Mobile Agent Security

Mousa M. Alfalayleh

A thesis submitted in fulfillment of the requirements for the degree of
Doctor of Philosophy

School of Electrical Engineering & Computer Science
University of Newcastle
Callaghan
New South Wales 2308
Australia

July 2009
The thesis contains no material which has been accepted for the award of any other degree or diploma in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. I give consent to this copy of my thesis, when deposited in the University Library, being made available for loan and photocopying subject to the provisions of the Copyright Act 1968.

(Signed) ____________________________

Mousa M. Alfalayleh
© Copyright 2009
Mousa M. Alfalayleh
I have the honour to be supervised by a great person, who offered me her trust, opportunity and assistance during my PhD study. I am very grateful to my supervisor Ljiljana Brankovic, who believed that I could do my PhD. My skills as a researcher have been well trained by her.

I would like also to acknowledge the support of all the staff of The School of Electrical Engineering and Computer Science at my University, especially my colleges Helen and Zahid.

I lived in Newcastle during my PhD studies. During this Period I met many wonderful people including Suzan, Jenny, and Andrew who made me feel home. I will always remember the wonderful time spent together with my accommodation mates Haitham and Shadi.

My grateful thanks to my parents who supported me in all of what I have done in my life. I am very grateful to all of my sisters especially Muntaha, who always encourage me to continue my higher education. I would like to extend same acknowledgment to my brothers especially Haitham, the eldest one for his invaluable advises.

Finally, I would like to highly appreciate the generous financial support that is introduced by Ljiljana's research grant and by Al-Hussein Bin Talal University in Jordan.
This thesis is gratefully dedicated to
my parents
sisters and brothers
for their patience, their unwavering support and their faith.
List of publications arising from this thesis


List of other publications

Abstract

Mobile agents are programs that travel autonomously through a computer network in order to perform some computation or gather information on behalf of a human user or an application. In the last several years numerous applications of mobile agents have emerged, including e-commerce. However, mobile agent paradigm introduces a number of security threats both to the agents themselves and to the servers that they visit. This thesis gives an overview of the main security issues related to the mobile agent paradigm.

The first part of the thesis focuses on security of mobile agent itself. In this part, we propose a new coupling technique based on trust as a social control to work together with existing traditional security mechanisms. It relies on the “reputation” of the hosts in the itinerary and ensures that the agent succeeds in accomplishing its task with a high probability.

Due to the fact that the coupling technique requires an agent’s itinerary to be known in advance, we introduce two new concepts: a “Scout mobile agent”, whose primary purpose is to determine the itinerary required for accomplishing a given task, and a “Routed mobile agent”, which operates with an itinerary known in advance. This enables the Routed agent to incorporate various security mechanisms, including our new coupling technique.

Our Routed agent technique is also applicable independently of the Scout agent, whenever the itinerary and the trust values of the platforms in the itinerary are known. We also proposed a Petrol Station as an entity that would provide a service to other entities, in the form of certifying mobile agents and equipping them with safe itinerary based on trust score and applying the Routed agent.

In the second part of the thesis, we shed some light on the security of mobile agent platforms as it is considered more critical than the security of agents. In particular, we consider a scenario where a platform hosts a database containing confidential individual information and allows mobile agents to query the database. This mobile agent may be behave maliciously which is similar to an intruder in the Statistical Disclosure Control (SDC), where measuring disclosure risk is still considered as a difficult and only partly solved problem [111]. We introduce a scenario that is not adequately
covered by any of the previous discloser risk measures. Shannon’s entropy can be considered a satisfactory measure for the disclosure risk that is related to the exact compromise. However, in the case of approximate compromise, we argue that Shannon’s entropy does not express precisely the intruder’s knowledge about a particular confidential value. We introduce a novel disclosure risk measure that is based on Shannon’s entropy but covers both exact and approximate compromise. The main advantage of our measure over previously proposed measures that it gives careful consideration to the attribute values in addition to the probabilities with which the values occur. We use a dynamic programming algorithm to calculate the disclosure risk for various levels of approximate compromise. Importantly, our proposed measure is independent of the applied SDC technique.

Finally, we show how this measure can be used to evaluate the security mechanisms for protecting privacy in statistical databases and data mining. We conduct extensive experiments and apply our proposed security measure to three different datasets protected by three different SDC techniques, namely Sampling, Query Restriction, and Noise Addition.
Contents

1 Introduction 1

2 Security Research in The Mobile Agent Paradigm 8
   2.1 Security Issues in Mobile Agent Paradigm .................. 8
   2.2 Security of Platforms ...................................... 11
   2.2.1 Sandboxing .............................................. 11
   2.2.2 Code Signing ............................................ 13
   2.2.3 Code Signing and Sandboxing Combined .................. 14
   2.2.4 Proof-Carrying Code .................................... 15
   2.2.5 State Appraisal .......................................... 16
   2.2.6 Path Histories .......................................... 17
   2.3 Security of Mobile Agents ................................ 18
   2.3.1 Co-Operating Agents .................................... 18
   2.3.2 Execution Tracing ....................................... 19
   2.3.3 Environmental Key Generation ............................ 21
   2.3.4 Non-Interactive Computing with Encrypted Functions 22
   2.3.5 Obfuscated Code ......................................... 23
   2.3.6 Partial Result Encapsulation ............................ 24
   2.3.7 Detection of Denial of Services ......................... 26

3 Trust and Mobile Agents 28
   3.1 Trust Definition ........................................... 29
   3.2 Trust Properties ........................................... 30
   3.3 Trust Classification ....................................... 33
   3.4 The Coupling Technique .................................. 37

4 Scout and Routed Mobile Agents 44
   4.1 Security Requirements and Protection Techniques .......... 46
4.2 Our Approach: Scout and Routed Mobile Agents .......... 47
   4.2.1 Calculating the trust ................................ 49
   4.2.2 The filtration process ................................. 50
4.3 Framework for Scout Mobile Agent ......................... 51
   4.3.1 Framework ............................................. 52
   4.3.2 A simple example ...................................... 53
   4.3.3 Experiments ........................................... 55
   4.3.4 Improving the performance ............................ 56
4.4 Petrol Stations .............................................. 57
   4.4.1 Assumptions ........................................... 57
   4.4.2 What is a Petrol Station? ............................. 58
   4.4.3 Suggested responsibilities for Petrol Station ........ 59
   4.4.4 A framework for Petrol Station ..................... 60

5 A Novel Entropy-Based Measure of Disclosure Risk .......... 64
   5.1 Introduction to Statistical Disclosure Control .......... 66
   5.2 Related Work on Disclosure Risk Measures .............. 68
   5.3 Introducing The Problem ................................ 73
   5.4 A Novel Entropy-Based Measure .......................... 78
      5.4.1 A numerical example ................................ 81
   5.5 An Algorithm to Compute $H(\varepsilon)$ .................. 84

6 Comparative Study of SDC Techniques ....................... 90
   6.1 Background on The Chosen SDC Techniques ............. 90
      6.1.1 Sampling ............................................ 91
      6.1.2 Query Restriction .................................. 93
      6.1.3 Noise Addition ..................................... 95
   6.2 The Used Datasets ....................................... 97
      6.2.1 PUMS dataset ...................................... 97
      6.2.2 WBC dataset ....................................... 98
      6.2.3 WINE dataset ...................................... 98
   6.3 The Experiments: Description and Implementation ...... 99
      6.3.1 Sampling ............................................ 99
      6.3.2 Query Restriction .................................. 103
      6.3.3 Noise Addition ..................................... 106
   6.4 The Experiments: PUMS Dataset Results ................ 111
      6.4.1 Sampling ............................................ 111
## CONTENTS

6.4.2 Query restriction ........................................... 118
6.4.3 Noise addition ............................................. 125

6.5 The Experiments: Other Dataset Results .......................... 132
6.5.1 Sampling: WBC dataset ..................................... 132
6.5.2 Sampling: WINE dataset ................................... 136
6.5.3 Query restriction: WBC dataset ............................. 139
6.5.4 Query restriction: WINE dataset ............................ 142
6.5.5 Noise addition: WBC dataset ............................... 145
6.5.6 Noise addition: WINE dataset ............................. 148

7 Conclusion .................................................. 151

A A Numerical Example ........................................ 155

B Extra Experiment Results ..................................... 161

B.1 Sampling: PUMS Dataset ..................................... 162
B.2 Sampling: WBC Dataset ..................................... 171
B.3 Sampling: WINE Dataset .................................... 174
B.4 Query Restriction: PUMS Dataset ........................... 177
B.5 Query Restriction: WBC Dataset ............................. 186
B.6 Query Restriction: WINE Dataset ............................ 189
B.7 Noise Addition: PUMS Dataset ............................... 192
B.8 Noise Addition: WBC Dataset ............................... 201
B.9 Noise Addition: WINE Dataset ............................. 204
List of Figures

3.1 Trust Properties: Context-Dependent ................................ 30
3.2 Trust Properties: Conditionally Transitive .......................... 31
3.3 Trust Properties: Asymmetry ......................................... 32
3.4 Trust Properties: Dynamic Nature .................................... 32
3.5 Trust Properties: Subjective ........................................... 33
3.6 Trust Classification: Trust by Execution Platform .................. 34
3.7 Trust Classification: Trust by Mobile Agent ........................... 34
3.8 Trust Classification: Trust by Home Platform ....................... 35
3.9 Trust Classification: Trust in Trusted Third Party - Certification Trust .................................................. 35
3.10 Trust Classification: Trust in Trusted Third Party - Identity Trust ............................................................... 35
3.11 Trust Classification: Trust in an entity itself ........................... 36
3.12 Ordering platforms with a Coupling technique: (a) Even, (b) Odd ................................................................. 38

4.1 An Activity Diagram with role-annotations for Scout and Routed mobile agents’ framework ................................. 51
4.2 Second round itinerary ..................................................... 55
4.3 Fitting Petrol Station, PS, into the Routed agent scenario ......... 58
4.4 An Activity Diagram with role-annotations for Petrol Station’s framework ..................................................... 61

5.1 A Roulette Table Layout .................................................. 74
5.2 Roulette Bets .............................................................. 75
5.3 Our Security Measure: for 5,6 and 1,10 ................................. 80
5.4 Our security measure for the numerical example ..................... 85
5.5 Our Security Measure: for the equally likely values [1,10] ... 88
5.6 Our Security Measure: for the Roulette example. ............ 88
5.7 Our Security Measure: for (77K, 80K) and (50K, 107K). ... 89

6.1 Binomial PDF for $n = 24$ and $p = 0.5$ .................... 108
6.2 SDC: Sampling, Dataset: PUMS, Sampling Factor: 0.1. Distribution of average initial entropy for 30 samples when SK is 1, 2, 3, or 6 attributes. .................................................. 112
6.3 SDC: Sampling, Dataset: PUMS, Sampling Factor: 0.1. Distribution of average entropy for 30 samples when SK is 1, 2, 3, or 6 attributes. .................................................. 113
6.4 SDC: Sampling, Dataset: PUMS. (average $H_0$ and average entropy) vs intruder’s SK (number of known attributes) ... 114
6.5 SDC: Sampling, Dataset: PUMS. (average $H_0$ and average entropy) vs sample size ................................................. 116
6.6 SDC: Sampling, Dataset: PUMS, Sampling Factor: 0.5. Distribution of average initial entropy and average entropy for different 1, 2, and 3 attribute combinations .......................... 118
6.7 SDC: Query Restriction, Dataset: PUMS, Query Set Size: 2. Distribution of average initial entropy for 30 shuffled datasets when SK is 1, 2, 3, or 6 attributes. ................................. 119
6.8 SDC: Query Restriction, Dataset: PUMS, Query Set Size: 2. Distribution of average entropy for 30 shuffled datasets when SK is 1, 2, 3, or 6 attributes. ................................. 120
6.9 SDC: Query Restriction, Dataset: PUMS. ($H_0$, Area) vs Number of Known Attributes .............................................. 122
6.10 SDC: Query Restriction, Dataset: PUMS. ($H_0$, Area) vs Query Set Size ................................................................. 123
6.11 SDC: Query Restriction, Dataset: PUMS, Query Set Size: 2. Distribution of the average initial entropy and the average entropy for different 1, 2 attribute combinations ............................ 125
6.12 SDC: Noise Addition, Dataset: PUMS, Noise Amount: 100%. Distribution of average initial entropy for 30 perturbed files when SK is 1, 2, 3, or 6 attributes. ................................. 126
6.13 SDC: Noise Addition, Dataset: PUMS, Noise Amount: 100%.
   Distribution of average entropy for 30 perturbed files when
   SK is 1, 2, 3, or 6 attributes. ........................................ 127
6.14 SDC: Noise Addition, Dataset: PUMS. (average $H_0$ and ave-
   rage entropy) vs intruder’s SK (number of known attributes 129
6.15 SDC: Noise Addition, Dataset: PUMS. (average $H_0$ and av-
   erage entropy) vs Noise Amount ................................. 130
6.16 SDC: Noise Addition, Dataset: PUMS, Noise Amount: 100%.
   Distribution of the average initial entropy and the average
   entropy for different 1, 2 attribute combinations .......... 132
6.17 SDC: Sampling, Dataset: WBC. (average $H_0$ and average
   entropy) vs intruder’s SK (number of known attributes) . 134
6.18 SDC: Sampling, Dataset: WBC. (average $H_0$ and average
   entropy) vs sample size ............................................. 135
6.19 SDC: Sampling, Dataset: WINE. (average $H_0$ and average
   entropy) vs intruder’s SK (number of known attributes) . 137
6.20 SDC: Sampling, Dataset: WINE. (average $H_0$ and average
   entropy) vs sample size ............................................. 138
6.21 SDC: Query Restriction, Dataset: WBC. (average $H_0$ and av-
   erage entropy) vs intruder’s SK (number of known attributes) 140
6.22 SDC: Query Restriction, Dataset: WBC. (average $H_0$ and
   average entropy) vs Query Set Size ........................... 141
6.23 SDC: Query Restriction, Dataset: WINE. (average $H_0$ and
   average entropy) vs intruder’s SK (number of known attributes) 143
6.24 SDC: Query Restriction, Dataset: WINE. (average $H_0$ and
   average entropy) vs Query Set Size ........................... 144
6.25 SDC: Noise Addition, Dataset: WBC. (average $H_0$ and aver-
   age entropy) vs intruder’s SK (number of known attributes) 146
6.26 SDC: Noise Addition, Dataset: WBC. (average $H_0$ and aver-
   age entropy) vs Noise Amount ................................. 147
6.27 SDC: Noise Addition, Dataset: WINE. (average $H_0$ and av-
   erage entropy) vs intruder’s SK (number of known attributes) 149
6.28 SDC: Noise Addition, Dataset: WINE. (average $H_0$ and av-
   erage entropy) vs Noise Amount ................................. 150
B.1 SDC: Sampling, Dataset: PUMS, Sampling Factor: 0.05. Distribution of average initial entropy for 30 samples when SK is 1, 2, 3, 4, 5, or 6 attributes. ................................. 162
B.2 SDC: Sampling, Dataset: PUMS, Sampling Factor: 0.05. Distribution of average entropy for 30 samples when SK is 1, 2, 3, 4, 5, or 6 attributes. ................................. 163
B.3 SDC: Sampling, Dataset: PUMS, Sampling Factor: 0.2. Distribution of average initial entropy for 30 samples when SK is 1, 2, 3, 4, 5, or 6 attributes. ................................. 164
B.4 SDC: Sampling, Dataset: PUMS, Sampling Factor: 0.2. Distribution of average entropy for 30 samples when SK is 1, 2, 3, 4, 5, or 6 attributes. ................................. 165
B.5 SDC: Sampling, Dataset: PUMS, Sampling Factor: 0.5. Distribution of average initial entropy for 30 samples when SK is 1, 2, 3, 4, 5, or 6 attributes. ................................. 166
B.6 SDC: Sampling, Dataset: PUMS, Sampling Factor: 0.5. Distribution of average entropy for 30 samples when SK is 1, 2, 3, 4, 5, or 6 attributes. ................................. 167
B.7 SDC: Sampling, Dataset: PUMS, Sampling Factor: 0.05. Distribution of average initial entropy and average entropy for different 1, and 2 attribute combinations ............................ 168
B.8 SDC: Sampling, Dataset: PUMS, Sampling Factor: 0.1. Distribution of average initial entropy and average entropy for different 1, and 2 attribute combinations ............................ 169
B.9 SDC: Sampling, Dataset: PUMS, Sampling Factor: 0.2. Distribution of average initial entropy and average entropy for different 1, and 2 attribute combinations ............................ 170
B.10 SDC: Sampling, Dataset: WBC, Sampling Factor: 0.1. Distribution of average initial entropy for 30 samples when SK is 1, 2, 3, 5, 7, or 9 attributes. ........................................... 171
B.11 SDC: Sampling, Dataset: WBC, Sampling Factor: 0.1. Distribution of average entropy for 30 samples when SK is 1, 2, 3, 5, 7, or 9 attributes. ........................................... 172
B.12 SDC: Sampling, Dataset: WBC, Sampling Factor: 0.1. Distribution of average initial entropy and average entropy for different 1, and 2 attribute combinations ............................ 173
B.13 SDC: Sampling, Dataset: WINE, Sampling Factor: 0.1. Distribution of average initial entropy for 30 samples when SK is 1, 2, 3, 5, 7, 9, 11, or 13 attributes.

B.14 SDC: Sampling, Dataset: WINE, Sampling Factor: 0.1. Distribution of average entropy for 30 samples when SK is 1, 2, 3, 5, 7, 9, 11, or 13 attributes.

B.15 SDC: Sampling, Dataset: WINE, Sampling Factor: 0.1. Distribution of average initial entropy and average entropy for different 1, and 2 attribute combinations.

B.16 SDC: Query Restriction, Dataset: PUMS, Query Set Size: 4. Distribution of average initial entropy for 30 shuffled datasets when SK is 1, 2, 3, 4, 5, or 6 attributes.

B.17 SDC: Query Restriction, Dataset: PUMS, Query Set Size: 4. Distribution of average entropy for 30 shuffled datasets when SK is 1, 2, 3, 4, 5, or 6 attributes.

B.18 SDC: Query Restriction, Dataset: PUMS, Query Set Size: 8. Distribution of average initial entropy for 30 shuffled datasets when SK is 1, 2, 3, 4, 5, or 6 attributes.

B.19 SDC: Query Restriction, Dataset: PUMS, Query Set Size: 8. Distribution of average entropy for 30 shuffled datasets when SK is 1, 2, 3, 4, 5, or 6 attributes.

B.20 SDC: Query Restriction, Dataset: PUMS, Query Set Size: 16. Distribution of average initial entropy for 30 shuffled datasets when SK is 1, 2, 3, 4, 5, or 6 attributes.

B.21 SDC: Query Restriction, Dataset: PUMS, Query Set Size: 16. Distribution of average entropy for 30 shuffled datasets when SK is 1, 2, 3, 4, 5, or 6 attributes.

B.22 SDC: Query Restriction, Dataset: PUMS, Query Set Size: 4. Distribution of average initial entropy and average entropy for different 1, and 2 attribute combinations.

B.23 SDC: Query Restriction, Dataset: PUMS, Query Set Size: 8. Distribution of average initial entropy and average entropy for different 1, and 2 attribute combinations.

