. Mathuria. *Protocols for Authentication and Key Establishment*. Information Security and Cryptography Series, Springer-Verlag, 2003.

[17] G. Brassard, David Chaum, and C. Crepeau. Minimum Disclosure Proofs of Knowledge. *Journal of Computer and System Sciences*, 37(1988):156–189, 1988.

[18] E. Brickell, J. Camenisch, and L. Chen. Direct Anonymous Attestation. In *Proceedings of 11th ACM Conference on Computer and Communications Security*, pages 132–145. ACM Press, 2004.

[19] J. Camenisch. Efficient Anonymous Fingerprinting with Group Signatures. In T. Okamoto, editor, *Advances in Cryptology - ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 415–428. Springer-Verlag, 2000.

[20] D. Chaum, A. Fiat, and M. Naor. Untraceable Electronic Cash. In S. Goldwasser, editor, *Advances in Cryptology - CRYPTO 1988*, volume 403 of *Lecture Notes in Computer Science*, pages 319–327. Springer-Verlag, 1990.

[21] B. Chen and G. W. Wornell. Quantization index modulation: A class of provably good methods for digital watermarking and information embedding. *IEEE Transaction on Information Theory*, 47(4):1423–1443, 2001.

[22] L. Chen, C. Kudla, and K. G. Paterson. Concurrent Signatures. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 287–305. Springer-Verlag, 2004.

[23] J.-G. Choi, G. Hanaoka, K. H. Rhee, and H. Imai. How to break COT-Based Fingerprinting Schemes and Design New One. *IEICE Trans. on Fundamentals, Special Section on Information Theory and Its Applications*, E88-A(10):2800–2807, 2005.

[24] J.-G. Choi and J.-H. Park. A Generalization of an Anonymous Buyer-Seller Watermarking Protocol and Its Application to Mobile Communications. In

I. J. Cox, T. Kalker, and H.-K. Lee, editors, *Digital Watermarking, Third International Workshop - IWDW 2004*, volume 3304 of *Lecture Notes in Computer Science*, page p. 232. Springer-Verlag, 2004.

[25] J.-G. Choi, K. Sakurai, and J.-H. Park. Does It Need Trusted Third Party? Design of Buyer-Seller Watermarking Protocol without Trusted Third Party. In J. Zhou, M. Yung, and Y. Han, editors, *Applied Cryptography and Network Security - ACNS 2003*, volume 2846 of *Lecture Notes in Computer Science*, pages 265–279. Springer-Verlag, 2003.

[26] Jae-Gwi Choi, Ji-Hwan Park, and Ki-Ryong Kwon. Analysis of COT-based Fingerprinting Schemes: New Approach to Design Practical and Secure Fingerprinting Scheme. In Jessica J. Fridrich, editor, *6th International Workshop on Information Hiding - IH 2004*, volume 3200 of *Lecture Notes in Computer Science*, pages 253–265. Springer-Verlag, 2004.

[27] Microsoft Corporation. Windows Media Digital Rights Management, accessed March 2009. Available at: `http://www.microsoft.com/windows/windowsmedia/forpros/drm/default.mspx`.

[28] I. J. Cox, J. Kilian, T. Leighton, and T. Shamoon. Secure Spread Spectrum Watermarking for Multimedia. *IEEE Trans. on Image Processing*, 6(12):1673–1687, 1997.

[29] I. J. Cox, M. L. Miller, J. A. Bloom, J. Fridrich, and T. Kalker. *Digital Watermarking and Steganography*. 2nd Edition, Morgan Kaufmann Publishers, 2008.

[30] C. Culnane, H. Treharne, and Anthony T. S. Ho. A New Multi-set Modulation Technique for Increasing Hiding Capacity of Binary Watermark for Print and Scan Processes. In Yun-Qing Shi and Byeungwoo Jeon, editors, *Digital Watermarking, Fifth International Workshop, IWDW 2006*, volume 4283 of *Lecture Notes in Computer Science*, pages 96–110. Springer-Verlag, 2006.

[31] W. Dai. Crypto++ Library 5.5 Benchmark, accessed February 2009. Available at: `http://www.cryptopp.com/benchmarks.html`.