B.24 SDC: Query Restriction, Dataset: PUMS, Query Set Size: 16. Distribution of average initial entropy and average entropy for different 1, and 2 attribute combinations.
Distribution of average initial entropy for 30 shuffled datasets
when SK is 1, 2, 3, 5, 7, or 9 attributes. . . . . . . . . . . . . . . . . 186

B.26 SDC: Query Restriction, Dataset: WBC, Query Set Size: 2.
Distribution of average entropy for 30 shuffled datasets when
SK is 1, 2, 3, 5, 7, or 9 attributes. . . . . . . . . . . . . . . . . . . . . 187

Distribution of the average initial entropy and the average
entropy for different 1, 2 attribute combinations. . . . . . . . 188

Distribution of average initial entropy for 30 shuffled datasets
when SK is 1, 2, 3, 5, 7, 9, 11, or 13 attributes. . . . . . . . . . . . . 189

B.29 SDC: Query Restriction, Dataset: WINE, Query Set Size: 2.
Distribution of average entropy for 30 shuffled datasets when
SK is 1, 2, 3, 5, 7, 9, 11, or 13 attributes. . . . . . . . . . . . . . . . . . 190

B.30 SDC: Query Restriction, Dataset: WINE, Query Set Size: 2.
Distribution of the average initial entropy and the average
entropy for different 1, 2 attribute combinations. . . . . . . . . . 191

B.31 SDC: Noise Addition, Dataset: PUMS, Noise Amount: 75%.
Distribution of average initial entropy for 30 perturbed files
when SK is 1, 2, 3, 4, 5, or 6 attributes. . . . . . . . . . . . . . . . . 192

B.32 SDC: Noise Addition, Dataset: PUMS, Noise Amount: 75%.
Distribution of average entropy for 30 perturbed files when
SK is 1, 2, 3, 4, 5, or 6 attributes. . . . . . . . . . . . . . . . . . . . . 193

B.33 SDC: Noise Addition, Dataset: PUMS, Noise Amount: 50%.
Distribution of average initial entropy for 30 perturbed files
when SK is 1, 2, 3, 4, 5, or 6 attributes. . . . . . . . . . . . . . . . . 194

B.34 SDC: Noise Addition, Dataset: PUMS, Noise Amount: 50%.
Distribution of average entropy for 30 perturbed files when
SK is 1, 2, 3, 4, 5, or 6 attributes. . . . . . . . . . . . . . . . . . . . . 195

B.35 SDC: Noise Addition, Dataset: PUMS, Noise Amount: 25%.
Distribution of average initial entropy for 30 perturbed files
when SK is 1, 2, 3, 4, 5, or 6 attributes. . . . . . . . . . . . . . . . . 196

B.36 SDC: Noise Addition, Dataset: PUMS, Noise Amount: 25%.
Distribution of average entropy for 30 perturbed files when
SK is 1, 2, 3, 4, 5, or 6 attributes. . . . . . . . . . . . . . . . . . . . . 197
B.37 SDC: Noise Addition, Dataset: PUMS, Noise Amount: 75%.
Distribution of average initial entropy and average entropy for
different 1, and 2 attribute combinations .......................... 198
B.38 SDC: Noise Addition, Dataset: PUMS, Noise Amount: 50%.
Distribution of average initial entropy and average entropy for
different 1, and 2 attribute combinations .......................... 199
B.39 SDC: Noise Addition, Dataset: PUMS, Noise Amount: 25%.
Distribution of average initial entropy and average entropy for
different 1, and 2 attribute combinations .......................... 200
B.40 SDC: Noise Addition, Dataset: WBC, Noise Amount: 100%.
Distribution of average initial entropy for 30 perturbed files
when SK is 1, 2, 3, 5, 7, or 9 attributes ................................. 201
B.41 SDC: Noise Addition, Dataset: WBC, Noise Amount: 100%.
Distribution of average entropy for 30 perturbed files when
SK is 1, 2, 3, 5, 7, or 9 attributes .......................................... 202
B.42 SDC: Noise Addition, Dataset: WBC, Noise Amount: 100%.
Distribution of the average initial entropy and the average
entropy for different 1, 2 attribute combinations ..................... 203
B.43 SDC: Noise Addition, Dataset: WINE, Noise Amount: 100%.
Distribution of average initial entropy for 30 perturbed files
when SK is 1, 2, 3, 5, 7, 9, 11, or 13 attributes ....................... 204
B.44 SDC: Noise Addition, Dataset: WINE, Noise Amount: 100%.
Distribution of average entropy for 30 perturbed files when
SK is 1, 2, 3, 5, 7, 9, 11, or 13 attributes ............................. 205
B.45 SDC: Noise Addition, Dataset: WINE, Noise Amount: 100%.
Distribution of the average initial entropy and the average
entropy for different 1, 2 attribute combinations ..................... 206
List of Tables

4.1 First Round Itinerary Results ........................................ 54
4.2 Experiment Results ...................................................... 56

5.1 University Database: AcademicStaff Table ...................... 66
5.2 Example of $k$-anonymity, $l$-diversity: Initial Microdata .... 72
5.3 Example of $k$-anonymity, $l$-diversity: 4-anonymous Released
Micradata ................................................................. 73
5.4 Various Roulette bets and their payoffs .......................... 75
5.5 Example 1 .............................................................. 77
5.6 Example 2 .............................................................. 78
5.7 Structure of “filling a table” by our algorithm ................. 85
5.8 A table filled by our algorithm for the numerical example .. 87

6.1 The selected attributes from PUMS dataset ................... 97
6.2 The selected attributes from WBC dataset ..................... 98
6.3 The selected attributes from WINE dataset .................... 99
6.4 Original Dataset ....................................................... 100
6.5 Sample Dataset ....................................................... 100
List of Algorithms

1 A dynamic programming algorithm to compute $H(\varepsilon)$  . . . . . 87
2 Finding $d_i$ and their corresponding $p_i$  . . . . . . . . . . . . . 105
3 Analysing noise  . . . . . . . . . . . . . . . . . . . . . . . . . . . 110
Chapter 1

Introduction

Recent years have brought some fundamental changes into distributed and client-server computer systems. While in the past these systems had a static hardware configuration, advances in cellular phone technology, wireless networks and mobile Internet access introduced a dynamic capability to the systems. Similarly, in the past, software applications were bound to particular nodes in a computer network. This reality has changed with the appearance of mobile code technologies, which include weakly mobile (e.g., Java) and strongly mobile, i.e., mobile agents [108]. Mobile agents are programs that travel autonomously through a computer network in order to perform some computation or gather information on behalf of a human user or an application [58]. As such, they provide an appealing alternative to client-server architecture for many applications.

A mobile agent system consists of two main components: a mobile agent and an agent platform. Home platform is the one that creates the mobile agent and sends it through the network to roam from platform to platform (server, host). When the mobile agent finishes its task, it either terminates or returns to its home platform to report to the user. Each agent platform offers a service to mobile agents that visit it and provides the computational environment, as well as other resources required for mobile agent execution. An agent typically carries some information, which includes a code to be executed, collected data, and an execution state.
Among the most important characteristics of mobile agents are mobility and autonomy. Mobility allows a mobile agent to travel among different agent platforms. Autonomy enables a mobile agent to exercise control over its own actions and its internal state; for example, an agent is freely choosing the next platform to visit in its itinerary. The applications of mobile agent technology are abundant [68, 115] and include electronic commerce, personal assistance, distributed information search and retrieval, network management, distributed industrial automation and control systems, military command and control, and parallel processing [58].

There are numerous advantages of using the mobile agent paradigm rather than conventional paradigms such as client-server based technologies [58]. The mobile agent paradigm improves fault tolerance and concurrency and assists operating in heterogeneous environments. It also reduces the network load as users do not need to have a network connection established while their agents are performing operations on different servers in the network.

Unfortunately, mobile agent technology has some drawbacks, primarily in the area of security. Mobile agent paradigm introduces new security threats both to the agents themselves and to the servers that they visit. Current research efforts in the area of mobile agent security adopt two different points of view. Firstly, from the platform perspective, there is a need to protect hosts from malicious mobile agents that are visiting them and consuming their resources. Secondly, from the mobile agent point of view, it is necessary to protect the agent from malicious hosts. Clearly, the security of mobile agent platforms is more critical than the security of agents. However, it is important to note that these two perspectives are dependent on each other. For example, the level of trust that an executing platform has in the agent depends not only on the agent’s home platform but also on all the platforms previously visited by the agent [78]. Each one of these platforms had an opportunity to maliciously modify the agent in such a way so as to harm subsequently visited platforms. Thus the increased security of the agent implies the increased security of platforms.

Protecting security of mobile agents is a difficult problem and, despite a number of proposed techniques [52], it remains largely unresolved. There
are, however, a few restrictions and/or relaxations that could be imposed on the mobile agent paradigm and that would make the security problem much less complicated [78, 57]. For example, if a mobile agent is restricted to a network of mutually trusted platforms, then most security issues are alleviated [92]. However, this idea challenges one of the basic assumptions of the mobile agent paradigms, i.e., an open and dynamic network environment where a new platform can be added at any time [92]. As a second example, consider a mobile agent that interactively communicates with its home platform. This can surely diminish many security threats as the home platform can check the integrity of the agent after visit to each platform, choose the next platform to be visited and even modify the agent so as to provide maximal protection. However, this requires the home platform to be connected to the network and available to the agent and thus diminishes one of the main advantages of the mobile agent paradigm [92].

Finally, if the itinerary of the agent were known in advance, including both the list of platforms and the order in which they are visited, the task of securing the mobile agent suddenly becomes a great deal easier [78, 57]. It is now possible for the home platform to make use of cryptographic primitives, such as encryption and digital signatures, as all platforms involved in the process are known in advance. Importantly, this would reduce the risk and consequently increase the trust in the agent. Such an agent could also incorporate techniques for protection against Denial of Service (DoS) attacks [28]. A serious problem with determining the itinerary in advance is that this jeopardizes the autonomy of the agent, which is another one of the basic assumptions of the mobile agent paradigm.

Traditional security mechanisms typically rely on authentication, access control, and cryptographic techniques [83, 85]. These traditional security solutions are unable to fulfill various security requirements emerging in open and distributed online environment. Therefore, there is a need for new security solutions that can accommodate different aspects of open system architecture including the social behavior of its entities and can work side by side with traditional security mechanisms [85]. Rasmussen and Jansson [83] defined social control as “a group behaviour that indirectly forces the group members to behave in a particular way”. In this context, it is
assumed that malicious entities exist in a community. The goal is to identify these entities and prevent them from causing harm to other entities in the community. Social control is considered a soft security solution [83]. Trust is an example of social control mechanisms and is regarded as a new dimension of security in virtual online communities. A mobile agent system is an example of an online environment.

The above discussion focuses on the security of the mobile agent. We next turn our attention to the security of the platform. In particular, we consider a scenario where a platform hosts a database containing confidential individual information and allows mobile agents to query the database. For example, a Real Estate Agency (REA) maintains a database containing information about recently sold houses that may be made available to public for a fee. Suppose that Alice wants to invest in real estate and is interested in recent house sale prices in a particular region. Alice then sends her mobile agent to roam the Internet and collect the information from various real estate agencies. Suppose, however, that Alice’s neighbor Bob has just sold a property in that region. Alice is curious and interested to learn what price Bob achieved for his house but she knows that Bob is a private person and would prefer not to share such information with his neighbours. In a way of a stronger example, suppose that Alice is not interested in real estate investment at all but is rather using it as a way to infer Bob’s personal information. In other words, a malicious mobile agent might purposely extract a confidential individual information and thus compromise privacy of individuals. In the light of this example, REA may not wish to disclose individual house prices but only their aggregate values. Consequently, REA must ensure that no sequence of queries asked from the database by the same party or a group of collaborating parties is sufficient to infer an individual house price and to link it to a particular address. This problem is known as a Statistical Disclosure Control (SDC) [111].

There are various techniques that can be used to alleviate this problem [5]. Unfortunately, none of the available techniques is able to solve the problem completely, due to its intrinsic contradictory nature. On one hand, one must keep the risk of individual value disclosure as low as possible. On the other hand, the utility (usefulness) of the database must remain high.
However, low risk implies low utility and high utility implies high risk. A good SDC measure aims at finding a right balance between the two. In order to achieve this balance, it is crucial to adequately measure both utility and disclosure risk. While measuring data utility has been well studied in the literature [111], measuring disclosure risk is still considered as a difficult and only partly solved problem.

In what follow, we introduce a brief summary about the contents of each chapter in this thesis and our main contributions:

Chapter 2 surveys the main issues in the security of mobile agents. Mobile agent paradigm introduces new security threats both to the agents themselves and to the servers that they visit. It is much more challenging to ensure the security of mobile agents than the security of agent platforms. We discuss the security threats and requirements that need to be met in order to alleviate those threats. Finally, we present different security techniques proposed in the literature to keep mobile agents and platforms secure.

Chapter 3 looks at the concept of trust within mobile agent context. We first define trust and then explore its properties and classification in the context of mobile agent paradigm. Additionally, we introduce a security solution that is based on the trust dimension as a social control to work together with traditional existing security mechanisms. We call the proposed solution “coupling technique”. It aims to increase the probability that the agent successfully accomplishes its task by partitioning the itinerary into the pairs, or “couples” of platforms. The coupling radically increases the overall probability that the agent will not be harmed, especially if the itinerary contains servers with a very low level of trust.

Chapter 4 proposes a Scout and Routed Agent approach that balances security and autonomy of a mobile agent. Our approach relies on two copies of a mobile agent to do the task instead of one. We first send the Scout agent (the first copy) to determine the itinerary and return back to its originator. The originator then filters the path, determines the order of the platforms, and incorporates extra security measures in the second copy of the agent, the so-called Routed agent. Our approach is very suitable for e-commerce applications such as shopping mobile agent that collects prices
and offers. The Scout/Routed agent technique combines the advantages of mobile agent autonomy with increased security, at the expense of the one-time involvement of the home platform. Our Routed agent technique is also applicable independently of the Scout agent, whenever the itinerary and the trust values of the platforms in the itinerary are known. Moreover, we introduce the idea of Petrol Station, a highly trusted third party that is distributed within the mobile agents’ community, to ensure that mobile agent visits only a network of mutually trusted platforms. To do so, Petrol Station maintains a database of different platforms and their trust scores. It identifies any malicious platform and prevents it from causing damage or harm to mobile agent by eliminating it from the agent’s itinerary. The Petrol Station follows our Routed agent technique to couple the platforms in the agent itinerary.

The previous chapters of the thesis focus on the security of the mobile agent. In Chapter 5 and 6, we turn our attention to the security of the platform. In Statistical Disclosure Control (SDC), measuring disclosure risk is still considered as a difficult and only partly solved problem [111]. In Chapter 5 we present related work on discloser risk measures and we introduce a scenario that is not adequately covered by any of the previous discloser risk measures. A database is said to be compromised if a confidential value is disclosed [71]. The compromise can be exact or approximate. In the exact compromise, an intruder learns the exact confidential value. Shannon’s entropy can be considered a satisfactory measure for the disclosure risk that is related to the exact compromise. However, in the approximate compromise, we argue that Shannon’s entropy does not express precisely the intruder’s knowledge about a particular confidential value. We introduce a novel disclosure risk measure that is based on entropy. We believe that this novel measure is satisfactory for data confidentiality. The main advantage of our measure over previously proposed measures that it gives careful consideration to the attribute values in addition to the probabilities with which the values occur. Additionally, our proposed measure is independent of the applied SDC technique. We propose a dynamic programming algorithm for calculating the disclosure risk.

In Chapter 6, we apply our proposed discloser risk measure to three
common SDC techniques, in particular Sampling, Query Restriction and Noise Addition, to show how this measure can be used to evaluate the security mechanisms for protecting privacy in statistical databases and data mining.

Chapter 7 concludes this thesis and suggests further work for the future.
Chapter 2

Security Research in The Mobile Agent Paradigm

This chapter gives an overview of the main solutions that have been described in the literature to keep the mobile agent platform and the agent itself protected from each other. The chapter deals with the security issues related to the mobile agent paradigm such as security threats and requirements. It gives an overview of the main solutions for keeping a mobile agent platform secure against a malicious mobile agent. Similarly, it presents a set of solutions for ensuring the security of mobile agents against illegitimate platforms.

2.1 Security Issues in Mobile Agent Paradigm

The mobile agent paradigm appeals to many specialists working in different applications. This is especially true for e-commerce applications, including stock markets and electronic auctions. Such applications involve dealing with vast amounts of money and thus users will hesitate to use mobile agents unless they feel that they are secure and can be trusted. Therefore, the security of mobile agents is an important issue that has triggered much research effort in order to find a suitable solution.
2.1 Security Issues in Mobile Agent Paradigm

One of the most valuable characteristics of mobile agents is their mobility that enables them to travel autonomously through the network. However, it is precisely because of this property that mobile agents are exposed to different types of attacks. We next present these attacks, together with those that are launched by agents to harm platforms.

Unauthorized Access. Malicious mobile agents can try to access the services and resources of the platform without adequate permissions. In order to thwart this attack, a mobile agent platform must have a security policy specifying the access rules applicable to various agents, and a mechanism to enforce the policy.

Masquerading. In this attack, a malicious agent assumes the identity of another agent in order to gain access to platform resources and services, or simply to cause mischief or even serious damage to the platform. Likewise, a platform can claim the identity of another platform in order to gain access to the mobile agent data. In both cases, the malicious agent or platform will not receive any blame for its potentially detrimental actions. Instead, the unsuspecting agent or platform whose identity was misused will be held responsible [52, 58].

Denial of Service. A malicious platform can cause harm to a visiting mobile agent by ignoring the agent’s request for services and resources that are available on the platform, by terminating the agent without notification, or by assigning continuous tasks to the agent so that it will never reach its goal. Likewise, a malicious agent may attempt to consume the resources of the platform, such as disk space or processing time, or delete important files or even the whole hard disk contents, thus causing harm to the platform and launching a denial of service attack against other visiting agents [52, 58].

Annoyance attack. Examples of this attack include opening many windows on the platform computer or making the computer beep repeatedly [58]. Such attacks may not represent a very serious problem to the platform, however they still need to be prevented.

Eavesdropping. In this attack, a malicious platform monitors the behavior of a mobile agent in order to extract sensitive information from it. This
is typically used when the mobile agent code and data are encrypted. Monitoring may include the identity of the entities that mobile agent is communicating with, and the types of services requested by the mobile agent [52, 58].

**Alteration.** In the alteration attack, a malicious platform tries to modify mobile agent information, by performing an insertion, deletion and/or alteration to the agent’s code, data, and execution state. Modifying the mobile agent execution code and state may result in the agent performing harmful actions to other platforms, including the agent’s home platform [52, 58].

We next explore the different security requirements that the mobile agent paradigm needs to satisfy.

**Confidentiality.** It is important to ensure that the information carried by a mobile agent or stored on a platform is accessible only to authorized parties. This is also the case for the communication among mobile agent paradigm components.

**Integrity.** It is essential to protect the mobile agent’s code, state, and data from being modified by unauthorized parties. This can be achieved either by preventing or by detecting unauthorized modifications.

**Availability.** Platforms typically face a huge demand for services and data. In the case that a platform cannot meet mobile agents’ demands, it should notify them in advance. Additionally, a platform must be able to afford a certain level of fault-tolerance and fault-recovery from unpredictable software and hardware failures [52].

**Accountability.** Platforms need to establish audit logs to keep track of all visiting mobile agents’ actions in order to keep them accountable for their actions. Audit logs are also necessary when the platform needs to recuperate from a security penetration or a system failure.

**Anonymity.** As mentioned above, platforms need to keep track of mobile agents’ actions for accountability purposes. However, platforms also
have to balance between their needs for audit logs and mobile agents’ needs to keep their actions private [52].

In the next two sections we present the existing techniques for protecting agents and platforms. These techniques fall into two categories: Prevention and detection. Prevention techniques are aimed at making it impossible for platforms and agents to successfully perform an attack. For example, a tamper-proof device can be used to execute an agent in a physically sealed environment. However, in the literature the term “prevention mechanism” is often used to denote a technique that makes it impossible to modify an agent in a meaningful way [18]. Examples of such techniques include “Environmental Key Generation” and “Computing with Encrypted Functions”. On the other hand, detection techniques aim at detecting the attacks. The “Co-Operating Agents” technique and “Execution Tracing” belong to this category.

2.2 Security of Platforms

The primary issue in the security of mobile agent systems is to protect mobile agent platforms against malicious attacks launched by the agents. This section presents a set of detection and prevention techniques for keeping the platform secure against a malicious mobile agent.

2.2.1 Sandboxing

Sandboxing [109] is a software technique used to protect mobile agent platform from malicious mobile agents. In an execution environment (platform), local code is executed with full permission and has access to crucial system resources. On the other hand, remote code, such as mobile agents and downloadable applets, is executed inside a restricted area called a “sandbox” [43, 42]. Restriction affects certain code operations [19] such as interacting with the local file system, opening a network connection, accessing system properties on the local system, and invoking programs on the local system. This ensures that a malicious mobile agent cannot cause any harm
to the execution environment that is running it. A Sandboxing mechanism enforces a fixed security policy for the execution of the remote code. The policy specifies the rules and restrictions that mobile agent code should confirm to. A mechanism is said to be secure if it properly implements a policy that is free of flaws and inconsistencies [88].

The most common implementation of Sandboxing is in the Java interpreter inside Java-enabled web browsers. A Java interpreter contains three main security components: ClassLoader, Verifier, and Security Manager [43, 44, 49, 70, 88]. The ClassLoader converts remote code into data structures that can be added to the local class hierarchy. Thus every remote class has a subtype of the ClassLoader class associated with it [88]. Before the remote code is loaded, the Verifier performs a set of security checks on it in order to guarantee that only legitimate Java code is executed [44, 49]. The remote code should be a valid virtual machine code, and it should not overflow or underflow the stack, or use registers improperly [70, 88]. Additionally, remote classes cannot overwrite local names and their operations are checked by the Security Manager before the execution. For example, in JDK 1.0.x, classes are labelled as local and remote classes. Local classes perform their operations without any restrictions while remote classes should first surrender to a checking process that implements the platform security policy. This is implemented within the Security Manager. If a remote class passes the verification, then it will be granted certain privileges to access system resources and continue executing its code. Otherwise, a security exception will be raised [43, 44, 49, 70, 88].

A problem with the Sandboxing technique is that a failure in any of the three above mentioned interrelated security parts may lead to a security violation. Suppose that a remote class is wrongly classified as a local class. Then this class will enjoy all the privileges of a local class. Consequently, the security policy may be violated [88]. A downside of the Sandboxing technique is that it increases the execution time of legitimate remote code [109] but this can be overcome by combining Code Signing and Sandboxing, as will be explained later.
2.2 Security of Platforms

2.2.2 Code Signing

The “Code Signing” technique ensures the integrity of the code downloaded from the Internet. It enables the platform to verify that the code has not been modified since it was signed by its creator. Code Signing cannot reveal what the code can do or guarantee that the code is in fact safe to run [1, 2].

Code Signing makes use of a digital signature and one-way hash function. A well-known implementation of code signing is Microsoft Authenticode, which is typically used for signing code such as ActiveX controls and Java applets [1].

Code Signing enables the verification of the code producer’s identity but it does not guarantee that they are trustworthy. The platform that runs mobile code maintains a list of trusted entities and checks the code against the list. If the code producer is on the list, it is assumed that they are trustworthy and that the code is safe. The code is then treated as local code and is given full privileges; otherwise the code will not run at all. This is known as a “black-and-white” policy [70, 88], as it only allows the platform to label programs as completely trusted or completely untrusted.

There are two main drawbacks of the Code Signing approach. Firstly, this technique assumes that all the entities on the trusted list are trustworthy and that they are incorruptible. Mobile code from such a producer is granted full privileges. If the mobile agent is malicious, it can use those privileges not only to directly cause harm to the executing platform but also to open a door for other malicious agents by changing the acceptance policy on the platform. Moreover, the affects of the malicious agent attack may only occur later, which makes it impossible to establish a connection between the attack and the attacker [88]. Such attacks are referred to as “delayed attacks”. Secondly, this technique is overly restrictive towards agents that are coming from untrustworthy entities, as they do not run at all. The approach that combines Code Signing and Sandboxing described in the next subsection alleviates this drawback.
2.2 Security of Platforms

2.2.3 Code Signing and Sandboxing Combined

Java JDK 1.1 combines the advantages of both Code Signing and Sandboxing. If the code consumer trusts the signer of the code, then the code will run as if it were local code, that is, with full privileges being granted to it. On the other hand, if the code consumer does not trust the signer of the code then the code will run inside a Sandbox as in JDK 1.0 [67, 29].

The main advantage of this approach is that it enables the execution of the mobile code produced by untrustworthy entities. However, this method still suffers from the same drawback as Code Signing, that is, malicious code that is deemed trustworthy can cause damage and even change the acceptance policy.

The security policy is the set of rules for granting programs permission to access various platform resources. The “black-and-white” policy only allows the platform to label programs as completely trusted or untrusted, as the case in JDK 1.1. The combination of Code Signing and Sandboxing implemented in JDK 1.2 (Java 2) incorporates fine-grained access control and follows a “shades-of-grey” policy. This policy is more flexible than the “black-and-white” policy, as it allows a user to assign any degree of partial trust to a code, rather than just “trusted” and “untrusted” [70, 29]. There is a whole spectrum of privileges that can be granted to the code. In JDK 1.2 all code is subjected to the same security policy, regardless of being labelled as local or remote. The run-time system partitions code into individual groups called protection domains in such a way that all programs inside the same domain are granted the same set of permissions. The end-user can authorize certain protection domains to access the majority of resources that are available at the executing host while other protection domains may be restricted to the Sandbox environment. In between these two, there are different subsets of privileges that can be granted to different protection domains, based on whether they are local or remote, authorised or not, and even based on the key that is used for the signature [70, 65, 29]. Although this scheme is much more flexible than the one in JDK 1.1, it still suffers from the same problem, that an end user can grant full privileges to malicious mobile code, jeopardising the security of the executing platform.
2.2 Security of Platforms

2.2.4 Proof-Carrying Code

Lee and Necula [75] introduced the Proof-Carrying Code (PCC) technique in which the code producer is required to provide a formal proof that the code complies with the security policy of the code consumer. The code producer sends the code together with the formal safety proof, sometimes called machine-checkable proof, to the code consumer. Upon receipt, the code consumer checks and verifies the safety proof of the incoming code by using a simple and fast proof checker. Depending on the result of the proof validation process, the code is proclaimed safe and consequently executed without any further checking, or it is rejected [52, 64, 67, 75]. PCC guarantees the safety of the incoming code providing that there is no flaw in the verification-condition generator, the logical axioms, the typing rules, and the proof-checker [7].

PCC is considered to be “self-certifying”, because no cryptography or trusted third party is required. It involves low-cost static program checking after which the program can be executed without any expensive run-time checking. In addition, PCC is considered “tamper-proof” as any modification done to the code or the proof will be detected. These advantages make the Proof Carrying Code technique useful not only for mobile agents but also for other applications such as active networks and extensible operating systems [64, 75].

Proof Carrying Code also has some limitations, which need to be dealt with before it can become widely used. The main problem with PCC is the proof generation, and there is a lot of research on how to automate the proof generation process. For example, a certifying compiler can automatically generate the proof through the process of compilation [25, 75]. Unfortunately, at present many proofs still have to be done by hand [67]. Other limitations of the PCC technique include the potential size of the proof and the time consumed in the proof-validation process [75].
2.2.5 State Appraisal

While a mobile agent is roaming among agent platforms, it typically carries the following information: code, static data, collected data, and execution state. The execution state is dynamic data created during the execution of the agent at each platform and used as input to the computations performed on the next platform. The state includes a program counter, registers, local environment, control stack, and store. The state of a mobile agent changes during its execution on a platform. Farmer et al. [39] introduced the “State Appraisal” technique to ensure that an agent has not become malicious or modified as a result of its state alterations at an untrustworthy platform.

In this technique the author, who creates the mobile agent, produces a state appraisal function. This function calculates the maximum set of safe permissions that the agent could request from the host platform, depending on the agent’s current state. In other words, the author needs to anticipate possible harmful modifications to the agent’s state and to counteract them within the appraisal function. Similarly, the sender, who sends the agent to act on his behalf, produces another state appraisal function that determines the set of permissions to be requested by the agent, depending on its current state and on the task to be completed. Subsequently, the sender packages the code with these state appraisal functions. If both the author and the sender sign the agent, their appraisal functions will be protected against malicious modifications. Upon receipt, the target platform checks and verifies the correct state of the incoming agent. Depending on the result of the verification process, the platform can determine what privileges should be granted to this incoming agent given its current state. Clearly, when the author and the sender fail to anticipate certain attacks, they cannot include them in the appraisal functions and provide the necessary protection [39, 52, 97].

In addition to ensuring that an agent has not become malicious during its itinerary, the State Appraisal may also be used to disarm a maliciously altered agent [39]. Another advantage of this technique is that it provides a flexible way for an agent to request permissions depending on its current state and on the task that it needs to do on that particular platform [39, 97]. The main problem with this technique is that it is not easy to formulate
appropriate security properties for the mobile agent and to obtain a state appraisal function that guarantees those properties [97].

### 2.2.6 Path Histories

When an agent travels through a multi-hop itinerary, it visits many platforms that are not all trusted to the same extent. The newly visited platform may benefit from the answers to the following questions: Where has the agent been? How likely is it that the agent has been converted to a malicious one during its trip? To enable the platform to answer these questions, a mobile agent should maintain an authenticable record of the previously visited platforms during its travel life. Using this history, the platform makes the decision whether to run the agent and what level of trust, services, resources and privileges should be granted to the agent [18, 52, 78]. The list of the platforms visited previously by the agent is the basis of trust that the execution platform has in the agent. Typically, it is harder to maintain trust in agents that have previously visited a huge number of platforms. Likewise, it is harder to trust the agent whose travel path is unknown in advance, for example the agent that is searching for new information and creates its travel path dynamically [78].

The “Path History” is constructed in the following way. Each visited platform in the mobile agent’s travel life adds a signed record to the Path History. This record should contain the current platform’s identity together with the identity of the next platform to be visited in the mobile agent’s travel path. Moreover, in order to prevent tampering, each platform should include the previous record in the message digest that it is signing [52]. After executing the agent, the current platform should send the agent together with the complete Path History to the next platform. Depending on the information in the Path History, the new platform can decide whether to run the agent and what privileges should be granted to the agent. The main problem with the Path History technique is that the cost of the path verification process increases with the path history [18, 52, 78]. Constructing algorithms for Path History evaluation is an interesting research area [78].
2.3 Security of Mobile Agents

In the previous section, we presented several techniques for protecting mobile agent platforms against malicious mobile agents. On the other hand, mobile agents themselves are exposed to various threats by the platforms they visit. In order to improve the security of mobile agents against the attacks that are launched by malicious platforms, many security techniques have been suggested. In this section, we explore these techniques.

2.3.1 Co-Operating Agents

The Co-Operating Agent technique [52, 87, 86] distributes critical tasks of a single mobile agent between two co-operating agents. Each of the two co-operating agents executes the tasks in one of two disjoint sets of platforms. The co-operating agents share the same data and exchange information in a secret way. The Co-Operating Agent technique reduces the possibility of the shared data being pilfered by a single host. Each agent records and verifies the route of its co-operating agent [87, 86]. Co-Operating Agents can be used to perform e-commerce tasks or protocols such as the authorization of negotiation, bidding, auction, electronic payment, etc [87, 117].

When the agent travels from one platform to another, it uses an authenticated communication channel to pass information to its co-operating agent. The information includes details about the agent’s itinerary such as the last platform visited by the agent, the current platform, and the next platform to be visited. The peer agent takes a suitable action when anything wrong occurs, e.g., a platform sends the agent to a wrong destination, or claims to have received the agent from an incorrect source. However, this technique has some drawbacks. One of them is the cost of setting up the authenticated communication channel for each migration. Another drawback is that in the case of a co-operating agent being killed, it is difficult for its peer to decide which platform is responsible [52, 87, 86].

It is worth noting that an assumption made in the Co-Operating Agent technique, is that only a small percentage of platforms are in fact malicious.
and that it is not very likely that both agents will encounter such a host. However, care should be taken that the two sets of platforms assigned to the two agents are indeed disjoint, that is, that they never encounter the same host. This method can easily be extended to more than two co-operating agents.

### 2.3.2 Execution Tracing

Execution Tracing enables detection of any possible misbehavior by a platform, that is, improper modification of the mobile agent code, state, and execution flow. This technique is based on cryptographic traces that are collected during an agent’s execution at different platforms. Traces are logs of the actions performed by a mobile agent during its lifetime. Execution Tracing enables an agent’s owner to check the agent’s execution history and see if it contains any unauthorized modifications done by a malicious platform. Each trace contains identifiers of all the statements performed on a particular platform. In the case that some of the statements require information from the external execution environment, the trace must also contain a digital signature of the platform. Such statements are known as “black” statements. On the other hand, the statements that only use the values of the agent’s internal variables are called “white” statements [100, 108].

The Execution Tracing technique assumes that all the involved parties own a public and private key that can be used for digital signatures, in order to identify involved parties. Different parties, such as users and platform owners, communicate by using signed messages. A platform that receives the agent and agrees to execute it produces the associated trace during the agent’s execution. The message that an execution platform attaches to the mobile agent typically contains information such as the unique identifier of the message, the identity of the sender, the timestamp, the fingerprint of the trace, the final state and the trusted third party (which could later be used to resolve disputes). Later, the owner of the agent may suspect that certain platform cheated while executing the agent. If this is the case, the owner will ask the suspicious platform to reproduce the trace. Finally, the agent’s owner validates the execution of the agent by comparing the fingerprint of
2.3 Security of Mobile Agents

the reproduced trace against the fingerprint of the trace that is originally supplied by the suspicious platform [108].

In addition to detection of any modification of the agent performed by a malicious platform, Execution Tracing also provides a means to protect a legitimate platform against a malicious agent by obtaining the related traces from the involved parties. Execution Tracing has some limitations, such as the potential large size and number of logs to be retained. Another limitation of this technique is that the owner platform needs to wait until it obtains suspicious results in order to run the verification process. Also, this technique is considered to be too difficult to use in the case of multi-threaded agents [100, 108].

A new version of the Execution Tracing technique, proposed by Tan and Moreau [99, 100], modifies the original technique by assigning the trace verification process to a trusted third party, the verification server, instead of depending on the agent’s owner.

When a mobile agent travels to a new platform during its itinerary, a copy of the agent is submitted to a corresponding verification server. The visited platform receives the agent and produces the associated execution trace. Before the agent’s migration from the current platform to a new one, the current platform forwards the trace to a corresponding verification server. The verification server simulates the execution of the agent on the platform by using the corresponding execution trace and the agent’s copy. The simulation process is repeated for every platform in the agent’s path by the corresponding verification server, until the agent is sent back to its home platform. Tan and Moreau [100] provided a detailed protocol of message exchanges, as well as the formal modeling and verification of the protocol.

Execution Tracing with a verification server does not wait until a suspicion is raised in order to run the verification process. The verification here is compulsory and this is an advantage over the original Execution Tracing technique where the verification process is triggered only by suspicious results [100]. However, Execution Tracing with a verification server still suffers from the same limitation as the original technique, that is, the need to retain a potentially large size and number of logs. Additionally, each plat-
form chooses a verification server and that might encourage and facilitate a possible malicious collaboration between a platform and the server.

### 2.3.3 Environmental Key Generation

Riordan and Schneier [84] designed the Environmental Key Generation technique to be used when a platform wants to communicate with another platform by sending it a message, yet it only wants the receiving platform to obtain the message if some environmental condition is satisfied. This can be achieved by sending a mobile agent carrying an encrypted message. The encrypted message may include some data and/or executable code. Neither can the mobile agent precisely predict its own execution at the receiver platform, nor can the platform foresee the incoming agent task. The agent will wait at the receiving platform for some environmental condition to occur. The environmental condition could be, for example, matching a certain search string. When the environmental condition is met, an activation key is generated in order to decrypt the enciphered message that the mobile agent is carrying. Without meeting the environmental condition, the agent is unable to decrypt its own message [84].

The activation key, which is used to decrypt the agent’s message, could be hidden inside a fixed data channel. If this data channel is, for example, a file system, then the activation key could be hidden in a file or could be the hash of a certain file name. On the other hand, if the data channel is a mail message, the activation key could be a string inside this message or a hash of the message [84].

Environmental Key Generation may suit some applications other than mobile agents (some of which may even be malicious) including blind search engines, logic bombs, directed viruses, and remote alarms [84]. Tschudin exploited the idea of Environmental Key Generation for the purpose of the programmed death of a mobile service, that is, the self-destruction of a mobile service when it is no longer required [106]. However, this technique has some limitations. The receiving platform could act maliciously against the incoming agent. When the environmental condition is met and the activation key is generated, the platform could modify the agent to perform...
a different function, for example, to print out the executable code instead of running it [52]. Another limitation of the technique is that the platform may consider it unsafe to execute an encrypted code that is attached to a mobile agent, as it could be, for example, a virus.

### 2.3.4 Non-Interactive Computing with Encrypted Functions

This technique represents a software solution for protecting a mobile agent from a malicious executing platform during its itinerary. This is a cryptographic solution to achieve integrity and privacy of the mobile agent. Protecting integrity means that the mobile agent is made safe against tampering by a malicious platform. Achieving privacy means that the mobile agent can conceal its program (code) when it is executed remotely in an untrusted environment. In addition to this, a mobile agent can safely compute cryptographic primitives on a remote platform by using this approach. An example of cryptographic primitives is a digital signature or encryption.

This technique is based on executing a program embodying an encrypted function on a mobile agent platform. It also ensures that the platform does not learn anything substantial about the encrypted function. Abadi and Feigenbaum [3] suggested the initial version of this technique. Their solution was interactive and required several rounds of message exchange with the agent’s home platform. However, the interactive solution does not suit the mobile agent scenario, as agents operate autonomously without much interaction with their home platform.

Sander and Tschudin [91, 92] suggested a non-interactive solution, which is suitable for the mobile agent paradigm. In their solution, the home platform has an algorithm to compute a function \( f \). The target platform has an input \( x \) and can provide a service to the home platform by computing \( f(x) \). However, the home platform doesn’t want the target platform to learn anything about the function \( f \). The home platform launches the operation by encrypting the function \( f \) to get \( E(f) \), and then it implements \( E(f) \) using the program \( P(E(f)) \). The home platform embeds the program \( P(E(f)) \) within the mobile agent and sends it to the target platform for execution. The target platform receives the agent and runs it. This includes executing
2.3 Security of Mobile Agents

\( P(E(f)) \) at \( x \) to produce \( P(E(f))(x) \). Then, the target platform sends the agent back to its home platform. The home platform extracts the result from the agent and then decrypts it to get \( f(x) \).

This solution enables the owner of the agent to execute encrypted programs over untrusted platforms. The executing platforms do not need to decrypt programs before running them. Assume that \( f \) is an encryption algorithm or a signature algorithm that contains an embedded key within it. That means that the agent has the ability to encrypt information or sign it without revealing anything about the value of the key being used.

The main challenge in this technique is to find a way to apply it to an arbitrary function \( f \). At the moment the only classes of functions for which a suitable encryption is known are polynomial and rational functions [3, 91]. Although this technique protects the mobile agent’s integrity and privacy, it is vulnerable to certain attacks such as denial of service and replay attacks [91].

2.3.5 Obfuscated Code

**Obfuscation** is a technique in which the mobile code producer enforces the security policy by applying a behavior-preserving transformation to the code before it sends it to run on different platforms that are trusted to various degrees [30]. Obfuscation aims to protect the code from being analysed and understood by the host. Consequently, the host should not be able to modify the mobile code’s behavior or expose sensitive information that is hidden inside the code such as a secret key, credit card number, or bidding limits [30].

Typically, the transformation procedure that is used to generate the obfuscated code aims to make the obfuscated code very hard to understand or analyse by malicious parties. There are different useful obfuscating transformations [26, 48, 114]. Layout Obfuscation tries to remove or modify some information in the code, such as comments and debugging information, without affecting the executable part of the code. Data Obfuscation concentrates on obfuscating the data and data structures in the code without modifying
the code itself. Control Obfuscation tries to alter the control flow in the code without modifying the computing part of the code. Preventive Obfuscation concentrates on protecting the code from decompilators and debuggers.

Hohl [50] suggested using the Obfuscation technique to obtain a time-limited black box agent that can be executed safely on a malicious platform for a certain period of time but not forever. D’Anna et al. [30] pointed out that Obfuscation could delay, but not prevent the attacks on agent via reverse engineering. They also argue that an attacker with enough computational resources, such as enough time, can always deobfuscate the code. Barak et al. [8] studied the theoretical limits of Obfuscation techniques and showed that in general achieving completely secure Obfuscation is impossible.

In addition to protecting a mobile agent, Obfuscation can also be used for other applications such as protecting digital watermarking, enforcement of software licensing, and protecting protocols from spoofing [30, 114]. As far as the performance is concerned, some Obfuscation techniques reduce the size of the code and thus speed up its execution (Layout and Data Obfuscation), while others achieve the opposite (Control Obfuscation) [48]. Obfuscation is considered resistant to impersonation and denial of service attacks [114]. The main challenge in this technique is to make it easy to apply in practice.

2.3.6 Partial Result Encapsulation

Partial Result Encapsulation (PRE) is a detection technique that aims to discover any possible security breaches on an agent during its execution at different platforms. PRE is used to encapsulate the results of agent execution at each visited platform in its travel path. The encapsulated information is later used to verify that the agent was not attacked by a malicious platform. The verification process can be done when the agent returns to its home platform or at certain intermediate points in its itinerary.

The PRE technique has different implementations. In certain scenarios, the agent itself performs the encapsulation, while in others the platform
2.3 Security of Mobile Agents

does it. To meet certain security requirements such as integrity, accountability, and privacy of the agent, PRE makes use of different cryptographic primitives, such as encryption, digital signatures, authentication codes, and hash functions.

To ensure the confidentiality of its results, the agent encrypts the results by using the public key of its originator to produce small pieces of ciphertext that are decrypted later at the agent’s home platform using the corresponding private key. This is one scenario of PRE where the agent itself does the encapsulation process. The agent uses a special implementation of encryption called “Sliding Encryption” that was suggested by Young and Yung [119]. Sliding Encryption encrypts small amounts of data within a larger block and thus obtains small pieces of ciphertext. Sliding Encryption is particularly suitable for certain application where storage space is valuable such as smartcards [79].

Yee [118] suggested “Partial Result Authentication Code” (PRAC), where again the agent does the encapsulation of the results. However, the agent’s originator also takes part in this scenario by providing the agent with a list of secret keys before launching it. For each visited platform in an agent’s itinerary, there is an associated secret key. When an agent finishes an execution at a certain platform in its itinerary, it summarizes the results of its execution in a message for the home platform, which could be sent either immediately or later. In order to produce the PRAC, the agent uses the associated secret key for the current platform to compute a Message Authentication Code (MAC), which is encapsulated together with the message to produce PRAC. It is important to note that the agent erases the used secret key of the current visited platform before its migration to the next platform. Destroying the secret key ensures the “forward integrity” of the encapsulation results. Forward integrity [118] guarantees that no platform to be visited in the future is able to modify any results from the previously visited platforms, as there is no secret key to compute the PRAC for these results. Only the agent’s originator has a copy of all used secret keys and thus can verify the encapsulated results. The result verification enables the originator to detect any modification (tampering) of the agent’s results. Yee [118] suggested that the results could also be encrypted using
the originator’s public key, in order to guarantee both privacy and integrity.

Karjoth et al. [57] proposed a “strong forward integrity”, which, in addition to forward integrity, also requires that the visited platform can not later modify its own results. Karjoth et al’s approach depends on the visited platform doing the encapsulation process instead of the agent doing it. The visited platform encrypts the agent’s results by using the originator’s public key to ensure the confidentiality of the results. Then the visited platform uses its private key to digitally sign the encrypted results together with a hash chain. The hash chain links the results from the previous platform with the identity of the next platform to be visited. This prevents the platform from changing its results later and thus ensures strong forward integrity [57].