[32] M. Deng and B. Preneel. Attacks On Two Buyer-Seller Watermarking Protocols And An Improvement For Revocable Anonymity. In *IEEE International Symposium on Electronic Commerce and Security - ISECS 2008*, 2008.

[33] M. Deng and B. Preneel. On Secure and Anonymous Buyer-Seller Watermarking Protocol. In *Third International Conference on Internet and Web Applications and Services, ICIW 2008*, pages 524–529. IEEE Computer Society, 2008.

[34] M. Deng, L. Weng, and B. Preneel. Anonymous Buyer-Seller Watermarking Protocol with Additive Homomorphism. In *SIGMAP 2008 - International Conference on Signal Processing and Multimedia Applications*, pages 300–307, 2008.

[35] A. Dent and C. Mitchell. *User's Guide to Cryptography and Standards*. Artech House, 2004.

[36] T. Dierks and E. Rescorla. The TLS Protocol Version 1.1. *RFC 4346*, 2006.

[37] H. Dobbertin, A. Bosselaers, and B. Preneel. RIPEMD-160: A Strengthened Version of Ripemd. In D. Gollmann, editor, *Fast Software Encryption - FSE 1996*, volume 1039 of *Lecture Notes in Computer Science*, pages 71–82. Springer-Verlag, 1996.

[38] D. Dolev and A. C. Yao. On the security of public key protocols. In *IEEE 22nd Annual Symposium on Foundations of Computer Science*, pages 350 – 357. IEEE Computer Society Press, 1981.

[39] J. Domingo-Ferrer. Anonymous Fingerprinting of Electronic Information with Automatic Identification of Redistributors. *IEEE Electronics Letters*, 43(13):1303–1304, 1998.

[40] J. Domingo-Ferrer. Anonymous Fingerprinting Based on Committed Oblivious Transfer. In H. Imai and Y. Zheng, editors, *Second International Workshop on Practice and Theory in Public-Key Cryptography - PKC 1999*, volume 1560 of *Lecture Notes in Computer Science*, pages 43–52. Springer-Verlag, 1999.

[41] S. Katzenbeisser (editor). List of potential applications interested by s.p.e.d. *D3.1, Philips Research (Philips), for Signal Processing in the Encrypted Domain (SPEED) Project, IST-2006-034238, Information Society Technologies*, 2007. Available at: `www.speedproject.eu`.

[42] P. Ekdahl and T. Johansson. A New Version of the Stream Cipher SNOW. In K. Nyberg and H. M. Heys, editors, *Selected Areas in Cryptography, 9th*

Annual International Workshop, SAC 2002, volume 2595 of *Lecture Notes in Computer Science*, pages 47–61. Springer-Verlag, 2003.

[43] T. Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469 – 472, 1985.

[44] Z. Erkin, A. Piva, S. Katzenbeisser, R. L. Lagendijk, J. Shokrollahi, G. Neven, and M. Barni. Protection and Retrieval of Encrypted Multimedia Content: When Cryptography Meets Signal Processing. *EURASIP Journal on Information Security*, 2007:78943, 20 pages, 2007. doi:10.1155/2007/78943.

[45] S. Even and Y. Yacobi. Relations among public key signature systems. *Technical Report 175, Technion, Haifa, Israel*, March 1980.

[46] C.-I Fan, M.-T. Chen, and W.-Z. Sun. Buyer-Seller Watermarking Protocols with Off-line Trusted Parties. In *Proceedings of the International Conference on Multimedia and Ubiquitous Engineering (MUE'07)*, pages 1035–1040. IEEE Compute Society Press., 2007.

[47] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *Advances in Cryptology - CRYPTO 1986*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer-Verlag, 1987.

[48] C. Fontaine and F. Galand. A Survey of Homomorphic Encryption for Nonspecialists. *EURASIP Journal on Information Security*, 2007:13801, 10 pages, 2007. doi:10.1155/2007/13801.

[49] M. Franklin and M. K. Reiter. Fair Exchange with a Semi-trusted Third Party. In *Proceedings of 4th ACM Conference on Computer and Communications Security*, pages 1–5, 1997.

[50] F. Frattolillo and S. D'Onofrio. A Web Oriented and Interactive Buyer-Seller Watermarking Protocol. In *Security, Steganography, and Watermarking of Multimedia Content VIII, Proc. of SPIE*, volume 6072, pages 718–716, 2006.

[51] C. Gehrmann and M. Naslund (ERICS) (editors). ECRYPT Yearly Report on Algorithms and Keysizes (2006). *D.SPA.21, Katholieke Universiteit Leuven*

(KUL), for ECRYPT Project, IST-2002-507932, Information Society Technologies, 2006. Available at: `www.ecrypt.eu.org/documents/D.SPA.21-1.1.pdf`.

[52] Gnutella.com. Gnutella website, accessed February 2009. Available at: `www.gnutella.com`.

[53] B.-M. Goi, Raphael C.-W. Phan, and M. U. Siddiqi. Cryptanalysis of a Generalized Anonymous Buyer-seller Watermarking Protocol of IWDW 2004. In T. Enokido, L. Yan, B. Xiao, D. Kim, Y.-S. Dai, and L. T. Yang, editors, *Embedded and Ubiquitous Computing - EUC 2005*, volume 3823 of *Lecture Notes in Computer Science*, pages 936–944. Springer-Verlag, 2005.

[54] B.-M. Goi, Raphael C.-W. Phan, Y. Yang, F. Bao, Robert H. Deng, and M. U. Siddiqi. Cryptanalysis of Two Anonymous Buyer-Seller Watermarking Protocols and an Improvement for True Anonymity. In M. Jakobsson, M. Yung, and J. Zhou, editors, *Applied Cryptography and Network Security - ACNS 2004*, volume 3089 of *Lecture Notes in Computer Science*, pages 369–382. Springer-Verlag, 2004.

[55] O. Goldreich. Zero-Knowledge twenty years after its invention. *Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel*, 2002.

[56] S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.

[57] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 291–304, 1985.

[58] S. Goldwasser, S. Micali, and R. L. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM Journal of Computing*, 17(2):281–308, 1988.

[59] Joint Photographic Experts Group. JPEG website, accessed February 2009. Available at: `www.jpeg.org/jpeg/index.html`.

[60] Moving Picture Experts Group. MPEG website, accessed February 2009. Available at: `www.chiariglione.org/mpeg`.

[61] M. Hirt and K. Sako. Efficient Receipt-Free Voting Based on Homomorphic Encryption. In B. Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 539–556. Springer-Verlag, 2000.

[62] Anthony T. S. Ho and F. Shu. A Robust Spread-Spectrum Watermarking Method Using Two-Level Quantization. In *Proceedings of the 2004 International Conference on Image Processing (ICIP 2004)*, volume 2, pages 725–728. IEEE, 2004.

[63] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall International, 1985.

[64] Nicholas Hopper, David Molnar, and David Wagner. From Weak to Strong Watermarking. In S. P. Vadhan, editor, *Theory of Cryptography - TCC 2007*, volume 4392 of *Lecture Notes in Computer Science*, pages 362–382. Springer-Verlag, 2007.

[65] I. M. Ibrahim, S. H. Nour El-Din, and A. F. A. Hegazy. An Effective and Secure Buyer-Seller Watermarking Protocol. In *Third International Symposium on Information Assurance and Security (IAS 07), IEEE Computer Society Press*, pages 21–26, 2007.

[66] I. M. Ibrahim, S. H. Nour El-Din, and A. F. A. Hegazy. An Effective and Secure Watermarking Protocol for Digital Rights Protection Over the Second-Hand Market. In *SECRYPT 2007 - International Conference on Security and Cryptography*, pages 263–268, 2007.

[67] Amazon.com Inc. Amazon Unbox, accessed February 2009. Available at: `www.amazon.com`.

[68] Apple Inc. iTune Store, accessed February 2009. Available at: `www.apple.com/itunes/store`.

[69] CinemaNow Inc. CinemaNow, accessed February 2009. Available at: `www.cinemanow.com`.

[70] Federal information processing standards (fips 180-2). Secure Hash Standard, 2001. Available at: `csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf`.

[71] Federal information processing standards (fips 186-2). Digital Signature Standard (DSS), 2001. Available at: csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf.