2.3.7 Detection of Denial of Services

Cubaleska and Schneider proposed a posteriori method in order to enable detection of any Denial of Services’ attack (DoS) on the agent [28]. DoS attack includes preventing the agent from accomplishing its task, preventing the agent from migrating to its next destination, and destroying the agent. Their proposed method is based on the usage of undeniable proofs, e.g., digital signature. When agent’s owner suspects that the agent suffered from DoS attack, e.g. the agent did not return back after a certain threshold period of waiting time. The owner then asks all visited platforms in agent’s itinerary to introduce the undeniable proof in order to judge that the visited platform did not launch DoS attack against the agent and correctly dispatch the agent. By doing so, the owner is able to detect the malicious platform that launched a DoS attack on the visiting agent as the malicious platform is unable to show the undeniable proof to the agent’s owner [28]. Note that Cualeska and Schneider approach requires the route of the agent to be fixed and known in advance. The knowledge gained from their DoS detection method enables the agent’s owner to build a trust policy in order to eliminate malicious hosts, create agent’s route, and detect the appropriate sequence in which different trusted platforms should be visited in the future [28].

The mobile agent system is a very promising paradigm that has already established its presence in many applications including e-commerce and dis-
tributed information search and retrieval. At the same time, this technology has introduced some very serious security problems and emphasized some existing security issues. It is more difficult to ensure security in the mobile agent paradigm than in some other technologies where hardware solutions are practical.

In this chapter we surveyed the main issues in the security of mobile agents. We considered both the mobile agent and the agent platform points of view, and reconfirmed that it is much more difficult to ensure the security of mobile agents than the security of agent platforms. We discussed the security threats and requirements that need to be met in order to alleviate those threats.

In the next chapter, we look at the concept of trust within mobile agent context. We first define trust and then explore its properties and classification in the context of mobile agent paradigm. Additionally, we introduce a security solution that are based on the trust as a social control to work together with traditional existing security mechanisms. We call the proposed solution coupling technique. It aims to increase the probability that the agent successfully accomplishes its task by partitioning the itinerary into the pairs, or “couples” of platforms. The coupling radically increases the overall probability that the agent will not be harmed, especially if the itinerary contains servers with a very low trust level.
Chapter 3

Trust and Mobile Agents

Traditional security mechanisms typically rely on authentication, access control, and cryptographic techniques [83, 85]. These traditional security solutions are unable to fulfill various security requirements emerging in open and distributed online environment. Therefore, there is a need for new security solutions that can accommodate different aspects of open system architecture including the social behavior of its entities and can work side by side with traditional security mechanisms [85]. Rasmusson and Jansson [83] defined social control as “a group behaviour that indirectly forces the group members to behave in a particular way”. In this context, it is assumed that malicious entities exist in a community. The goal is to identify these entities and prevent them from causing harm to other entities in the community. Social control is considered a soft security solution [83]. Trust is an example of a social control mechanism and is regarded as a new dimension of security in virtual online communities. A mobile agent system is an example of an online environment.

Trust has been studied in various fields including sociology, philosophy, history, economics, and more recently in computer security. In this chapter we look at the concept of trust within mobile agent context. We first define trust and then explore its properties and classification in the context of mobile agent paradigm. Additionally, we introduce a security solution that is based on the trust dimension as a social control to work together with tradi-
3.1 Trust Definition

It is challenging to provide an accurate generic definition of trust as it is a multi-disciplinary concept. Trust has been studied extensively in various fields and has different definitions and meanings which make it hard to grasp the essence of trust. In order to avoid a confusion coming from this diversity, trust definition should be a domain specific. We mention here some widely used definitions of trust.

Gambetta [41] defines trust as “a particular level of the subjective probability with which an agent assesses that another agent or group of agents will perform a particular action, both before he can monitor such action (or independently of his capacity ever to be able to monitor it) and in a context in which it affects his own action”. Dasgupta [31] defines trust as “a belief an agent has that the other party will do what it says it will (being honest and reliable) or reciprocate (being reciprocative for the common good of both), given an opportunity to defect to get higher payoffs”.

A related concept of distrust has also been defined in literature. For example, Grandison [45] define distrust as “the quantified belief by a trustor that a trustee is incompetent, dishonest, not secure or not dependable within a specified context”.

Reputation is considered an essential part of many trust models and it has many definitions in the research literature as well. Abdul-Rahman and Hailes [4] define reputation as “what is generally said or believed about a person’s or thing’s character or standing”. Sabater and Sierra [89] consider reputation to be “the opinion or view of someone about something”.

tional existing security mechanisms. We call the proposed solution coupling technique. It aims to increase the probability that the agent successfully accomplishes its task by partitioning the itinerary into the pairs, or “couples” of platforms. The coupling radically increases the overall probability that the agent will not be harmed, especially if the itinerary contains servers with a very low trust level.
Blaze et al. [12] define trust management as “a unified approach to specifying and interpreting security policies, credentials, and relationships that allow direct authorization of security-critical actions”. In order to offer different levels of reasoning about interaction partners, many trust models were suggested to assist an entity in taking the right decision about its interaction partner [81].

### 3.2 Trust Properties

Trust relationship can be considered as a binary relation between the trustor and trustee. Stakeholders in a mobile agent community include: mobile agent (MA), home platform (H-PF), execution platform (E-PF), and trusted third party (TTP) such as Petrol station (PS). Trust relationship has many properties as follows:

- **Context-Dependent**: Trust relationship is not absolute [45]. Entity A trusts entity B to perform a certain action in certain context for a certain purpose. For example, in Figure 3.1, a mobile agent may trust an execution environment to send it to its next destination in its itinerary. The same mobile agent may not agree to give credit card details to that execution environment as it does not trust the execution environment within this context.

![Figure 3.1: Trust Properties: Context-Dependent](image)

- **Conditionally Transitive**: Trust relationship can be transitive under certain semantic conditions [55]. Transitivity extends a trust relationship to include more entities than the two who form this relationship.
If entity A trusts entity B, and entity B trusts entity C, then that does not necessarily mean A trusts C unconditionally. A may trust C only if certain conditions are met. For example, in Figure 3.2, a mobile agent trusts execution environment “1” in context “1” and this execution environment “1” trusts execution environment “2” in the same context. This will lead to a transitive trust relationship from the mobile agent to the execution environment “2”. Trust between the mobile agent and the execution environment “3” is non transitive as some conditions are not met. The context in which the mobile agent trusts the first platform is different from the context in which the first platform trusts the third platform. Trust delegation is considered an example of trust transitivity.

![Figure 3.2: Trust Properties: Conditionally Transitive](image)

- **Asymmetry:** If entity A trusts entity B, that does not necessarily mean that entity B trusts entity A in an equivalent or identical way [45]. In Figure 3.3, a mobile agent trusts an execution environment with trust level = 0.60 which is lower than the trust level of 0.95 that the execution environment puts into the mobile agent.
3.2 Trust Properties

- **Dynamic Nature:** Trust relationship changes with time [54]. For example, in Figure 3.4, at certain time slot T1, a mobile agent trusts an execution environment with trust level = 0.40. This trust level increases at time slot T2 to reach 0.90.

- **Subjective:** Trust of entity A in entity B is considered a personal matter of entity A. Trust of entity C in the same entity B could be lower or higher as entity C may have a different opinion about entity B than entity A. For example, in Figure 3.5, mobile agent “1” trusts an execution environment with trust level = 0.98 while mobile agent “2” puts lower trust of 0.55 in the same execution environment.
3.3 Trust Classification

Grandison and Solman [46] present a useful way to classify trust in Internet services. They divide trust into five different classes: Resource Access Trust, Service Provision Trust, Certification or Identity Trust, Delegation Trust, and Infrastructure Trust. In what follows we link Grandison and Solman trust classification scheme with different stakeholders in mobile agent community. We reclassify trust according to these stakeholders as follows:

- **Trust by Execution Platform**: An execution platform owns or controls resources and trusts other stakeholders in the community, such as mobile agent, to use these resources without causing any harm or damage to the execution platform. This corresponds to resource access trust. In Figure 3.6, an execution environment plays the role of trust originator in trusting a mobile agent, trust target, of accessing the execution environment resources. The execution environment trusts the mobile agent not to launch a Denial of Service attack on its resources.
• **Trust by Mobile Agent:** A mobile agent trusts other stakeholders in the community, such as execution platform, to provide it with a service without doing any malicious damage to the trust originator (mobile agent). This corresponds to the service provision trust. In Figure 3.7, a mobile agent trusts an execution environment not to compromise its integrity while the mobile agent is being executed there, not to terminate or damage it, and to send it to its next destination upon completion of its task at that platform.

Figure 3.7: Trust Classification: Trust by Mobile Agent

• **Trust by Home Platform:** A home platform (trust originator) delegates some of its authority to other stakeholders in the community, such as mobile agent (trust target). This corresponds to delegation trust. In Figure 3.8, a home platform grants some privileges to its own mobile agent. The home platform believes that its agent acts on its behalf to legitimately perform certain authorized actions such as negotiations. The trusted agent should not misuse the power that is given to it by the trusting home platform.

Figure 3.8: Trust Classification: Trust by Home Platform
3.3 Trust Classification

- **Trust in Trusted Third Party:** Trust here is based on certification issued by a trusted third party and is presented by a trust target to a trust originator to prove the trustworthiness of the trust target and to authenticate its identity [46]. This corresponds to certification or identity trust. In Figure 3.9, an execution environment trusts a Petrol Station, trusted third party, to honestly certify a mobile agent as harmful/harmless. In Figure 3.10, an execution environment trusts a mobile agent to claim its real identity and not to launch an impersonation attack on the trusting platform. Note that this is different from an execution environment trusting a mobile not to misuse its resources (Figure 3.6).

**Figure 3.8:** Trust Classification: Trust by Home Platform

**Figure 3.9:** Trust Classification: Trust in Trusted Third Party - Certification

**Figure 3.10:** Trust Classification: Trust in Trusted Third Party - Identity

Trust
- **Trust in an entity itself:** Here trust target and trust originator are the same entity (implicit trust). The trust originator trusts its administrative procedures and its security policy. It also believes that its strong infrastructure such as a local network or an operating system will be available at the right time to support its different activities. This corresponds to Infrastructure trust. In Figure 3.11, an execution environment has implicit trust that its infrastructure, such as operating system, local network ...etc, supports its security policy.

![Execution Platform](image)

**Figure 3.11: Trust Classification: Trust in an entity itself**

Additionally, trust management has been classified [96] into three different categories based on how trust relationships are established and evaluated. In the first category, *credential and policy-based trust management systems*, different entities in a system use credential verification in order to establish a trust relationship with other entities. Then, according to application-defined policies, restrict access to resources based on the results from credential verification [46]. Examples of such trust management system are PolicyMaker [11] and REFEREE [23].

The second category, *Reputation-based trust management systems*, provides a mechanism that enables an entity to request a resource and evaluate its trust in the entity that is providing the resource and in the reliability of the resource itself. In such systems, a trust value depends on the global reputation of the entity and on the personal perception of that entity. Examples of such trust management system are SPORAS, HISTOS [120] and Beta [53].
In the third category, social network-based trust management systems, different aspects of the social network contribute in computing trust and reputation values and in forming conclusions about other entities in the network. Examples of such systems include REGRET [89] and NodeRanking [80].

Next, we introduce a security solution that is based on the trust as a social control to work together with traditional existing security mechanisms. Coupling technique aims to increase the probability that the agent successfully accomplishes its task by partitioning the itinerary into the pairs, or “couples” of platforms. Coupling is a powerful technique that requires the mobile agent’s itinerary and the trust values of the platforms in the itinerary to be known in advance. This technique is suitable to be used within the Routed mobile agent that will be presented in the next chapter.

3.4 The Coupling Technique

The main idea behind the coupling technique is to increase the probability that the agent successfully accomplishes its task by partitioning the itinerary into the pairs, or “couples” of platforms. Each platform sends the agent to both members of the next couple, and each platform receives the agent from both members of the previous couple. If the number of platforms in the itinerary is odd, the remaining platform is “coupled” with the home platform, and it sends the agent to the home platform (see Figure 3.12). The agent can only be harmed or purposely destroyed if there is a couple of platforms such that both members of the couple are malicious. Thus the coupling radically increases the overall probability that the agent will not be harmed, especially if the itinerary contains servers with a very low trust level.
Let us denote by $p_i$ the probability that the agent “survives” unharmed after visiting the platform $S_i$ and let $q_i = 1/p_i$. Let $P$ be the overall probability of success, that is, the probability that the agent will “survive” the whole itinerary. Let $n$ be the number of platforms in the itinerary, and let $k = \lfloor n/2 \rfloor$. If the platforms are visited sequentially one after another, we denote the overall probability of success by $P_S$ and we have

$$P_S = \prod_{i=1}^{n} p_i \quad (3.1)$$

If, however, the platforms are coupled and the couples are visited sequentially, we denote the overall probability of success by $P_C$ and we have

$$P_C = \prod_{i=1}^{k} (p_i^1 + p_i^2 - p_i^1p_i^2) \quad (3.2)$$

Furthermore, for $n = 2k$ we have

$$P_C/P_S = \prod_{i=1}^{k} (q_1^i + q_2^i - 1) \quad (3.3)$$

and for $n = 2k + 1$ we have

$$P_C/P_S = q^0 \prod_{i=1}^{k} (q_1^i + q_2^i - 1) \quad (3.4)$$

where $p^0$ is the probability of survival at the platform which is coupled
with the home platform, \( p_1^i \) and \( p_2^i \) are the probability of survivals at the first and second platform of couple \( i \), respectively, and \( q_0^0 = 1/p_0^0, q_1^i = 1/p_1^i, q_2^i = 1/p_2^i \). Our goal is clearly to perform the coupling of the platforms in such a way so as to maximise \( P_C \), and consequently also \( P_C/P_S \). The following Theorem shows how to achieve this goal.

**Theorem 1.** Let the itinerary of a Routed agent contain \( n \) platforms and let \( k = \lfloor n/2 \rfloor \). Furthermore, let the trust levels of the platforms \( S_1, S_2, S_3, \ldots, S_{2k} \) satisfy the inequality: \( p_1 \geq p_2 \geq p_3 \geq \ldots \geq p_{2k} \) or, equivalently, \( q_1 \leq q_2 \leq q_3 \leq \ldots \leq q_{2k} \). In the case of \( n \) odd, that is, \( n = 2k + 1 \), let the remaining platform have a trust level \( p_{2k+1} \leq p_{2k} \). Then \( P_C/P_S \) is maximised when the coupling \( K \) of the platforms is:

\[
K = \{ (S_i, S_{2k-i+1}) \mid 1 \leq i \leq k \} \tag{3.5}
\]

**Proof.** We first prove the Theorem for the case where \( n \) is even, that is, \( n = 2k \), and then we extend our proof to the case of \( n \) odd. We use the mathematical induction.

**Base case:** For \( n = 2 \), there is no choice. For \( n = 4 \) we have:

\[
P_C/P_S = (q_1^1 + q_2^1 - 1)(q_1^2 + q_2^2 - 1) = 1 - \sum_{i=1}^{4} q_i + \sum_{i,j=1, i \neq j}^{4} q_i q_j - (q_1^1 q_2^1 + q_1^2 q_2^2) \tag{3.6}
\]

We thus need to minimise \((q_1^1 q_2^1 + q_1^2 q_2^2)\). It is an easy exercise to show that this expression is minimised when the couples of platforms are \((S_1, S_4)\) and \((S_2, S_3)\), and we leave it to the reader.

**Inductive hypothesis:** Assume that Theorem 1 holds for every \( n = 2k' \), where \( k' \leq k \).

**Inductive step:** We now show that the Theorem 1 holds for \( n = 2k + 2 \). Let the trust levels of the platforms \( S_1, S_2, \ldots, S_{2k+2} \) satisfy the equality \( p_1 \geq p_2 \geq \ldots \geq p_{2k+2} \). Suppose that in an optimal coupling \( C \) the platforms \( S_1 \) and \( S_{2k+2} \) are not in the same couple. Then there exist platforms \( S_i \) and \( S_j \) such that \((S_1, S_i)\) and \((S_j, S_{2k+2})\) are in the optimal coupling \( C \) and we have
3.4 The Coupling Technique

\[ P_C / P_S = (q_1 + q_i - 1)(q_j + q_{2k+2} - 1) \prod_{i=1}^{k-1} (q_i^1 + q_i^2 - 1) \]  

(3.7)

However, from the base case it follows that the optimal coupling for platforms \( S_1, S_i, S_j, S_{2k+2} \) is \( \{(S_1, S_{2k+2}), (S_i, S_j)\} \) and thus there exists an optimal coupling \( C' \) in which \( S_1 \) and \( S_{2k+2} \) are in the same couple:

\[ P_{C'} / P_S = (q_1 + q_{2k+2} - 1)(q_i + q_j - 1) \prod_{i=1}^{k-1} (q_i^1 + q_i^2 - 1) \geq (q_1 + q_i - 1)(q_j + q_{2k+2} - 1) \prod_{i=1}^{k-1} (q_i^1 + q_i^2 - 1) = P_C / P_S \]  

(3.8)

We now remove the \( (S_1, S_{2k+2}) \) couple and consider the remaining \( 2k \) platforms \( S_2, S_3, ..., S_{2k+1} \). From the inductive hypothesis it follows that the optimal coupling for the remaining \( 2k \) platforms is \( \{(S_i, S_{2k-i+3}), 2 \leq i \leq k+1\} \). Then the optimal total coupling is \( \{(S_i, S_{2k-i+3}) | 1 \leq i \leq k+1\} \).

We now consider the case where \( n \) is odd, that is, \( n = 2k+1 \). Let \( j \) be the platform to be coupled with the home platform. Then we need to optimally couple the set \( M \) of platforms,

\[ M = \{S_i | 1 \leq i \leq 2k+1 \text{ and } i \neq j\} \]

As \( |M| \) is even, from the above result it follows that the optimal coupling \( C \) is:

\[ C = \{(S_i, S_{2k-i+2}) | 1 \leq i < j\} \cup \{(S_i, S_{2k-i+3}) | j < i \leq k+1\} \]

for \( j \leq k \), and:

\[ C = \{(S_i, S_{2k-i+2}) | j < 2k - i + 2 \leq 2k + 1\} \cup \{(S_i, S_{2k-i+1}) | k + 1 \leq 2k - i + 1 < j\} \]

for \( j > k \).

While the above notation may look a bit complicated at the first sight, it represents a very simple coupling where in set \( M \) of platforms, the platform
3.4 The Coupling Technique

with the largest trust level is coupled with the platform with the smallest, the platform with the second largest with the one with the second smallest, and so on. It is straightforward to see that $P_C/P_S$ is maximised when $j = 2k + 1$.

We note that the improved security in the coupling technique does not come for free but rather involves a certain cost, mostly due to the growing size of the agent as it progresses through its itinerary. In order to reduce the size overheads, the home platform may choose not to couple all the platforms in the itinerary, but only some of them. In the following Theorem we show how to choose the platforms to be coupled.

**Theorem 2.** Let the itinerary of the Routed agent $S_1, S_2, S_3, ..., S_n$ satisfy the inequality: $p_1 \geq p_2 \geq p_3 \geq ... \geq p_n$. If only $2l \leq n$ platforms are to be coupled, then $P_C/P_S$ is maximised when the coupling $C$ is:

$$C = \{(S_{n-2l+i}, S_{n-i+1}) \mid 1 \leq i \leq l\}$$

**Proof.** Consider an arbitrary collection $M$ of platforms such that $|M| = 2l$. Let us denote the platforms in $M$ as $S_{\pi(1)}, S_{\pi(2)}, S_{\pi(3)}, ..., S_{\pi(2l)}$, such that $p_{\pi(1)} \geq p_{\pi(2)} \geq p_{\pi(3)} \geq ... \geq p_{\pi(2l)}$. Then the optimal coupling $C$ of platforms in $M$ is $C = \{(S_{\pi(i)}, S_{\pi(2l-i+1)}) \mid 1 \leq i \leq l\}$ and the corresponding $P_C/P_S$ is $P_C/P_S = \prod_{i=1}^{l} (q_{\pi(i)} + q_{\pi(2l-i+1)} - 1)$.

Consider now the collection $M'$ of platforms where:

$$M' = \{S_{n-2l+1}, S_{n-2l+2}, ..., S_n\}$$

The optimal coupling $C'$ of platforms in $M'$ is:

$$C' = \{(S_{n-2l+i}, S_{n-i+1}) \mid 1 \leq i \leq l\}$$

and the corresponding $P_{C'}/P_S$ is:

$$P_{C'}/P_S = \prod_{i=1}^{l} (q_{n-2l+i} + q_{n-i+1} - 1)$$

As $q_{n-2l+i} \geq q_{\pi(i)}$ and $q_{n-i+1} \geq q_{\pi(2l-i+1)}$ for all $i, 1 \leq i \leq l$, it follows that
3.4 The Coupling Technique

\( \frac{P_C}{P_S} \leq \frac{P_{C'}}{P_S} \) and thus \( C' \) is an optimal coupling.

Upon arrival of the agent from one of the platforms in the previous couple, the current platform waits only for the prescribed amount of time for another copy of the agent to arrive. If both copies arrive, the platform merges them, executes the code and appends its own results. Otherwise it acts on the only copy that it has received.