[72] Federal information processing standards (fips 197). Advanced Encryption Standard (AES), 2001. Available at: csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[73] ISO. Information Technology - Security Techniques - Entity Authentication Mechanisms - Part 3: Entity Authentication Using a Public Key Algorithm ISO/IEC 9798-3. *ISO/IEC International Standard, 2nd Edition*, 1998.

[74] H. S. Ju, H. J. Kim, D. H. Lee, and J. I. Lim. An Anonymous Buyer-Seller Watermarking Protocol with Anonymity Control. In P. J. Lee and C. H. Lim, editors, *Information Security and Cryptology - ICISC 2002*, volume 2587 of *Lecture Notes in Computer Science*, pages 421–432. Springer-Verlag, 2002.

[75] S. Katzenbeisser and F. A. P. Petitcolas, editors. *Information Hiding: Techniques for Steganography and Digital Watermarking*. Computer Security Series. Artech House, 2000.

[76] S. Katzenbeisser, B. Skorić, M. U. Celik, and A.-R. Sadeghi. Combining Tardos Fingerprinting Codes and Fingercasting. In *9th International Workshop on Information Hiding - IH 2007*, volume 4567 of *Lecture Notes in Computer Science*, pages 294–310. Springer-Verlag, 2007.

[77] J. Kilian, F. T. Leighton, L. R. Matheson, T. G. Shamoon, R. E. Tarjan, and F. Zane. Resistance of Digital Watermarks to Collusive Attacks. Technical Report TR-585-98, Princeton University, Department of Computer Science, 1988. Available at: ftp://ftp.cs.princeton.edu/techreports/1998/585.ps.gz.

[78] M. Kim, J. Kim, and K. Kim. Anonymous fingerprinting as secure as the bilinear diffie-hellman assumption. In R. H. Deng, S. Qing, F. Bao, and J. Zhou, editors, *Information and Communications Security - ICICS 2002*, volume 2513 of *Lecture Notes in Computer Science*, pages 97–108. Springer-Verlag, 2002.

[79] N. Koblitz. *A Course in Number Theory and Cryptography*. Graduate Texts in Mathematics Series, Vol. 114, 2nd Ed., Springer-Verlag, 1994.

[80] M. Kuribayashi and H. Tanaka. A New Anonymous Fingerprinting Scheme with High Enciphering Rate. In C. Pandu Rangan and Cunsheng Ding, editors, *Progress in Cryptology - INDOCRYPT 2001*, volume 2247 of *Lecture Notes in Computer Science*, pages 30–39. Springer-Verlag, 2001.

[81] M. Kuribayashi and H. Tanaka. Fingerprinting Protocol for Online-Line Trade Using Information Gap between Buyer and Merchant. *IEICE Trans. on Fundamentals*, E89-A(4):1108–1115, 2006.

[82] RSA Labs. RSA Encryption Scheme - Optimal Asymmetric Encryption Padding, 2000. Available at: `ftp://ftp.rsasecurity.com/pub/rsalabs/rsa_algorithm/rsa-oaep_spec.pdf`.

[83] RSA Labs. RSA Signature Scheme with Appendix - Probabilistic Signature Scheme, 2000. Available at: `ftp://ftp.rsasecurity.com/pub/rsalabs/rsa_algorithm/nessie_pss.zip`.

[84] RSA Labs. How Fast Is The RSA Algorithm?, accessed February 2009. Available at: `www.rsa.com/rsalabs/node.asp?id=2215`.

[85] C.-L. Lei, P.-L. Yu, P.-L. Tsai, and M.-H. Chan. An Efficient and Anonymous Buyer-Seller Watermarking Protocol. *IEEE Trans. on Image Processing*, 13(12):1618–1626, 2004.

[86] A. Leung and G. S. Poh. An Anonymous Watermarking Scheme for Content Distribution Protection using Trusted Computing. In *SECRYPT 2007 - International Conference on Security and Cryptography*, pages 319–326. INSTICC Press., 2007.

[87] K. J. Ray Liu, W. Trappe, Z. J. Wang, M. Wu, and H. Zhao. *Multimedia Fingerprinting Forensics for Traitor Tracing*. EURASIP Book Series on Signal Processing and Communication, Volume 4, Hindawi Publishing Corporation, 2005.

[88] YouTube LLC. YouTube - broadcast yourself, accessed February 2009. Available at: `www.youtube.com`.

[89] S. H. Low, N. F. Maxemchuk, and S. Paul. Anonymous Credit Cards and Their Collusion Analysis. *IEEE/ACM Trans. on Networking*, 4(6):809–816, 1996.

[90] M. T. Malkin. *Cryptographic Methods In Multimedia Identification And Authentication.* PhD thesis, Department of Computer Science, Stanford University, 2006.

[91] K. Markantonakis, K. Mayes, and F. Piper. Smart Cards for Security and Assurance. In H. R. Rao and S. J. Upadhyaya M. Gupta, editors, *Managing Information Assurance in Financial Services*, pages 166–189. IGI Publishing Hershey - New York, 2007.

[92] O. Markowitch, D. Gollmann, and S. Kremer. On Fairness in Exchange Protocols. In P. J. Lee and C. H. Lim, editors, *5th International Conference on Information Security and Cryptology - ICISC 2002*, volume 2587 of *Lecture Notes in Computer Science*, pages 451–464. Springer-Verlag, 2002.

[93] Mastercard. Mastercard payment solutions, accessed February 2009. Available at: `http://www.mastercard.com`.

[94] N. Memon and P. W. Wong. A Buyer-Seller Watermarking Protocol. *IEEE Trans. on Image Processing*, 10(4):643–649, 2001.

[95] Chris J Mitchell, editor. *Trusted Computing.* IEE Press, 2005.

[96] P. Moulin and R. Koetter. Data hiding codes. *Invited Paper, Proceedings of The IEEE*, 93(10), 2005.

[97] T. Okamoto and S. Uchiyama. A New Public-Key Cryptosystem as Secure as Factoring. In K. Nyberg, editor, *Advances in Cryptology - EUROCRYPT 1998*, volume 1403 of *Lecture Notes in Computer Science*, pages 308–318. Springer-Verlag, 1998.

[98] H. Pagnia, H. Vogt, and F. C. Gartner. Fair Exchange. *The Computer Journal, British Computer Society*, 46(1):55–75, 2003.

[99] P. Paillier. Public-key Cryptosystems Based on Composite Degree Residuosity Classes. In J. Stern, editor, *Advances in Cryptology - EUROCRYPT 1999*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer-Verlag, 1999.

[100] K. G. Paterson. Id-based signatures from pairings on elliptic curves. *IEEE Electronics Letters*, 38(18):1025–1026, 2002. Also available at IACR ePrint: `eprint.iacr.org/2002/004`.

[101] PayPal and Inc. PayPal - a safer, simpler way to send and receive money online, accessed February 2009. Available at: `http://www.paypal.com`.

[102] B. Pfitzmann and A.-R. Sadeghi. Coin-Based Anonymous Fingerprinting. In J. Stern, editor, *Advances in Cryptology - EUROCRYPT 1999*, volume 1592 of *Lecture Notes in Computer Science*, pages 150–164. Springer-Verlag, 1999.

[103] B. Pfitzmann and A.-R. Sadeghi. Anonymous Fingerprinting with Direct Non-Repudiation. In T. Okamoto, editor, *Advances in Cryptology - ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 401–414. Springer-Verlag, 2000.

[104] B. Pfitzmann and M. Schunter. Asymmetric Fingerprinting. In U. M. Maurer, editor, *Advances in Cryptology - EUROCRYPT 1996*, volume 1070 of *Lecture Notes in Computer Science*, pages 84–95. Springer-Verlag, 1996.

[105] B. Pfitzmann and M. Waidner. Anonymous Fingerprinting. In W. Fumy, editor, *Advances in Cryptology - EUROCRYPT 1997*, volume 1233 of *Lecture Notes in Computer Science*, pages 88–102. Springer-Verlag, 1997.

[106] B. Pfitzmann and M. Waidner. Asymmetric Fingerprinting for Larger Collusions. In *4th ACM Conference on Computer and Communication Security*, pages 151–160, 1997.