If none of the platforms in the previous couple acted maliciously, the two copies of the agent will differ only in the results appended by the platforms and not in the agent’s code or the results of the previous platforms. However, if at least one of the platforms in the previous couple acted maliciously and changed the results submitted by previous platforms, then the two copies of the agent may differ in the whole result section. In that case, the current platform simply keeps both copies of the result section, which doubles the size of the section. If the malicious platforms are frequently encountered, then the size of the agent can grow significantly, which can further jeopardize the performance of the agent. We next derive the expression for the expected value of the agent’s size. For simplicity, we assume that the length of results appended by different platforms are all the same and are equal to one unit. Let \( p_i \) be a probability that the agent will not be harmed by either platform in the couple \( i \), and let \( a_i \) be the expected value of the size of the result section after merging the results of the couple \( i \). Note that \( a_0 = 0 \). We then have:

\[
a_i = 2a_{i-1} + 2 - p_i a_{i-1}
\]

Note that the minimum possible size \( a_i \) is \( 2i \) and that is achieved in the case where no platform is malicious. In Section 4.3.3, we perform some experiments to show that in the realistic network where malicious agents are rare, the size of the agent remains small.

In the next chapter, we propose Scout and Routed Agent approach that balances security and autonomy of a mobile agent. Our approach relies on two copies of a mobile agent to do the task instead of one. We first send the Scout agent (the first copy) to determine the itinerary and return back to its originator. The originator then filters the path, determines the order of the platforms, and incorporates extra security measures in the second copy
of the agent, the so-called Routed agent. Our approach is very suitable for e-commerce applications such as shopping mobile agent that collects prices and offers. The Scout/Routed agent technique combines the advantages of mobile agent autonomy with increased security, at the expense of the one-time involvement of the home platform. Our Routed agent technique is also applicable independently of the Scout agent, whenever the itinerary and the trust values of the platforms in the itinerary are known. Routed agent technique uses our coupling technique, that is introduced in this chapter, in order to construct its itinerary. Moreover, we introduce the idea of Petrol Station, a highly trusted third party that is distributed within the mobile agents’ community, to ensure that mobile agent visits only a network of mutually trusted platforms. To do so, Petrol Station maintains a database of different platforms and their trust scores. It identifies any malicious platform and prevents it from causing damage or harm to mobile agent by eliminating it from the agent’s itinerary. The Petrol Station follows our Routed agent technique to couple the platforms in the agent itinerary.
Chapter 4

Scout and Routed Mobile Agents

Protecting security of mobile agents is a difficult problem and, despite a number of proposed techniques [52], it remains largely unresolved. There are, however, a few restrictions and/or relaxations that could be imposed on the mobile agent paradigm and that would make the security problem much less complicated [78, 57]. For example, if a mobile agent is restricted to a network of mutually trusted platforms, then most security issues are alleviated [92]. However, this idea challenges one of the basic assumptions of the mobile agent paradigms, i.e., an open and dynamic network environment where a new platform can be added at any time [92]. As a second example, consider a mobile agent that interactively communicates with its home platform. This can surely diminish many security threats as the home platform can check the integrity of the agent after visit to each platform, choose the next platform to be visited and even modify the agent so as to provide maximal protection. However, this requires the home platform to be connected to the network and available to the agent and thus diminishes one of the main advantages of the mobile agent paradigm [92].

Finally, if the itinerary of the agent were known in advance, including both the list of platforms and the order in which they are visited, the task of securing the mobile agent suddenly becomes a great deal easier [78, 57].
It is now possible for the home platform to make use of cryptographic primitives, such as encryption and digital signatures, as all platforms involved in the process are known in advance. Importantly, this would reduce the risk and consequently increase the trust in the agent. Such an agent could also incorporate techniques for protection against Denial of Service (DoS) attacks [28]. A serious problem with determining the itinerary in advance is that this jeopardizes the autonomy of the agent, which is another one of the basic assumptions of the mobile agent paradigm.

In this chapter we assume that the mobile agent environment is open, heterogeneous and dynamic. We propose Scout and Routed Agent approach that balances security and autonomy of a mobile agent. Our approach relies on two copies of a mobile agent to do the task instead of one. We first send the Scout agent (the first copy) to determine the itinerary and return back to its originator. The originator then filters the path, determines the order of the platforms, and incorporates extra security measures in the second copy of the agent, the so-called Routed agent. Our approach is very suitable for e-commerce applications such as shopping mobile agent that collects prices and offers. The Scout/Routed agent technique combines the advantages of mobile agent autonomy with increased security, at the expense of the one-time involvement of the home platform. Our Routed agent technique is also applicable independently of the Scout agent, whenever the itinerary and the trust values of the platforms in the itinerary are known.

Additionally, we introduce the idea of Petrol Station, a highly trusted third party that is distributed within the mobile agents’ community, to ensure that mobile agent visits only a network of mutually trusted platforms. To do so, Petrol Station maintains a database of different platforms and their trust scores. It identifies any malicious platform and prevents it from causing damage or harm to mobile agent by eliminating it from the agent’s itinerary. The Petrol Station follows our Routed agent technique to couple the platforms in the agent itinerary.
4.1 Security Requirements and Protection Techniques

In what follows, we shall use a comparison shopping mobile agent [37] to illustrate various existing security techniques that are applicable in our approach. A user creates a shopping agent and endows it with a description of the task to be accomplished together with a set of parameters or preferences to guide the execution of the task. The shopping mobile agent then roams the network in order to accomplish its task, for example, collects prices and offers for certain products, and returns back to its owner, which then compares the offers and makes the best possible agreement [17].

The agent’s code and collected offers need to be protected from malicious platforms. The main security requirements that arise in this context are as follows.

- **Integrity:** The most important security requirement is the integrity of both agent’s code and the collected offers. No visited platform should be able to tamper with the code or other platforms’ offers in order to win the competition or prevent the agent from accomplishing its task.

- **Confidentiality:** The collected offers should be confidential and not accessible by other platforms. Additionally, for some applications it may be important that each platform can only read the portion of the agent’s code that is relevant to the agent’s execution on that particular platform.

- **Non-repudiability:** In a case of dispute, no visited platform should later be able to deny its submitted offers.

In addition to the above requirements, it would be highly desirable to ensure that the agent is not exposed to the so-called Denial of Service (DoS) attack, where an agent is prevented from accomplishing its task. This attack includes destroying the agent or preventing it from migrating to its next intended destination in its itinerary.
4.2 Our Approach: Scout and Routed Mobile Agents

Generally speaking, protection techniques for mobile agents can be divided into two broad categories: prevention and detection [52]. It appears unfeasible to find a technique that would satisfy all the security requirements and be efficient in all applications of mobile agents. The future developments in security for mobile agents are likely to incorporate a combination of techniques that are best suited to the targeted application [101].

Techniques such as Execution Tracing [100, 108] and Partial Result Encapsulation [57] are used to protect mobile agents against malicious platforms. These techniques are suitable to be incorporated within the Scout mobile agent. In Section 2.3.7, we pointed out “Detection of Denial of Services” technique [28] that requires the itinerary to be known in advance. This technique is suitable to be used within the Routed mobile agent.

4.2 Our Approach: Scout and Routed Mobile Agents

Our approach depends on sending two versions of a mobile agent. We refer to the first version as the “Scout agent” and the second version as the “Routed Agent”. Initially, the platform sends the Scout agent. There are three possible scenarios:

1. The Scout agent may successfully accomplish the task and return to the home platform unharmed.

2. The agent may get exposed to a malicious platform and fail to accomplish the task, but still return to the home platform.

3. The agent may get destroyed by a malicious platform and fail to return to the home platform.

In the first scenario, there is no need for the home platform to send another version of the agent, as the mission has been accomplished and the owner has obtained the required results. In the third scenario, the home platform has no other option but to resend the Scout agent. It is the second scenario in which the home platform makes use of the Routed agent. In what follows, we assume that the second scenario has taken place.
In this scenario, the idea behind sending the Scout agent is to explore and determine the travel path, so that the Routed mobile agent can be sent with the itinerary known in advance. The advantage of this approach is that we simultaneously preserve autonomy and mobility, and provide a high level of security. The former characteristic is achieved by the Scout agent, which moves and migrates freely from one server to another and exercises control over its itinerary as it is free to choose its next destination. The latter is achieved by both the Scout and the Routed agents. As we shall see, in the very unlikely case that the Routed agent gets harmed or terminated, it is indeed possible to identify the responsible platform and exclude it from the itinerary of the following copy of the agent.

In order to create a safe itinerary for the Routed agent, the home platform equips the Scout agent with one or more of the existing tampering detection techniques presented in Chapter 2, for example Execution Tracing, Execution Tracing with a Verification Server, or the modified version of Partial Result Encapsulation. When the Scout returns, the home platform examines the agent’s itinerary and execution results and identifies the platforms that acted maliciously.

Then the home platform creates the safe itinerary for the Routed agent in the following “3” steps:

1. Calculating the Trust: Obtaining the reputation of all the platforms visited by the Scout, and calculating the trust level for each one of them.

2. The Filtration Process: Excluding the platforms whose trust level is below the given threshold, or which are not suitable for another reason.

3. Construction of the Itinerary: Coupling the platforms and constructing the itinerary.

The third step was discussed in the previous chapter. We shall next describe the first and second steps.
4.2 Our Approach: Scout and Routed Mobile Agents

4.2.1 Calculating the trust

We assume the existence of a *Reputation Server* that provides a “Reputation Service”, that is, a system for evaluating the platforms in a particular domain. The Server calculates and stores the “reputation” for each platform in the domain, and (optionally) classifies them as trusted or untrusted. Upon the return of the Scout agent, the home platform contributes to the Reputation Service by sending a report describing the Scout agent’s experience, as registered through the tampering detection techniques employed in the Scout agent. The Reputation Server then recalculates the reputation for each visited platform, updates its records, and also sends the updated reputations to the home platform.

Based on the received updated reputations, the home platform calculates the trust level for each visited platform. We define both the reputation and the trust in a platform as a probability that the platform will not harm or terminate the agent. While the reputation is typically tightly connected to a collective belief that some entity would behave as expected, trust is more related to the personal perception of the same. The home platform might calculate the trust level of some platform visited by the Scout agent as a weighted average of the platform’s reputation and the Scout’s personal experience with the platform.

As we pointed out in the previous chapter, trust relationship can be quantified and expressed in different ways such as discrete numeric values, continuous numeric values, qualitative text labels, and binary values. Josang et al. [54] describe various principles for calculating trust and reputation values including: Simple Summation or Average of Ratings, Bayesian Systems, Discrete Trust Models, Belief Models, Fuzzy Models, and Flow Models. We are interested in a computational model of trust and reputation that result in continuous values of trust. It has been pointed out that researchers in the area of trust continue to introduce new models of trust instead of using already existing models, or at least building upon them. Consequently, we studied different computational trust models and found two of them that could be useful within our framework to calculate trust values.

The first model proposed by Mui et al. [72] and is based on sociological
and biological understanding of certain concepts including trust, reputation,
and a related concept: reciprocity. They propose a probabilistic mechanism
for inference among these concepts. However, their model is severely re-
stricted by two assumptions. They assume that social networks in their
model are static which means that no new agents are allowed to enter or
leave the network. The other assumption is that the action space is restricted
to be binary actions.

The second model is proposed by Ramchurn et al. [82]. Their mo-
del is developed based on what they called it confidence (direct or personal
experience) and reputation (indirect). Their measures are modeled using
fuzzy sets that enable agents to assess and choose their interaction partners.
We believe this second model can be plugged into our framework in order
to give the home platform the opportunity to calculate the trust level of
platforms visited by the Scout agent. Our only requirement is that the
value of trust must lie into the range \([0,1]\). The trust home platform has in
other platforms has the value between 0 (no trust) and 1 (absolute trust).

### 4.2.2 The filtration process

The filtration process serves two main purposes:

1. It improves security of the Routed agent by removing the platforms
   that either exhibited malicious behavior towards the Scout agent or
   whose calculated trust levels are below a given threshold.

2. It improves the performance of the second round (Routed) agent. For
   example, it may eliminate a platform whose offer is invalid or whose
   arrangements for payment are insecure [37].

Thus the filtration not only contributes toward making the agent more
secure but also improves the whole process by meeting the user requirements
and preferences better.
In this Section, we introduce a framework for our approach and present it using an activity diagram with role-based annotations. This type of activity diagram is one of the UML 2.0 standards that is particularly suitable for building agent-based systems [9]. It associates a role with each process and represents them inside a round-cornered rectangle. Figure 4.1 explains the behavior of different roles in our framework, including home platform (Home PF), visited platform (Visited PF), Scout mobile agent, Routed mobile agent, and reputation server (Rep. Serv.).
4.3 Framework for Scout Mobile Agent

4.3.1 Framework

The framework consists of three stages as follow.

**Stage I: Scout Mobile Agent**

- The Scout mobile agent is launched by its home platform with a set of preferences; its goal is to accomplish a prescribed task, e.g., collect offers for a certain product. The Scout agent incorporates tamper detection techniques.

- The Scout freely roams an open and dynamic network such as the Internet and autonomously decides which platform to visit next.

- Upon completing its task, the Scout returns back to its home platform with the results, e.g., collected offers.

**Stage II: Filtration**

- The home platform receives the Scout agent and checks for any malicious modification of it; note that, in the case of a malicious modification, applied detection techniques enable identification of the platform responsible for the modification.

- The home platform sends the report to the Reputation Service which maintains a reputation record for each platform in the system. Every time it receives a report from a platform, it recalculates the reputations and updates reputation records accordingly.

- If no tampering with Scout agent has been detected, the task is accomplished. If the tampering has been detected, the Reputation Service provides the platform with the updated reputations of all the platforms in the Scout Agent’s itinerary.

- The home platform now calculates the trust level for each platform in the Scout Agent itinerary. The trust level is a number between 0 and 1 and represents the probability that a particular platform will properly execute the agent and not harm it in any way. The trust
level generally depends on the reputation of the platform, as well as some other considerations (e.g., the context, self-experience etc).

- The home platform proceeds by filtering the first round results: it eliminates any detected malicious platforms, any platforms with very low level of trust, and any invalid offers.

- The home platform couples the platforms to be visited by the Routed Agent. This coupling is made based on the trust levels of the remaining platforms, as explained in the previous chapter.

### Stage III: Routed Mobile Agent

- The home platform sends the Routed mobile agent with the fixed, coupled itinerary (route). If the confidentiality is particularly important, the home platform can also choose to encrypt the Routed agent with the public key of each platform in the itinerary. Thus only these platforms are able to read and execute the Routed agent. To provide integrity and non-repudiability, each platform sings the execution results (e.g., an offer) with its private key. Similarly, to ensure confidentiality, the platforms encrypt the signed offers with the public key of the home platform. The home platform may also choose to employ one of the existing detection techniques in the Routed Agent, for example, detection of DoS [28], for the case of unlikely event that the Routed agent gets harmed or terminated.

- After completing the itinerary, the Routed agent returns back to its home platform.

### 4.3.2 A simple example

In the following simple example, a Scout mobile agent visits 14 different platforms \([S_1, S_2, ..., S_{14}]\) in a serial fashion migration. The results from the first round trip, together with the trust calculated from the reputation obtained from the Reputation Server, are shown in Table 4.1.

The owner of the agent first filters the data from the first round trip. In
Table 4.1: First Round Itinerary Results

<table>
<thead>
<tr>
<th>Server</th>
<th>$P_i$</th>
<th>Malicious Behaviour</th>
<th>Invalid Offer (YES/NO)</th>
<th>Safe Arrangements for Payment</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>0.99</td>
<td>Negative</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>$S_2$</td>
<td>0.85</td>
<td>Negative</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>$S_3$</td>
<td>0.70</td>
<td>Negative</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>$S_4$</td>
<td>0.20</td>
<td>Negative</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>$S_5$</td>
<td>0.75</td>
<td>Negative</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>$S_6$</td>
<td>0.60</td>
<td>Negative</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>$S_7$</td>
<td>0.50</td>
<td>Negative</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>$S_8$</td>
<td>0.95</td>
<td>Negative</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>$S_9$</td>
<td>0.30</td>
<td>Positive</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>$S_{10}$</td>
<td>0.60</td>
<td>Negative</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>$S_{11}$</td>
<td>0.90</td>
<td>Negative</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>$S_{12}$</td>
<td>0.30</td>
<td>Negative</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>$S_{13}$</td>
<td>0.10</td>
<td>Negative</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>$S_{14}$</td>
<td>0.80</td>
<td>Negative</td>
<td>NO</td>
<td>YES</td>
</tr>
</tbody>
</table>

In this case the owner might eliminate $S_3$ (invalid offer), $S_4$ (very low degree of trust 0.20 and insufficient arrangements for payment), $S_6$ (insufficient arrangements for payment), $S_9$ (untrusted, malicious behaviour detected, and insufficient arrangements for payment), $S_{12}$ (untrusted 0.30), and $S_{13}$ (untrusted 0.10). Thus the platforms remaining to be visited by the Routed agent are $S_1$, $S_8$, $S_{11}$, $S_2$, $S_{14}$, $S_5$, $S_{10}$, and $S_7$. The owner then chooses the following coupling as optimal (see Figure 4.2): $(S_1, S_7)$, $(S_8, S_{10})$, $(S_{11}, S_5)$, and $(S_2, S_{14})$.

Note that the order of couples in the Routed agents itinerary does not affect the overall probability $P_C$ of agent’s survival, which is $P_C = 0.922$, see Section 3.4. On the other hand, if the agent would visit the same platforms sequentially, then the overall probability of success would be $P_S = 0.116$. Thus the Routed agent is $P_C/P_S \cong 8$ times more likely to succeed than its sequential counterpart on the same itinerary.

As another example consider an itinerary of 8 platforms, where each has trust level of 0.99. The coupling method gives $P_C = 0.9996$, while sequential method gives $P_S = 0.9227$, which is still significant improvement of agents reliability.
4.3 Framework for Scout Mobile Agent

4.3.3 Experiments

We present some experimental results which show that the Routed agent performs efficiently in real life environment. Our simulation includes sending the Routed agent to visit different 20 couples of platforms. The experiment is repeated 12 times. In each experiment, we study the changes on the Routed agent size. Table 4.2 summaries experimental results. Column 2 shows the identity of malicious couples. For instance, in experiment 3, couple number 16 acted maliciously against the Routed agent. In experiment 11, there were 3 different malicious couples 1, 10, 11. The Routed agent’s real size, at the end of its itinerary, is shown in column 3. Column 4 shows the expected size of the Routed agent, which is calculated according to the following expression given in Section 3.4:

\[ a_i = 2a_{i-1} + 2 - p_i a_{i-1} \]

In our case, the minimum possible size of the agent is 40 and it occurs when no platform in the itinerary acted maliciously. In experiment 2, the Routed agent encountered 5 different malicious couples. This is very rare to happen in real life environment.

The effect of encountering a malicious couple at the beginning of the Routed agent itinerary has a small effect on the real size. For example, in experiment 2 the second couple acted maliciously but the change on agent’s real size was small while in experiment 8 the change was more significant.
### Table 4.2: Experiment Results

<table>
<thead>
<tr>
<th>Experiment Number</th>
<th>Malicious Couple</th>
<th>Real Size</th>
<th>Expected Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>42</td>
<td>93.28</td>
</tr>
<tr>
<td>2</td>
<td>3, 5, 9, 11, 17</td>
<td>304</td>
<td>93.28</td>
</tr>
<tr>
<td>3</td>
<td>16</td>
<td>70</td>
<td>93.28</td>
</tr>
<tr>
<td>4</td>
<td><em>None</em></td>
<td>40</td>
<td>93.28</td>
</tr>
<tr>
<td>5</td>
<td>14</td>
<td>66</td>
<td>93.28</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>40</td>
<td>93.28</td>
</tr>
<tr>
<td>7</td>
<td><em>None</em></td>
<td>40</td>
<td>93.28</td>
</tr>
<tr>
<td>8</td>
<td>17</td>
<td>72</td>
<td>93.28</td>
</tr>
<tr>
<td>9</td>
<td>5, 16</td>
<td>86</td>
<td>93.28</td>
</tr>
<tr>
<td>10</td>
<td>16</td>
<td>70</td>
<td>93.28</td>
</tr>
<tr>
<td>11</td>
<td>1, 10, 11</td>
<td>96</td>
<td>93.28</td>
</tr>
<tr>
<td>12</td>
<td>11</td>
<td>60</td>
<td>93.28</td>
</tr>
</tbody>
</table>

### 4.3.4 Improving the performance

We discuss here some issues that are related to improving a system performance in the case of using Scout and Routed agent approach:

- Since the time is likely to increase, and home platform also needs some involvement when the scout agent returns, the suitability of this systems will depend on the likelihood of an agent being killed - if it is very low then the overhead by Routed agent is not justified.

- The Routed agent may encounter many malicious couples. This is very rare to happen in real life environment. Normally, the filtration process in our technique would exclude any low trusted platforms and consequently improve the performance of the Routed agent.

- The effect of encountering a malicious couple at the beginning of the Routed agent itinerary has a small effect on the real size. On the other hand, the effect will be larger when a malicious couple is encountered at late stages. In order to lessen this effect, we may rearrange the couples in such a way that low trusted ones are visited first.

- The total time taken by the Routed agent is expected to be shorter
than the time taken by an agent going sequentially through the same itinerary, as the platforms in a couple are processing the agent in parallel. Moreover, a limit can be imposed on the time each platform waits to receive a second copy of the agent, which limits the total time taken by a Routed agent.