[107] Raphael C.-W. Phan and B.-M. Goi. (In)security of an Efficient Fingerprinting Scheme with Symmetric and Commutative Encryption of IWDW 2005. In *Digital Watermarking, 6th International Workshop - IWDW 2007*, Lecture Notes in Computer Science. Springer-Verlag, 2007.

[108] A. Piva and A. D. Rosa (editors). State of the art report and accompanying public presentation(s). *D2.1, Universit'a degli Studi di Firenze (UNIFI), for Signal Processing in the Encrypted Domain (SPEED) Project, IST-2006-034238, Information Society Technologies*, 2007. Available at: `www.speedproject.eu`.

[109] G. S. Poh and K. M. Martin. A Framework for Design and Analysis of Asymmetric Fingerprinting Protocols. In *2007 International Workshop on Data Hiding for Information and Multimedia Security attached to IAS 07, IEEE Computer Society Press*, pages 457–461, 2007.

[110] G. S. Poh and K. M. Martin. An Efficient Buyer-Seller Watermarking Scheme Based on Chameleon Encryption. In H. J. Kim, S. Katzenbeisser, and Anthony T. S. Ho, editors, *To appear in Digital Watermarking, Seventh International Workshop, IWDW 2008*, Lecture Notes in Computer Science. Springer-Verlag, 2008.

[111] G. S. Poh and K. M. Martin. On the (In)security of Two Buyer-Seller Watermarking Protocols. In *SECRYPT 2008 - International Conference on Security and Cryptography*, pages 253–260, 2008.

[112] G. S. Poh and K. M. Martin. Design Flaws of A Secure Watermarking Scheme for Buyer-Seller Identification and Copyright Protection. *International Journal of Cryptology Research*, 1(1):55–64, 2009.

[113] G. S. Poh and K. M. Martin. On the Design of Buyer-Seller Watermarking Protocols Without A Watermark Authority. In *E-Business and Telecommunication Networks (To appear)*, Communications in Computer and Information Science. Springer-Verlag, 2009.

[114] Niladri B. Puhan and Anthony T. S. Ho. Secure Authentication Watermarking for Localization Against the Holliman-Memon Attack. *Multimedia Systems*, 12(6):521–532, 2007.

[115] L. Qiao and K. Nahrstedt. Watermarking schemes and protocols for protecting rightful ownerships and customer's rights. *Journal of Visual Communication and Image Representation*, 9(3):194–210, 1998.

[116] I. Ray, I. Ray, and N. Natarajan. An Anonymous and Failure Resilient Fair-Exchange E-Commerce Protocol. *Decision Support Systems*, 39:267–292, 2005.

[117] R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. of the ACM*, 2(2):120–126, 1978.

[118] A.-R. Sadeghi. How to Break A Semi-Anonymous Fingerprinting Scheme. In I. S. Moskowitz, editor, *4th International Workshop on Information Hiding - IH 2001*, volume 2137 of *Lecture Notes in Computer Science*, pages 384–394. Springer-Verlag, 2001.

[119] A.-R. Sadeghi and M. Schneider. Electronic Payment Systems. In E. Becker, W. Buhse, D. Günnewig, and N. Rump, editors, *Digital Rights Management - Technological, Economic, Legal and Political Aspects*, volume 2770 of *Lecture Notes in Computer Science*, page 113137. Springer-Verlag, 2003.

[120] A.-R. Sadeghi, J. Shokrollahi, and C. Wachsmann (editors). Identification of Requirements and Constraints. *D3.2, Horst Gortz Institute for IT-Security, Ruhr-University, Bochum, for Signal Processing in the Encrypted Domain (SPEED) Project, IST-2006-034238, Information Society Technologies*, 2007. Available at: `www.speedproject.eu`.

[121] M. Schmucker and P. Ebinger. Promotional and Commercial Content Distribution based on a Legal and Trusted P2P Framework. In *Proceedings of the Seventh IEEE International Conference on E-Commerce Technology (CEC '05)*, pages 439–442. IEEE Computer Society Press, 2008.

[122] C. Schnorr. Efficient Identification and Signatures for Smart Cards. In G. Brassard, editor, *Advances in Cryptology - CRYPTO 1989*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer-Verlag, 1990.