- Due to the coupling technique, it is crucial that each platform only appends its own (encrypted) results, and does not encrypt its own results together with all the previous results. Otherwise, when a platform merges the two copies of the agent received from the members of the previous couple, the size of the agent would double, and thus it would grow exponentially through the itinerary. As routed agent only appends new result, the doubling in size will only occur after malicious behavior, that is, when the agent would have otherwise been terminated or harmed. We may assume that these occasions are very rare and thus that the agent’s size will not grow significantly. A possible way to overcome the last drawback of the Routed agent is to couple only some of the platforms, in particular those with lower trust level, as shown in Theorem “2” in Section 3.4. In this way we can balance security and performance of the Routed agent.

- In Scout and Routed Agent approach only platforms are permitted to invoke mobile agents to act on behalf of them. Our approach does not deal with the idea of giving permission to mobile agents to invoke other agents to act on behalf of them. This actually opens the door for further research in the future.

### 4.4 Petrol Stations

#### 4.4.1 Assumptions

Before we present the concept of Petrol Station, we first introduce the assumptions under which Petrol Station operates. At first, we consider a restricted scenario where a mobile agent is restricted to a network of mutually trusted platforms [92]. Indeed, in a real-life situation the majority of
platforms are honest. At second, in Petrol Station approach, mobile agent’s itinerary and the trust values of the platforms in the itinerary are known in advance to the Petrol Station.

### 4.4.2 What is a Petrol Station?

A Petrol Station is a highly trusted third party that is distributed within the mobile agents’ community. On one hand, Petrol Station manages trust in this community and on the other hand it offers various services and resources to different entities, including agents and platforms in the community. Figure 4.3 shows how the proposed Petrol Station fits into the Routed agent scenario. A home platform, H-PF, sends a mobile agent initially to a Petrol Station, PS. The Petrol Station receives the agent and equips it for its trip with, for example, an itinerary to follow. The platforms to be visited by mobile agents, E-PF, are selected by Petrol Station. The selection is based on E-PF trust score and E-PF’s compliance with the nature of agent’s task. As a result of the itinerary being known in advance, the Routed agent technique is applicable independently of the Scout agent. The Petrol Station decides on coupling the trusted platforms to be visited by the Routed Agent. The coupling radically increases the overall probability that the agent will not be harmed. The Petrol Station sends the Routed mobile agent with the fixed, coupled itinerary (route). At the end, the agent accomplishes its task and returns back to the Petrol Station. The Petrol Station then updates its trust records and sends the agent together with its final results to the home platform. The mobile agent here acts as a car that stops at a Petrol Station in order to satisfy its needs for petrol and other services before and after its journey.

![Figure 4.3: Fitting Petrol Station, PS, into the Routed agent scenario.](image-url)
4.4 Petrol Stations

4.4.3 Suggested responsibilities for Petrol Station

Petrol Station plays an essential role in ensuring that mobile agent visits only a network of mutually trusted platforms. To do so, Petrol Station collects ratings about the trustworthiness of different platforms in mobile agent community. These ratings come from two sources: direct ratings and indirect ratings. Direct ratings come from Petrol Station’s direct experience with platforms through, for example, previously sent mobile agents. Indirect ratings come from other Petrol Stations that are spread within the community. More weight should be given to direct ratings. Based on the collected ratings, Petrol Station calculates trust score for different platforms. Computational trust models such as the one proposed by Ramchurn et al. [82] can be used by a Petrol Station to calculate the trust score of the platforms. Recall from Section 4.2.1 that our only requirement is that trust score must be mapped into the range [0,1]. Trust with score “0” refers to no trust, while the score “1” indicates absolute trust.

Petrol Station manages a database of different platforms in mobile agent community including their trust level. As trust is dynamic in nature, Petrol Stations need to continuously update their records by collecting and re-evaluating trust values for different platforms in the community. This also allows Petrol Stations to operate efficiently in a dynamic environment where new platforms may be added and existing platforms may be vanished.

In a more general web services environment, there are four primary roles as follows [98]: service provider, service consumer, service directory, and service broker. In our scenario, a Petrol Station acts not only as service provider by supplying a safe itinerary, but also as a service directory. It guides a mobile agent to get services from the right platform that is trustworthy and at the same time suitable for the nature of mobile agent’s task. Additionally, Petrol Station could provide a safe place for mobile agents from different sources to meet, negotiate, and accomplish different required tasks.

There is a need for insurance service within a mobile agent community [61, 62]. Petrol Stations may offer an insurance service for any hesitating mobile agent in the community in order to reduce the risk that is
associated with a task. A Petrol Station may also play a role of a court that resolves any conflict between entities in the community. To do so, a Petrol Station should have a mechanism to penalize malicious entities, and reward entities with good behavior. One option would be to increase or decrease an entity trust value. Additionally, Petrol Station should provide a good notification service to advise entities about the consequences of their actions.

The idea of Petrol Station contributes not only towards safer mobile agents but also towards the safety of platforms. Upon receiving a mobile agent from its home platform, Petrol Station checks mobile agent’s code in order to assure that this agent is harmless and does not jeopardizes the security of visited platforms. Consequently, Petrol Station certifies the mobile agent as harmless. This will help the involved platforms in mobile agent’s itinerary to receive and execute this agent without concern for the safety of its resources. Petrol Stations could also issue references to other entities in the community about the trustworthiness of a mobile agent. A trust level of a mobile agent will be inherited from its home platform trust level. This will lessen the confusion about the trustworthiness of new mobile agents that join the community.

4.4.4 A framework for Petrol Station

In what follows we introduce a framework for Petrol Station. Similarly to Scout agent framework, we use an activity diagram with role-based annotations [9]. Figure 4.4 explains the behavior of different roles in Petrol Station’s framework including home platform (H-PF), Petrol Station (PS), Routed mobile agent (MA), and visited execution environment or platform (visited E-PF). We next explain each step in Petrol Station’s framework as shown in Figure 4.4:
A home platform (H-PF) prepares its mobile agent (MA) to accomplish a prescribed task, e.g., collect offers for a certain product. For example, the mobile agent may carry a set of preferences provided by its home platform.

The home platform sends its mobile agent to a Petrol Station. The home platform handles a list of preferred Petrol Stations. The home platform trusts the selected Petrol Station and delegates all trust decisions that concern its MA’s itinerary to that PS.

The chosen Petrol Station (PS) receives the mobile agent. The Petrol Station initially studies the nature of the received agent’s task.

The Petrol Station examines the mobile agent code to make sure that it is harmless against any execution environment to be visited by it in the future. If this is the case, the Petrol Station certifies this agent as harmless. This certificate contributes toward having more secure platforms in mobile agent community.

The Petrol Station selects a set of platforms to be visited by the mobile agent. It selects a set of platforms that best suit the mobile agent’s task. Among the selected platforms, the Petrol Station further selects those with the highest trust score.
• The Petrol Station follows the Routed agent technique to couple the platforms in the agent itinerary. Coupling decision is made based on the trust scores of the selected platforms.

• The Petrol Station dispatches the Routed mobile agent in the fixed, coupled itinerary (route).

• The Petrol Station may also decide to encrypt the Routed agent with the public key of each platform in the itinerary. Thus only these platforms are able to read and execute the Routed agent.

• To provide integrity and non-repudiability, each visited platform is required to sign the mobile agent’s execution results (e.g., an offer) with its private key. Similarly, to ensure confidentiality, the platform encrypts the signed offer by the public key of the Petrol Station.

• Additionally, the Petrol Station may choose to employ one of the existing detection techniques in the Routed Agent, for example, detection of DoS [28], for the case of unlikely event that the Routed agent gets harmed or terminated.

• The Routed agent visits all platforms in its itinerary. Then it returns back to the Petrol Station where it was deployed from.

• The Petrol Station retrieves the Routed agent results and study if there has been any malicious action detected during the execution of the agent at different platforms in its itinerary.

• Based on the outcomes from studying the Routed agent results, the Petrol Station re-evaluates the trust level for the platforms in the itinerary. The Petrol Station penalizes malicious platforms by decreasing their trust score. The Petrol Station also rewards platforms with good behavior by increasing their trust level.

• The Petrol Station updates its records according to the re-evaluation results of the trust values. The Petrol Station might notify the involved parties of their action and might notify other Petrol Stations in the community about its changes in trust scores.

• The Petrol Station sends the mobile agent with its final results to its home platform.
The home platform receives its mobile agent from the Petrol Station and extracts the results from its agent. The mission is complete.

The previous chapters of this thesis focuses on the security of the mobile agent. In Chapter 5 and 6, we turn our attention to the security of the platform. In particular, we consider a scenario where a platform hosts a database containing confidential individual information and allows mobile agents to query the database. In Statistical Disclosure Control (SDC), measuring disclosure risk is still considered as a difficult and only partly solved problem [111]. We present related work on discloser risk measures and we introduce a scenario that is not adequately covered by any of the previous discloser risk measures. A database is said to be compromised if a confidential value is disclosed [71]. The compromise can be exact or approximate. In the exact compromise, an intruder learns the exact confidential value. Shannon’s entropy can be considered a satisfactory measure for the disclosure risk that is related to the exact compromise. However, in the approximate compromise, we argue that Shannon’s entropy does not express precisely the intruder’s knowledge about a particular confidential value. We introduce a novel disclosure risk measure that is based on entropy. We believe that this novel measure is satisfactory for data confidentiality. The main advantage of our measure over previously proposed measures that it gives careful consideration to the attribute values in addition to the probabilities with which the values occur. Additionally, our proposed measure is independent of the applied SDC technique. Moreover, we introduce a dynamic programming algorithm for calculating the disclosure risk.
Chapter 5

Security of the Platform: A Novel Entropy-Based Measure of Disclosure Risk

The previous chapter of this thesis focuses on the security of the mobile agent. In this chapter we turn our attention to the security of the platform. Chapter 2 of this thesis discusses the security threats on mobile agent platforms in addition to a discussion about the security requirements that need to be met in order to alleviate these threats. It is important to ensure that the information stored on a platform is accessible only by authorized parties. Moreover, it is crucial to ensure the privacy of such information due to the growing attention and the increasing importance of privacy in general. This part of the thesis addresses how to evaluate the security mechanisms for protecting the privacy of the information, which is stored in a mobile agent platform, from being disclosed by inferences. This problem is studied separately from the context of mobile agent technologies and that is within Privacy Preserving Data Mining.

We consider a scenario where a platform hosts a database containing confidential individual information and allows mobile agents to query the database. For the benefit of the reader we repeat the example we presented in the Introduction in which a Real Estate Agency (REA) maintains
a database containing information about recently sold houses that is available to public for a fee. Suppose that Alice wants to invest in real estate and is interested in recent house sale prices in a particular region. Alice then sends her mobile agent to roam the Internet and collect the information from various real estate agencies. Suppose, however, that Alice’s neighbor Bob has just sold a property in that region. Alice is curious and interested to learn what price Bob achieved for his house but she knows that Bob is a private person and would prefer not to share such information with his neighbours. To use a stronger example, suppose that Alice is not interested in real estate investment at all but is rather using it as a way to infer Bob’s personal information. In other words, a malicious mobile agent might purposely extract a confidential individual information and thus compromise privacy of individuals whose data is stored in various databases. In the light of this example, REA may not wish to disclose individual house prices but only their aggregate values. Consequently, REA must ensure that no sequence of queries asked from the database by a same party or a group of collaborating parties is sufficient to infer an individual house price and to link it to a particular address.

This problem is known as a Statistical Disclosure Control (SDC) [111]. There are various techniques that can be used to alleviate this problem [5]. Unfortunately, none of the available techniques is able to solve the problem completely, due to its intrinsic contradictory nature. On one hand, one must keep the risk of individual value disclosure as low as possible. On the other hand, the utility (usefulness) of the database must remain high. However, low risk implies low utility and high utility implies high risk. A good SDC measure aims at finding a right balance between the two. In order to achieve this balance, it is crucial to adequately measure both utility and disclosure risk. While measuring data utility has been well studied in the literature [111], measuring disclosure risk is still considered as a difficult problem and has been only partly solved. In this chapter we propose a novel entropy based measure of disclosure risk whose main advantage over previously proposed measures is its independence of the applied SDC technique. In the next chapter we apply our measure to a few common SDC techniques.

This chapter is organised as follows. In section 5.1 we present in detail
the SDC problem as well as the abstract model of a statistical database. We give an example of statistical databases that we shall use throughout this chapter. In section 5.2 we present related work on discloser risk measures and in section 5.3 we introduce a scenario that is not adequately covered by any of the previous measures. We further illustrate the issue on an example of a Roulette game. In section 5.4 we introduce a novel entropy-based measure and in section 5.5 we present a dynamic programming algorithm for calculating the disclosure risk. We give concluding remarks in section 5.6.

5.1 Introduction to Statistical Disclosure Control

Recent years have seen a huge increase in collection and storage of personal data and consequently privacy has become a focus of public attention. In this section we explain a Statistical Disclosure Control and discuss various protection techniques proposed in the literature. We start by introducing an abstract model of a statistical database (Table 5.1).

<table>
<thead>
<tr>
<th>Name</th>
<th>Sex</th>
<th>Age</th>
<th>Occupation</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qays</td>
<td>M</td>
<td>37</td>
<td>Lecturer</td>
<td>77,000</td>
</tr>
<tr>
<td>Layla</td>
<td>F</td>
<td>45</td>
<td>Associate Professor</td>
<td>107,000</td>
</tr>
<tr>
<td>Tom</td>
<td>M</td>
<td>47</td>
<td>Associate Lecturer</td>
<td>49,000</td>
</tr>
<tr>
<td>Tony</td>
<td>M</td>
<td>37</td>
<td>Senior Lecturer</td>
<td>80,000</td>
</tr>
<tr>
<td>Robert</td>
<td>M</td>
<td>62</td>
<td>Professor</td>
<td>128,000</td>
</tr>
<tr>
<td>Sam</td>
<td>M</td>
<td>42</td>
<td>Lecturer</td>
<td>73,550</td>
</tr>
<tr>
<td>John</td>
<td>M</td>
<td>57</td>
<td>Professor</td>
<td>132,000</td>
</tr>
<tr>
<td>Angela</td>
<td>F</td>
<td>32</td>
<td>Associate Lecturer</td>
<td>50,000</td>
</tr>
<tr>
<td>Mark</td>
<td>M</td>
<td>33</td>
<td>Lecturer</td>
<td>76,100</td>
</tr>
<tr>
<td>George</td>
<td>M</td>
<td>54</td>
<td>Associate Professor</td>
<td>101,500</td>
</tr>
</tbody>
</table>

In the “AcademicStaff” database, each row (record) corresponds to an academic staff member at a University and each column corresponds to a property (attribute) of staff. In our example, the attributes include “Name”, “Sex”, “Age”, “Occupation” and “Salary”. Among all these attributes, some may be deemed confidential, while others may be considered public knowledge and used to identify records. The later are also refereed to
as “identifying” attributes. In our example, we consider “Salary” as confidential attribute, and the remaining attributes as identifying. We note that some attributes act as a key, that is, have a property of uniquely identifying a record. Such attributes would normally be removed from a statistical database, or at least made inaccessible. In the Academic Staff Database, we consider “Name” as a key attribute.

Some of the attributes in a statistical database assume values that have natural ordering and we refer to such attributes as “numerical”. Other attributes do not exhibit such an ordering and we refer to them as “categorical”. In Academic Staff Database attributes “Name”, “Sex” and “Occupation” are categorical, while “Age” and “Salary” are numerical.

In a statistical database users are not allowed to query individual records but rather to submit only statistical types of queries, such as MIN, MAX, SUM, etc. However, this restriction is not sufficient to guarantee the prevention of database compromise. Indeed, in the absence of SDC methods, a skilled intruder can extract some or all values of confidential attributes as illustrated in the following example: In the above example, Mark is the only 33 years old academic and thus “Q1” reveals his exact salary, although the query appears to be statistical. In general, whenever an individual can be uniquely identified by some combination of identifying attributes, it is always possible to extract their confidential information.

The above example indicates the need for an appropriate SDC technique. These techniques can be classified as modification techniques and query restriction techniques [16, 13, 111, 34]. Modification techniques involve some kind of alternation of the original data set before it is released to statistical users. This includes noise addition, data swapping, aggregation, suppression and sampling [16].

The common denominator of all modification techniques is that the modified database is released to users who are free to perform any query on

<table>
<thead>
<tr>
<th>Q1:</th>
<th>SELECT AVG(Salary) FROM AcademicStaff WHERE Age = 33</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1:</td>
<td>Average salary = 76, 100</td>
</tr>
</tbody>
</table>
5.2 Related Work on Disclosure Risk Measures

In the literature, disclosure risk measures may be classified as measures for record re-identification or confidential value disclosure [38, 63, 13]. The latter focuses on measuring the risk of compromising a confidential value of a particular individual, while the former focuses on measuring the risk of inferring an individual’s identity. On the other hand, the disclosure risk measures may applied to the database as a whole, or can be applied to individual records.
Several methods have been proposed to estimate the disclosure risk in sampling and they fall under the category of record identification. Winkler [112] refers to these methods as Sample-Unique-Population-Unique (SUPU) methods as disclosure risk estimation requires assessing the uniqueness of records in the released sample and in the population. Skinner and Elliot [93] introduce a new disclosure risk measure for microdata which falls under SUPU methods [112]. Their measure is based on the probability that a microdata record and a population unit are correctly matched:

$$\theta = Pr(\text{correctmatch}|\text{uniquematch})$$

(\(\theta\)) is interpreted as the conditional probability that a unique match will be correct. Additionally, they introduce a simple variance estimator and they claim that their measure is able to evaluate the different ways of releasing microdata from a sample survey. Truta et al. [105] introduce other SUPU measures and they call them minimal, maximal, and weighted disclosure risk measures. The minimal disclosure risk measure is the percentage of records in a population that can be correctly re-identified by an intruder. All these records must be population unique. The maximum disclosure risk measure takes into account records that are not population unique while the weighted disclosure risk measure assigns more weights to unique records over other records. Their measures are not linked to a certain individual but compute the overall disclosure risk for the database. These measures can only be applied to limited SDC methods such as sampling and microaggregation and it is considered hard to choose the disclosure risk weight matrix [105]. However, assigning weights enables a data owner to setup different levels of confidentiality. These measures are useful in deciding the order of applying more than one SDC method on the initial data.

Trottini and Fienberg [104] propose a simple Bayesian model for capturing user uncertainty after releasing the data by an agency. They distinguish between the legitimate user (researcher) uncertainty and the malicious user (intruder) uncertainty. This distinction is used as the basis of defining appropriate disclosure risk measure. The proposed measure is an arbitrary decreasing function of the user’s uncertainty about a confidential attribute value.
Spruill [94] proposed an early measure that evaluates the confidentiality of releasing a microdata given that a true dataset, where a released data is derived from, is made publicly available. The proposed measure is considered as an evaluation tool for disclosure risk for the whole database and not for an individual record. It is a percentage of records in the released data where a link with the true data can not be made. In order to decide if there is such a link, for each released record, we add up the difference between the released value and the true value for all common numerical attributes. The difference is either absolute or squared. A link is said to be made if a released record was derived from the true record that has the minimum sum of differences. Spruill evaluated five different releasing SDC methods for microdata. The examined methods were: adding random error, multiplying by random error, grouping, random rounding, and data swapping. Not surprisingly, it was shown that adding more noise or error to the true data produces higher level of confidentiality. Confidentiality decreases when the number of common attributes increases. The measure shows that using data swapping as a releasing strategy achieved very low level of confidentiality. A good side of this measure is that the level of confidentiality could be adjusted by not only selecting true records with minimum sum of differences but also selecting the second and third smallest sum. Another advantage is that it is not designed for a specific SDC method. However, this measure is only applicable to numerical attributes. Additionally, this measure evaluates confidentiality based on identifying records and not confidential attribute values. It only counts number of records which are identifiable by some criteria.

Onganian and Domingo-Ferrer [76] propose a disclosure risk measure to evaluate the security of releasing tabular data. The measure is equal to the reciprocal of conditional entropy given the knowledge of an intruder:

\[ DR(X) = 1/H(X|Y = y) = 1/(\sum_x p(x|y) \log_2 p(x|y)) \]  

(5.2)

where \( X \) represents a confidential attribute for a given record and \( Y \) represents intruder’s knowledge. The disclosure risk is inversely proportional to the uncertainty about the confidential attribute given intruder’s knowledge. The measure performs \textit{a posteriori}, that is, after applying one of
5.2 Related Work on Disclosure Risk Measures

SDC methods to the tables. It is a complement to a priori measures such as some currently used sensitivity rules including \((n,k)\)-dominance and \(pq\)-rule, which help a data owner in deciding whether to release the data or not. The above a posteriori measure is general and applicable to various SDC methods such as Cell Suppression, Rounding, and Table Redesign. In order to evaluate this disclosure risk, one has to find a set of the possible confidential attribute values and their probabilities given the condition \(Y = y\). A downside of this measure is that it does not capture accurately the knowledge that an intruder has about a confidential attribute, as it does not give careful consideration to the attribute values but only the probabilities with which the values occur. Our proposed method considers the attribute values in addition to their probabilities. Let us explain this point by introducing a working example.

Assume, in one scenario, that an intruder has a knowledge about an individual whose corresponding record is stored in Table 5.1. They know that the individual in question is an Associate Lecturer and they try to infer their salary. According to Table 5.1, the Associate Lecturer of interest has a salary of either 49,000 or 50,000. In a different scenario, the intruder has different knowledge that the individual in question is female and therefore their salary is either 107,000 or 50,000. According to the above Onganian and Domingo-Ferrer’s measure, the amount of disclosure risk that is involved in both scenarios are the same. In both scenarios there are 2 values and they have the same probabilities (0.5). As we will show later, our proposed measure does consider the attribute values. It accurately captures the intruder’s knowledge about the salary and yields different amount of risk that is involved in both scenarios. In the first scenario, the intruder knows the salary is in the range \((49,000 - 50,000)\) which constitutes approximate compromise (see Section 5.1). In the second scenario the salary range is much wider \((107,000 - 50,000)\) so we cannot talk about approximate compromise. Recall from Section 5.1 that some attributes in a table may be considered public knowledge and used to identify records. They are referred to as “identifying” attributes or “quasi-identifiers” (QI). Samarati and Sweeney [90] propose an approach to ensure the QIs do not uniquely identify records and they call it generalization. It replaces QI attribute’s values with others that are less specific and divides records into classes where each class
5.2 Related Work on Disclosure Risk Measures

Table 5.2: Example of $k$-anonymity, $l$-diversity: Initial Microdata

<table>
<thead>
<tr>
<th>Age</th>
<th>Postcode</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>22</td>
<td>2299</td>
</tr>
<tr>
<td>2</td>
<td>23</td>
<td>2293</td>
</tr>
<tr>
<td>3</td>
<td>22</td>
<td>2293</td>
</tr>
<tr>
<td>4</td>
<td>22</td>
<td>2294</td>
</tr>
<tr>
<td>5</td>
<td>34</td>
<td>3211</td>
</tr>
<tr>
<td>6</td>
<td>35</td>
<td>3218</td>
</tr>
<tr>
<td>7</td>
<td>35</td>
<td>3217</td>
</tr>
<tr>
<td>8</td>
<td>39</td>
<td>3210</td>
</tr>
<tr>
<td>9</td>
<td>43</td>
<td>4131</td>
</tr>
<tr>
<td>10</td>
<td>49</td>
<td>4130</td>
</tr>
<tr>
<td>11</td>
<td>54</td>
<td>4137</td>
</tr>
<tr>
<td>12</td>
<td>59</td>
<td>4135</td>
</tr>
</tbody>
</table>

or set shares the same QI attributes’ values. A class of records where values of QI attributes are the same is called equivalence class. To limit disclosure, Samarati and Sweeney [90] propose the so-called $k$-anonymity that requires each equivalence class to have no less than $k$ records. Table 5.3 represents an example of a released dataset that is a generalization of Table 5.2. The released microdata in Table 5.3 is anonymous with “$k = 4$”, which implies that each equivalence class has at least 4 records. For example, the third equivalence class consists of the last four records. These records share same values of QI attributes. Unfortunately, all of them also share the same value of the confidential attribute. If an intruder knows that an individual who lives in a region with postcode starts with “41” then an exact compromise will occur, as the intruder can infer that the salary of individual in question is exactly “101.5 K”. Thus $k$-anonymity as a notion of privacy does not impose any restriction on confidential attribute values in an equivalence class, as while it successfully prevents record re-identification, it fails to prevent confidential value disclosure.

To overcome this, Machanavajjhala et al propose the so-called $l$-diversity [69]. In addition to the $k$-anonymity requirement that each equivalence class should have no less than $k$ records, $l$-diversity requires that the distribution of confidential attribute values in each equivalence class has at least $l$ distinct values. Going back to the example in Table 5.3, we see that the sec-
Table 5.3: Example of $k$-anonymity, $l$-diversity: 4-anonymous Released Microdata

<table>
<thead>
<tr>
<th></th>
<th>Age</th>
<th>Postcode</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2*</td>
<td>22*</td>
<td>77K</td>
</tr>
<tr>
<td>2</td>
<td>2*</td>
<td>22*</td>
<td>107K</td>
</tr>
<tr>
<td>3</td>
<td>2*</td>
<td>22*</td>
<td>49K</td>
</tr>
<tr>
<td>4</td>
<td>2*</td>
<td>22*</td>
<td>80K</td>
</tr>
<tr>
<td>5</td>
<td>3*</td>
<td>32*</td>
<td>124K</td>
</tr>
<tr>
<td>6</td>
<td>3*</td>
<td>32*</td>
<td>125K</td>
</tr>
<tr>
<td>7</td>
<td>3*</td>
<td>32*</td>
<td>126K</td>
</tr>
<tr>
<td>8</td>
<td>3*</td>
<td>32*</td>
<td>127K</td>
</tr>
<tr>
<td>9</td>
<td>≥40</td>
<td>41*</td>
<td>101.5K</td>
</tr>
<tr>
<td>10</td>
<td>≥40</td>
<td>41*</td>
<td>101.5K</td>
</tr>
<tr>
<td>11</td>
<td>≥40</td>
<td>41*</td>
<td>101.5K</td>
</tr>
<tr>
<td>12</td>
<td>≥40</td>
<td>41*</td>
<td>101.5K</td>
</tr>
</tbody>
</table>

The second equivalence class has distinct values (124K-127K). Although $l$-diversity does consider the attribute values, it does not consider how close these values are from each other in the case of numerical attributes. Our proposed measure takes this into account.

Li et al. [66] introduce a new privacy property that is considered as an improvement over $k$-anonymity and $l$-diversity and they call it $t$-closeness. It requires that the distribution of confidential attribute values in each equivalence class should be close to the distribution of the confidential attribute values in the whole dataset. The distance between the two distributions should not exceed a given threshold $t$. We argue that $t$-Closeness as a notion of privacy is too coarse and often overly restrictive. Our proposed measure is more accurate and is able to represent precisely how much an intruder can learn about confidential values.

5.3 Introducing The Problem

This section introduces our main contribution. Our motivation for this work comes from the need for a confidentiality measure that captures the different
quantitative and qualitative characteristics in different elementary events’ values. Shannon’s entropy measures the uncertainty and the information supplied by a probabilistic experiment [47]. It is worth to notice that this is not enough to express the different characteristics. In addition to the probabilities for different events in an experiment, it is necessary to associate with these probabilities a reasonable weight for the variance within these events in the experiment. This weight considers a reasonable qualitative characteristic that successfully captures the meaning of different events in an experiment while their probabilities capture the quantitative characteristics of these events. We introduce a formal measure for confidentiality breach based on entropy. Our measure takes into account both the number of different values and their probabilities from one side and their variance in distribution from the other side. Thus in a way our measure represents a hybrid between entropy and variance.

<table>
<thead>
<tr>
<th>1st Half</th>
<th>2nd Half</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 4 9 10 15 16</td>
<td>21 22 27 28 33 34</td>
</tr>
<tr>
<td>2 5 8 11 14 17</td>
<td>20 23 26 29 32 35</td>
</tr>
<tr>
<td>1 6 7 12 13 18</td>
<td>19 24 25 30 31 36</td>
</tr>
</tbody>
</table>

First 1/3  Second 1/3  Third 1/3

Figure 5.1: A Roulette Table Layout

We explain the need for such a measure on the example of the Roulette game. Roulette is one of the most popular European gambling games. The Roulette table layout is marked with numbers “1” through “36” as shown in Figure 5.1. We will exclude zero here for simplicity. Roulette players place their bets on individual numbers or groups of numbers inside and/or outside the Roulette table layout. A dealer spins a Roulette wheel. Eventually, a Roulette ball drops on a slot in the Roulette wheel and that represents the winning number. Each slot in the Roulette wheel is represented by a number in the Roulette table layout. The winners are those players whose bets are around or on the winning number in the Roulette table layout. The goal of the Roulette player is to predict what is the winning number in advance. If their prediction is correct, they will be awarded the amount
of money equivalent to their placed bet in the Roulette table. There are different Roulette bets where a player can place his chips. Figure 5.2 shows the majority of Roulette bets where Table 5.4 indicates the payoffs for single chip bets.

![Figure 5.2: Roulette Bets](image)

Table 5.4: Various Roulette bets and their payoffs

<table>
<thead>
<tr>
<th>Bet Name</th>
<th>Payoff</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Inside the Roulette table layout bets:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 number, Straight-up</td>
<td>pays 35 to 1</td>
<td>A in Fig. 2</td>
</tr>
<tr>
<td>2 numbers, Split Bet</td>
<td>pays 17 to 1</td>
<td>B in Fig. 2</td>
</tr>
<tr>
<td>3 numbers, Street Bet</td>
<td>pays 11 to 1</td>
<td>C in Fig. 2</td>
</tr>
<tr>
<td>4 numbers, Corner Bet</td>
<td>pays 8 to 1</td>
<td>D in Fig. 2</td>
</tr>
<tr>
<td>6 numbers, Line Bet</td>
<td>pays 5 to 1</td>
<td>E in Fig. 2</td>
</tr>
<tr>
<td><strong>Outside the Roulette table layout bets:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 numbers, Dozen Bet</td>
<td>pays 2 to 1</td>
<td>F in Fig. 2</td>
</tr>
<tr>
<td>18 numbers, Low or High Bet</td>
<td>pays 1 to 1</td>
<td>G in Fig. 2</td>
</tr>
</tbody>
</table>

Recall that we assumed there is no zero in the Roulette wheel. We further assume that each player has only one chip to place on the Roulette table and the only types of bets available are those listed in Table 5.4. The players need to place their chip in such a way that they predict the winning
number. Assume that one player follows a certain prediction procedure that led them to predict with high probability that the next winning number is one of three numbers. Now, the way those number values spread affects the chances that this player will win given that they have only one chip to place. If those three numbers are close to each other, then their chances to win are high while if those numbers are spread very far from each other then their chances to win are low. Assume that our player knows that the next number to win is in the set \{10,11,12\}. Then they can easily place their bet inside the Roulette table layout by using Street bet to cover the three numbers as they are consecutive. This will allow them to win their money back and get paid “11” times of their bet amount. We next assume that in a different round our player learns that the next winning number is one of the following three numbers: \{1,18,32\}. Given that they have only one chip to place, their chances of winning are much smaller than in the first case. Since none of the available kinds of bets can cover all three numbers using different kinds of bet from Table 5.4, the player has to make a choice and needs luck in order to win. We notice that our player learns the same number of values to win next round but their chances of winning depends really on how these values are positioned in relation to each other. That leads us to say that our player in the first set of values knows much more than in the second set. However, the player might feel that what they really need to know is where to place the chip; in that sense they certainly know more in the first case than in the second.

Let us measure the player knowledge in the different two rounds. Traditionally, the amount of information is expressed by Shannon’s Entropy is:

\[
H(X) = \sum_{i=1}^{n} p(x_i) \cdot \log\left(\frac{1}{p(x_i)}\right)
\]

where $H$ is Shannon’s entropy that measures the player’s uncertainty about $X$, $n$ is the number of values, and each value has a given probability $p(x_i)$ such that $p(x_i) \geq 0$ and the probabilities for all values add up to 1. According to Shannon’s entropy, we need “1.585” bits to represent the player’s knowledge about the confidential variable $X$ or the winning number.
The entropy is the same in both cases, \{10,11,12\} and \{1,18,32\}, which implies that the amount of information, that is, the knowledge the player has, is the same in both cases.

There is an interesting analogy between a player in the Roulette scenario and an intruder to data privacy. The intruder is trying to unlawfully disclose confidential information from a data warehouse. They use all the available information they can get from the data warehouse, as well as any external knowledge they may have. At the end of their analysis, the intruder limits their suspicion to a certain data value. Shannon’s entropy helps us to measure the intruder’s uncertainty about these values, but does not really take into consideration how far or close these values are from each other.

Recall from Section 5.1 that in statistical databases, only aggregate queries such as: COUNT, MAX, MIN, SUM, STDEV, etc., are permitted. Queries that returns the exact value of a confidential data are not usually permitted. However, by analysing the answers to statistical queries, it may be possible make inference about the values of individual records [15]. Assume in our scenario here that we manage a university database in Table 5.1 and that there is an intruder who is trying to unlawfully disclose confidential information from the database. In what follows, we present two examples where an intruder can perform queries on AcademicStaff table. We then try to measure the intruder’s knowledge in the two examples by using Shannon’s entropy.

In the first example, Table 5.5 shows the queries submitted by the intruder and the database response to them. As a result of having two females in the table, Layla and Angela, the intruder learns that Layla’s salary is one of the two values: It is either 107,000 or 50,000.

<table>
<thead>
<tr>
<th>Q1:</th>
<th>SELECT MAX(Salary) FROM AcademicStaff WHERE Sex = &quot;F&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1:</td>
<td>Maximum salary = 107,000</td>
</tr>
<tr>
<td>Q2:</td>
<td>SELECT AVG(Salary) FROM AcademicStaff WHERE Sex = &quot;F&quot;</td>
</tr>
<tr>
<td>A2:</td>
<td>Average salary = 78,500</td>
</tr>
</tbody>
</table>

Table 5.5: Example 1

In the second example, Table 5.6 shows the queries submitted by the
intruder and the database response to them. There are two individuals aged 37 in the table, Qays and Tony. The intruder knows that Qays’s salary is either 80,000 or 77,000.

<table>
<thead>
<tr>
<th>Q1: SELECT MAX(Salary) FROM AcademicStaff WHERE Age = 37</th>
<th>A1: Maximum salary = 80,000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q2: SELECT AVG(Salary) FROM AcademicStaff WHERE Age = 37</td>
<td>A2: Average salary = 78,500</td>
</tr>
</tbody>
</table>

Table 5.6: Example 2

Again, we will use Shannon’s entropy to measure the difference in the intruder’s knowledge in example 1 and 2. Shannon’s entropy indicates that the amount of information that the intruder has is the same in both example 1 and 2. However, we argue that the intruder knowledge in example 1 is more than his knowledge in example 2. In example 2, he almost learns the exact value of salary. It is either 80,000 or 77,000 which is so close to each other. While in example 1, he learns less as the salary was either 107,000 or 50,000. This highlights the need for more accurate measure to capture such difference between the two examples as Shannon’s entropy failed to accurately capture that difference.

5.4 A Novel Entropy-Based Measure

In this section, we introduce a novel confidentiality measure that is based on entropy. Our proposed measure is more accurate than the one introduced by Onganian and Domingo-Ferrer [76] (See Section 5.1). We argue that this novel measure is satisfactory for data confidentiality, that is, for the amount of information that an intruder is able to gain about a particular confidential value.

Recall from Section 5.1 that a database is said to be compromised if a confidential value is disclosed [71]. The compromise can be exact or approximate. In the exact compromise, an intruder learns the exact confidential value. Shannon’s entropy can be considered a satisfactory measure for the disclosure risk that is related to the exact compromise. However, in the ap-
proximate compromise, we argue that Shannon’s entropy does not express precisely the intruder’s knowledge about a particular confidential value. We introduce a notion of privacy for the so-called approximate compromise range ($\varepsilon$). In the approximate compromise an intruder learns that the confidential value $X$ lies within a range $\varepsilon$:

$$X \in [X_0 - \frac{\varepsilon}{2}, X_0 + \frac{\varepsilon}{2}]$$

Let us take an example to clarify the notion of $\varepsilon$. Assume in one scenario that the intruder learns that the confidential value $X$ is either “5” or “6” with equal probability. The approximate compromise range ($\varepsilon$) between “5” and “6” is $|6 - 5| = 1$, then $X$ can be expressed as:

$$X \in [5, 6]$$

Assume in another scenario that the intruder learns that the confidential value $X$ is either “1” or “10” with equal probability. The approximate compromise range ($\varepsilon$) between “1” and “10” is $|10 - 1| = 9$, then $X$ can be expressed as:

$$X \in [1, 10]$$

It is obvious that the approximate compromise range ($\varepsilon$) is different in these two scenarios and it is larger when the values in the intruder’s knowledge about $X$ are far from each other. Assume that the intruder knows two employees, Alex and Bob, who work for a particular company. Assume further that the intruder came to a conclusion that Alex’s annual income is either “50,000” or “51,000” and that Bob’s annual income is either “170,000” or “40,000”. Obviously, the intruder knows more about Alex’s salary than Bob’s salary. Shannon’s entropy does not help us to measure precisely the the difference in the intruder’s uncertainty about the confidential value in terms of the approximate compromise. It does not take into consideration
how far or close these values are from each other. Our proposed measure is based on “ε” to capture the effect of the range in the approximate compromise and represents the entropy $H$ as a function of $ε$. The graphs in Figure 5.3 correspond to the intruder’s uncertainty in the above example where $X \in [5, 6]$ and $X \in [1, 10]$. We notice that the area is smaller in the case $X \in [5, 6]$ as the intruder can perform approximate compromise within the approximate compromise range “1” while the range for the other case is “9”.

Each instance of intruder’s knowledge about a given confidential value will give us $H$ as a function of approximate compromise range $ε$. We can then evaluate intruder’s uncertainty for any given $ε$. In particular, we use $H_0$ to denote intruder’s uncertainty in the case of exact compromise, that is, approximate compromise range of “0” and we call it initial entropy. Additionally, in what follows we shall often examine the area $(A)$ determined as an integral:

$$A = \int_0^{ε_{max}} H(ε)$$

where “$ε_{max}$” is the absolute difference between maximum and minimum value with probability greater than zero. In the above example (Figure 5.3), we see that the two instances have the same initial entropy ($H_0 = 1$) but they differ greatly in terms of the area $A$. In other words, the disclosure
risks are equal for exact compromise, but the second instance (1, 10) is much more resistant against approximate compromise than the first one (5, 6).

The examples introduced above show how essential is the approximate compromise range in expressing the intruder’s uncertainty. Let us explain in general how $H(\varepsilon)$ is calculated. We introduce a “window” of length $\varepsilon$. When a window “covers” two or more values, then they are replaced with a single value whose probability is equal to the sum of probabilities of all the values covered by the window. The problem is how to cover values with window of length $\varepsilon$ or less such that the resulting entropy is minimised.

### 5.4.1 A numerical example

We introduce a numerical example to show how the matter of computing $H(\varepsilon)$ becomes a complex issue as there are more values involved.

Consider as input a set of values $X_i$ and for each $X_i$ a given probability $P_i$, where $P_i \geq 0$ and $\Sigma P_i = 1$. In order to produce our security measure and compute the area, we need to calculate $H(\varepsilon)$ for each $\varepsilon$. In our example, an intruder learns that a confidential variable $X \in [1-10]$ has four possible values with probabilities as follow:

<table>
<thead>
<tr>
<th>$X_1$ = 1</th>
<th>$X_2$ = 3</th>
<th>$X_3$ = 8</th>
<th>$X_4$ = 9</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$ = 0.15</td>
<td>$P_2$ = 0.1</td>
<td>$P_3$ = 0.7</td>
<td>$P_4$ = 0.05</td>
</tr>
</tbody>
</table>

1. We start by calculating initial entropy $H(\varepsilon = 0)$, as the approximate compromise range $\varepsilon$ is initially “0”; this is equivalent to Shannon’s entropy for given events and their probabilities:

$$H(X) = \sum_{i=1}^{4} p(x_i) \cdot \log\left(\frac{1}{p(x_i)}\right)$$
\[ H(\varepsilon = 0) = 1.319 \]

2. We calculate the minimum entropy when \( \varepsilon = 1 \), that is, \( H(\varepsilon = 1) \). We cover \( X_3 \) and \( X_4 \) with window of length 1 and we obtain a combined value \( X_{3,4} = 8.5 \) with probability \( P_{3,4} = P_3 + P_1 = 0.75 \):

\[
H(X) = p_1 \cdot \log\left(\frac{1}{p_1}\right) + p_2 \cdot \log\left(\frac{1}{p_2}\right) + (p_3 + p_4) \cdot \log\left(\frac{1}{p_3 + p_4}\right)
\]

\[ H(\varepsilon = 1) = 1.054 \]

3. We calculate the minimum entropy when \( \varepsilon = 2 \), that is, \( H(\varepsilon = 2) \). We cover \( X_3 \) and \( X_4 \) with window of length 1 and \( X_1 \) and \( X_2 \) with window of length 2:

\[
H(X) = (p_1 + p_2) \cdot \log\left(\frac{1}{p_1 + p_2}\right) + (p_3 + p_4) \cdot \log\left(\frac{1}{p_3 + p_4}\right)
\]

\[ H(\varepsilon = 2) = 0.811 \]

4. Note that when when \( \varepsilon = 3 \) or 4, we can not do better than for \( \varepsilon = 2 \).

5. We calculate the minimum entropy when \( \varepsilon = 5 \), that is, \( H(\varepsilon = 5) \). Here we have 2 options so as how to cover the values with window of length 5 or less. We choose with the option that gives us the minimum entropy, that is, covering \( X_3 \) and \( X_4 \) with window of length 1 and \( X_1 \) and \( X_2 \) with window of length 2:
5.4 A Novel Entropy-Based Measure

\[
H(X) = p_1 \cdot \log \left( \frac{1}{p_1} \right) + (p_2 + p_3) \cdot \log \left( \frac{1}{p_2 + p_3} \right) + p_4 \cdot \log \left( \frac{1}{p_4} \right)
\]

\[
H(\varepsilon = 5) = 0.884
\]

\[
H(X) = (p_1 + p_2) \cdot \log \left( \frac{1}{p_1 + p_2} \right) + (p_3 + p_4) \cdot \log \left( \frac{1}{p_3 + p_4} \right)
\]

\[
H(\varepsilon = 5) = 0.811
\]

6. We calculate the minimum entropy when \( \varepsilon = 6 \), that is, \( H(\varepsilon = 6) \). We cover \( X_2, X_3 \) and \( X_4 \) with window of length 6:

\[
H(X) = p_1 \cdot \log \left( \frac{1}{p_1} \right) + (p_2 + p_3 + p_4) \cdot \log \left( \frac{1}{p_2 + p_3 + p_4} \right)
\]

\[
H(\varepsilon = 6) = 0.610
\]

7. We calculate the minimum entropy when \( \varepsilon = 7 \), \( H(\varepsilon = 7) \). We cover \( X_1, X_2 \) and \( X_3 \) with window of length 7:
5.5 An Algorithm to Compute $H(\varepsilon)$

Recall that our measure is based on representing the entropy $H$ as a function of $\varepsilon$. This section describes how to calculate the optimal entropy for each $\varepsilon$. To do so, we introduce a dynamic programming algorithm. Dynamic
5.5 An Algorithm to Compute $H(\varepsilon)$

programming is used to solve optimization problems and it is based on solving a series of subproblems. The results of those subproblems are used to calculate solutions to a larger problem. Dynamic programming algorithms are often viewed as “filling a table”. Each cell of the table correspond to a solution of a subproblem [27].