[123] M-H Shao. A Privacy-Preserving Buyer-Seller Watermarking Protocol with Semi-trust Third Party. In C. Lambrinoudakis, G. Pernul, and A M. Tjoa, editors, *4th International Conference on Trust, Privacy & Security in Digital Business (TrustBus 2007)*, volume 4657 of *Lecture Notes in Computer Science*, pages 44–53. Springer-Verlag, 2007.

[124] D. R. Stinson. *Cryptography Theory and Practice*. Third Edition, Series on Discrete Mathematics and Its Applications, Chapman & Hall/CRC, 2006.

[125] J. K. Su, F. hartung, and B. Girod. Digital watermarking of text, image, and video documents. *Computers & Graphics, Elsevier*, 22(6):687–695, 1998.

[126] Trusted Computing Group (TCG). Trusted computing group website, accessed February 2009. Available at: `www.trustedcomputinggroup.org`.

[127] A. Tomlinson. Application and Business Security: Payment and e–commerce applications. *Lecture Notes IY5601, MSc. of Information Security, Information Security Group, Royal Holloway, University of London*, 2008.

[128] A. Tomlinson. *Security For Video Broadcasting*. Springer, 2008.

[129] P. Tomsich and S. Katzenbeisser. Towards a robust and de-centralized digital watermarking infrastructure for the protection of intellectual property. In K. Bauknecht, S. Kumar Madria, and G. Pernul, editors, *Electronic Commerce and Web Technologies - EC-WEB 2000*, volume 1875 of *Lecture Notes in Computer Science*, pages 38–47. Springer-Verlag, 2000.

[130] Trusted Computing Group (TCG). TCG Specification Architecture Overview. Version 1.2, The Trusted Computing Group, 2004.

[131] Visa and Inc. Visa payment solutions, accessed February 2009. Available at: `http://www.visa.com`.

[132] Neal R. Wagner. Fingerprinting. In *IEEE Symposium on Security and Privacy*, pages 18–22, 1983.

[133] T. Wiegand, G. J. Sullivan, G. Bjntegaard, and A. Luthra. Overview of the H.264/AVC Video Coding Standard. *IEEE Transactions On Circuits and Systems for Video Technology*, 13(7):560 – 576, 2003.

[134] D. M. Williams, H. Treharne, and Anthony T. S. Ho. Using a Formal Analysis Technique to Identify an Unbinding Attack on a Buyer-Seller Watermarking Protocol. In *Proceedings of the 10th ACM Workshop on Multimedia and Security (MM & Sec 2008)*. ACM, 2008.

[135] D. M. Williams, H. Treharne, Anthony T. S. Ho, and A. Walker. Formal Analysis of Two Buyer-Seller Watermarking Protocols. In H. J. Kim, S. Katzenbeisser, and Anthony T. S. Ho, editors, *To appear in Digital Watermarking, Seventh International Workshop, IWDW 2008*, Lecture Notes in Computer Science. Springer-Verlag, 2008.

[136] Y. Wu. Security Flaws in Kuribayashi-Tanaka Fingerprinting Protocol. In *IEEE International Conference on Communications - ICC 2007*, pages 1317 – 1322, 2007.

[137] Y. Wu and H. Pang. A Lightweight Buyer-Seller Watermarking Protocol. *Advances in Multimedia*, 2008:905065, 7 pages, 2008. `doi:10.1155/2008/905065`.

[138] S. Yong and S.-H. Lee. An Efficient Fingerprinting Scheme with Symmetric and Commutative Encryption. In M.Barni, I. J. Cox, T. Kalker, and H. J.

Kim, editors, *Digital Watermarking, 4th International Workshop - IWDW 2005*, volume 3710 of *Lecture Notes in Computer Science*, page p. 54. Springer-Verlag, 2005.

[139] J. Zhang, W. Kou, and K. Fan. Secure Buyer-Seller Watermarking Protocol. *IEE Proceedings - Information Security*, 153(1):15–18, 2006.

[140] Q. Zhang, K. Markantonakis, and K. Mayes. A Practical Fair-exchange e-payment Protocol for Anonymous Purchase and Physical Delivery. In *Proceeding of the 4th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA-06)*, 2006.