We will be given as input a set of values $X_i$ and each $X_i$ has a given probability $P_i$ where $P_i \geq 0$ and and $\Sigma P_i = 1$. In order to produce our security measure, we need to calculate $H(\varepsilon)$ for each $\varepsilon$. We need to solve the problem of finding minimum entropy for a given epsilon.

Table 5.7: Structure of “filling a table” by our algorithm

<table>
<thead>
<tr>
<th>$\varepsilon_1$</th>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$X_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H(X_1, \varepsilon_1)$</td>
<td>$H(X_2, \varepsilon_1)$</td>
<td>$\cdots$</td>
<td>$H(X_n, \varepsilon_1)$</td>
<td></td>
</tr>
<tr>
<td>$H(X_1, \varepsilon_2)$</td>
<td>$H(X_2, \varepsilon_2)$</td>
<td>$\cdots$</td>
<td>$H(X_n, \varepsilon_2)$</td>
<td></td>
</tr>
<tr>
<td>$H(X_1, \varepsilon_3)$</td>
<td>$H(X_2, \varepsilon_3)$</td>
<td>$\cdots$</td>
<td>$H(X_n, \varepsilon_3)$</td>
<td></td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td></td>
</tr>
<tr>
<td>$H(X_1, \varepsilon_n)$</td>
<td>$H(X_2, \varepsilon_n)$</td>
<td>$\cdots$</td>
<td>$H(X_n, \varepsilon_n)$</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5.4: Our security measure for the numerical example.
5.5 An Algorithm to Compute $H(\varepsilon)$

We break the problem into stages (rows) and states (columns). Each row in the table corresponds to a stage or epsilon. For each epsilon, we need to break the problem of finding $H(\varepsilon)$ into subproblems (states), and the solution of the main problem will be formed by combining the solutions to these subproblems. We order the given set of values as $X_1 < X_2 < X_3 < \cdots < X_n$. Column “$i$” in the table corresponds to subproblems containing values $X_1, X_2, \cdots, X_i$. For a given row (stage) in the table, each cell in this row is viewed as a subproblem $H(i, \varepsilon)$ of the original problem $H(\varepsilon)$.

For a given stage and state, $H(i, \varepsilon)$ is computed by the following recurrence relation:

$$H(i, \varepsilon) = \min\left[(H(i - 1, \varepsilon) + a_i), (H(i - 2, \varepsilon) + a_{i-1}), \ldots, (H(j - 1, \varepsilon) + a_j)\right]$$

where:

$$a_j = \left(\sum_{k=j}^{i} p_k\right) \cdot \log\left(\frac{1}{\sum_{k=j}^{i} p_k}\right),$$

$X_i - X_j \leq \varepsilon$ and $X_i - X_{j-1} > \varepsilon,$

$i, j \in [1, n]$ and $i \geq j,$

$H(0, \varepsilon) = 0,$

$H(1, \varepsilon) = p_1 \cdot \log\left(\frac{1}{p_1}\right),$
5.5 An Algorithm to Compute \(H(\varepsilon)\)

**Input:** \(X[\cdot]\): a set of integer values in ascending order
\(P[\cdot]\): a set of probabilities corresponding to the above integer values.

**Output:** \(H(\varepsilon)\)
\[
H_0 \leftarrow \sum_{i=1}^{n} p(x_i) \cdot \log\left(\frac{1}{p(x_i)}\right);
\]

**Algorithm 1**

foreach \(\varepsilon\) do

\[
H(0, \varepsilon) \leftarrow 0;
\]
\[
H(1, \varepsilon) \leftarrow P_1 \cdot \log\left(\frac{1}{P_1}\right);
\]

for \(i \leftarrow 2\) to \(n\) do

\[
j \leftarrow i;
\]

\[
P_{partial} \leftarrow 0;
\]

\[
H(i, \varepsilon) \leftarrow H(i, \varepsilon - 1);
\]

while \((X_i - X_j \leq \varepsilon)\) and \((j \neq 0)\) do

\[
P_{partial} \leftarrow P_{partial} + P_j;
\]

\[
H_{temp} \leftarrow P_{partial} \cdot \log\left(\frac{1}{P_{partial}}\right) + H(j - 1, \varepsilon);
\]

if \(H_{temp} < H(i, \varepsilon)\) then

\[
H(i, \varepsilon) \leftarrow H_{temp};
\]

end

\[
j \leftarrow j - 1;
\]

end

\[
H(\varepsilon) \leftarrow H(n, \varepsilon);
\]

**Display:** \(H(\varepsilon)\)

Table 5.8: A table filled by our algorithm for the numerical example

<table>
<thead>
<tr>
<th>(\varepsilon_1 = 1)</th>
<th>(X_1 = 1)</th>
<th>(X_2 = 3)</th>
<th>(X_3 = 8)</th>
<th>(X_4 = 9)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.411</td>
<td>0.743</td>
<td>1.103</td>
<td>1.054</td>
<td></td>
</tr>
<tr>
<td>(\varepsilon_2 = 2)</td>
<td>0.411</td>
<td>0.5</td>
<td>0.860</td>
<td>0.811</td>
</tr>
<tr>
<td>(\varepsilon_3 = 5)</td>
<td>0.411</td>
<td>0.5</td>
<td>0.668</td>
<td>0.811</td>
</tr>
<tr>
<td>(\varepsilon_4 = 6)</td>
<td>0.411</td>
<td>0.5</td>
<td>0.668</td>
<td>0.610</td>
</tr>
<tr>
<td>(\varepsilon_5 = 7)</td>
<td>0.411</td>
<td>0.5</td>
<td>0.070</td>
<td>0.286</td>
</tr>
<tr>
<td>(\varepsilon_6 = 8)</td>
<td>0.411</td>
<td>0.5</td>
<td>0.070</td>
<td>0.0</td>
</tr>
</tbody>
</table>

We apply “Algorithm 1” to compute \(H(\varepsilon)\) for the previously introduced numerical example. Table 5.8 shows in detail, for the numerical example, the computed values for different stages and states. The last column in the table represents the minimum entropy for a given epsilon \((H(\varepsilon))\). Recall
that Figure 5.4 shows \( H(\varepsilon) \) and the computed area (\( A \)) for the numerical example. Figure 5.5 shows the results of applying our security measure to the set of all values in \([1,10]\) with equal probability. Figure 5.6 shows the results of applying our security measure to the Roulette example in Section 5.3.

Figure 5.5: Our Security Measure: for the equally likely values \([1,10]\).

Figure 5.6: Our Security Measure: for the Roulette example.
5.5 An Algorithm to Compute $H(\varepsilon)$

![Figure 5.7: Our Security Measure: for (77K, 80K) and (50K, 107K).](image)

The graphs in Figure 5.7 correspond to an intruder’s uncertainty about a confidential attribute “$X$” where $X \in [77K, 80K]$ and $X \in [50K, 107K]$. We notice that the area is smaller in the case $X \in [77K, 80K]$ as the intruder can perform approximate compromise within the approximate compromise range “3K” while the range for the other case is “57K”. To present these findings in a more compact form, we use average entropy instead of area. Assume that a confidential value “$C$” is in the range $[lower, upper]$ and that the computed area according to our measure is $A$. We then compute average entropy as $\frac{A}{\text{upper} - \text{lower}}$ where “upper − lower” is actually the largest possible “$\varepsilon$” that we might have. In the rest of this thesis we shall use the average entropy instead of the area. It is worth to mention that, in the next chapter, our measure is computed as a conditional entropy $H_{SK}(\varepsilon)$ for the given intruder’s supplementary knowledge SK.

In the next chapter we apply our proposed security measure to a few common Statistical Disclosure Control (SDC) techniques. We first describe in detail the three used SDC techniques, which are Sampling, Query Restriction, and Noise Addition. We use three datasets to show how this measure can be used to evaluate the security mechanisms for protecting privacy in statistical databases and data mining. The experiments are explained in detail and followed by results and diagrams.
Chapter 6

Comparative Study of SDC Techniques

The previous chapter of this thesis proposes a novel entropy-based measure of disclosure risk. It also presents a dynamic programming algorithm for calculating the disclosure risk. In this chapter we apply our proposed security measure to a few common Statistical Disclosure Control (SDC) techniques.

This chapter is organised as follows. In section 6.1, we introduce the three SDC techniques to perform our comparative study on. Section 6.2 describe the three data sets that are used in our comparative study. The experiments are explained in section 6.3 following by results and diagrams in section 6.4. A discussion about the study results is introduced in section 6.5.

6.1 Background on The Chosen SDC Techniques

There are different statistical disclosure control (SDC) techniques available in order to protect released microdata files against disclosure of confidential information [111, 5]. These techniques are divided into two broad categories: restriction and modification techniques. Restriction techniques do not release database to a user but rather provide a query access [13, 111, 34, 5].
6.1 Background on The Chosen SDC Techniques

On the other hand, modification techniques involve some kind of alternation of the original data set, before it is released to statistical users. This includes noise addition, data swapping, aggregation, and sampling [16, 111, 5].

In what follows we briefly explore the main principles underlying the chosen SDC techniques for our comparative study.

6.1.1 Sampling

One of the common ways to release microdata files by offices of national statistics and census is the so-called sampling. Producing a sample out of the population is based on one of various sampling techniques [77]. These techniques may be classified as follows.

1. In terms of sample size, we can divide sampling techniques into:
   - **Fixed Sample Size:** where the produced sample has a specified size.
   - **Variable Sample Size:** where the sample has a variable size. For example, in binomial random samples the sample size is a binomial random variable [77].

2. In terms of probability of inclusion, we can divide sampling techniques into:
   - **Equal Probability of Inclusion:** Some sampling techniques assign an equal probability for each record in the population to be included in the produced sample. Examples of such techniques are:
     - Simple random sampling: In simple random sampling without replacement, duplicates are not allowed unlike in simple random sample with replacement.
     - Stratified random sampling: A stratified random sample is produced by partitioning the records in the original population based on one of the attributes in the database and then using simple random sample to obtain records of a specified size from each strata [77]. In stratified sampling, each
record still has equal probability to be included in the sample. However, unlike in simple random sampling, not all random samples are equally likely.

- **Weighted Probability:** A weighted random sample is obtained when each record in the original population has specified probability to be included in the produced sample. Examples of sampling techniques that are based on weighted random sampling are a probability proportional to size and a dollar unit sample [77].

Sampling has also been used as an essential disclosure control method [110]. It is a modification technique [51] where a database owner releases a subset of records from the original microdata file (population). The released records are in original, unperturbed form. The answers to the queries posed by users are approximate and not exact as the sample is just a subset of the population. A compromise may happen if a database user successfully matches a record in the released sample to a record in a published file [112].

One of the first uses of sampling in the context of statistical disclosure control goes back to early 80’s when Denning introduced random sample query control [33]. A user submits a query to a DBMS to obtain a statistical value. The DBMS finds the set of records from the original dataset that satisfy the conditions of the user’s query. This set of records is typically called query set. Random sample query control is based on generating a sample from the query set randomly based on a selection function [33]. The DBMS then returns a statistical value that is computed over the randomly sampled query set instead of the query set itself. The method is designed in such a way so as to preserve the consistency of the statistics and to guarantee that the same query will always produce an identical sample from the query set.

Resampling [24, 35, 36] is another disclosure control method that is based on sampling and it is considered a generalization of random-sample methods. In resampling, simple random sample with replacement is used to generate independently a number of samples of the values of a confidential numerical attribute in the original database. The size of each generated sample is identical to the original size. Each sample is then ordered inde-
pendently based on the same criteria, for example sorting the values of the confidential attribute in ascending order. The original is ordered in the same way. The confidential attribute value in the first record in the original is replaced with the average value that is computed over all confidential attribute values of the first record of all samples. The process of replacing the confidential attribute value with the average is repeated for all records in the original. The modified original is then released to users.

In the literature, measuring the disclosure risk that is result from releasing sample files by various organizations has received much attention from researchers [110]. Disclosure risk estimation requires assessing the uniqueness of records in the released sample and in the population. Several methods [111, 112] have been proposed to estimate the disclosure risk. Winkler [112] refers to these methods as sample-unique-population-unique (SUPU) methods. Disclosure risk involved in releasing sample files is discussed in the previous chapter.

Some agencies make sampling weights available to users. Willenborg and De Waal [110] show that publishing sampling weights provides additional information that helps intruders in their analysis. This can be avoided by, for example, adding noise to the weights before publishing them [110].

### 6.1.2 Query Restriction

Query Restriction techniques are based on restricting the amount of information that is available to a database user [34, 5, 13]. To facilitate statistical research, a database user is allowed to submit statistical queries via an access channel such as in Remote Access Facilities (RAF) or in Data Laboratories (DL) [103]. Statistical queries include: SUM, COUNT, AVG, MAX, MIN, to mention some [95]. A DBMS has the choice to answer or decline the user queries. It needs to ensure that the answered queries will not help the user to infer or disclose any confidential data in the database. Query Restriction does not alter the original database. The DBMS decides wether to provide an answer to a user query or to decline it to prevent an exact compromise [34]. There are various proposed techniques that can be used by the DBMS to make this decision [5, 13]:
• Query Set Size control: an intruder submits a query to a DBMS. The DBMS finds all records in the database that satisfy the constraints in the submitted query. This group of records are called query set. Query set size control [34, 5] sets a lower and upper bounds for the query set size. In order to answer a query, the query set size must fall into the permitted range between the given lower and upper bound.

• Query Set Overlap: Query set overlap technique [34, 5] makes sure that the query set of a new submitted query does not overlap with query sets of queries submitted previously by the same user. Query set overlap has a downside that it needs to keep an updated profile for each user. Additionally, it is considered ineffective against attacks that are launched by cooperating users [95].

• Maximum Order Control: In this technique the decision whether to answer a query is based on the total number of attributes that is involved in the query [34, 5].

• Partitioning: Partitioning technique [20] is based on physically reorganising database records into mutually exclusive groups. A group of single record is not allowed. The decision to answer a user query is made based on if the query involves each group as a whole [95]. This measure is in some ways very similar to the concept of $k$-anonymity.

• Auditing: In this technique [21], the DBMS keeps logs of all queries that are asked by any database user. The decision to answer a new query is based on verifying that this query, together with previously asked queries by that party or a group of collaborating parties, does not pose a danger and lead to a compromise of confidential data.

We next describe in more detail a technique that we use in our experiment and which is based on the so-called Audit Expert [21, 14]. Audit expert aims to prevent a database compromise that may result from answering a set of queries such as SUM queries. Consider a database that consists of $n$ records. Chin and Ozsoyoglu [21] thought of a SUM query as a linear equation,

\[
\lambda_1 c_1 + \lambda_2 c_2 + \ldots + \lambda_n c_n = q
\]  

(6.1)
In other words, each answered query reveals a linear equation \( \sum_{i=1}^{n} \lambda_i c_i = q \) with \( \lambda_i \in \{0, 1\} \). If the record \( i \) is in the query set then \( \lambda_i = 1 \), otherwise \( \lambda_i = 0 \); the value of the confidential attribute for the individual \( i \) in the database is expressed by \( c_i \); \( q \) is the response to the SUM query. Moreover, a set of SUM queries can be expressed as a system of linear equations,

\[
BC = Q
\]  

(6.2)

where \( B \) is the query basis matrix, \( C \) is the column vector of the unknown values of the confidential attribute \( C \), and \( Q \) is the column vector of answers to user queries. The Audit Expert only keeps track of the set of linearly independent queries. A set of queries is said to be linearly independent if the corresponding row vectors of the query basis matrix are linearly independent. That means if the DBMS answered \( k \) linearly independent queries then the Audit Expert stores corresponding query basis matrix \( B_k \) with size \( k \times n \).

In order to prevent a compromise, we must make sure that \( k \leq n - 1 \) [14].

A query is considered a range query if its query set is a range vector [22]. A query set is considered a range vector if it can be specified by two integers \( \alpha \) and \( \beta \) such that \( 1 \leq \alpha \leq \beta \leq n + 1 \). If the record \( i \) is in the query set but with \( \alpha \leq i < \beta \) then \( \lambda_i = 1 \), otherwise \( \lambda_i = 0 \).

### 6.1.3 Noise Addition

Noise Addition is a modification technique [5]. In this approach, an alteration is made either to original databases or to statistics that are computed over the original database [95]. In the former, data perturbation, the original values in the database is added to or multiplied by a random value that is drawn from a probability distribution. The altered database is then made available to public. One of the most common approaches to add noise to original microdata files is by using additive noise techniques [59, 102, 40, 60, 116]. The latter, output perturbation, does not altered the original database. It modifies the computed statistics over the original database. In output perturbation, no database is released to public but only the modified statistics [95]. Random sample query control [33] is considered an example of output perturbation. In contrast to Query Re-
6.1 Background on The Chosen SDC Techniques

Noise Addition techniques preserve the availability but not the exactness of statistics [13]. They give database users the chance to fully access microdata files but the modified ones. This mean all queries to a database are answered and none is declined. Therefore, an answer to any user query is approximate and not exact as it is performed on the altered microdata. Despite the fact that noise addition affects the quality of answers to user queries but its strength comes from the fact that it is easy to implement and its low running cost [13].

Noise can be added to the original microdata file in different ways by using different techniques. It can be applied to only sensitive attributes, Additive Noise [59, 102, 40, 60, 116], or for all attributes in the original file, General Additive Data Perturbation (GADP) [73]. Moreover, it is added in different way based on the nature of attributes. Attributes can be classified as numerical or categorical. Numerical attributes can be distinguished by the fact that its values have a natural ordering; otherwise, attributes are classified as categorical [13]. Post RAndomization Method (PRAM) [32] is an example of noise addition techniques that is used to add noise for categorical attributes.

One of the most common approaches to add noise to original microdata file is by using additive noise techniques [59, 102, 40, 60, 116]. In this approach, an amount of error “e” is added to a confidential attribute $X$ and to obtain a perturbed value of this attribute $X' = X + e$ [74]. The error ($e$) is drawn randomly from some probability distribution in such a way that when it is added to the original value, it still preserves the underlying probability properties [6, 56]. Some of most commonly used probability distributions for this purpose are uniform, Gaussian, and binomial. Determining the average amount of error to be added to the original values is a difficult task. Adding large amount of noise may lead to poor data quality and statistics while adding a very small amount of noise may make it easy for an intruder to infer the original values [95].
6.2 The Used Datasets

This comparative study is performed on three different datasets. In what follows we describe the used data sets throughout this chapter.

6.2.1 PUMS dataset

Our first data set is the Public Use Microdata Sample (PUMS) [107]. It is a sample of the actual responses to the American Community Survey (ACS) and is offered by the U.S. Census Bureau. We have chosen Illinois state sample for the year 2006 as a dataset. The dataset consists of 2500 records. Each record is described by 7 selected attributes that are shown in Table 6.1. The first column in the table shows the selected attributes. The minimum and maximum value for each attribute are shown in the second and third columns. For example, the minimum value for “Salary” is 10 while the maximum is 250. The fourth column indicates the data type for the corresponding attribute and it can be one of the following: Categorical, Numerical-Integer, or Numerical-Real. The next column, Rounding, indicates if the corresponding attribute values have been rounded. For example, for the attribute “Salary”, we round values to the nearest 10 K. Symbol “.” indicates no rounding. The last column contains number of values in the actual domain for the corresponding attribute after rounding. We select the first attribute, “Salary”, to be sensitive (confidential) and hence we need to keep its value confidential.

Table 6.1: The selected attributes from PUMS dataset

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Min.</th>
<th>Max.</th>
<th>Data Type</th>
<th>Rounding</th>
<th>No of Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Salary</td>
<td>10</td>
<td>250</td>
<td>Numerical – Integer</td>
<td>10</td>
<td>25</td>
</tr>
<tr>
<td>Age</td>
<td>16</td>
<td>84</td>
<td>Numerical – Integer</td>
<td>–</td>
<td>69</td>
</tr>
<tr>
<td>Sex</td>
<td>1</td>
<td>2</td>
<td>Categorical</td>
<td>–</td>
<td>2</td>
</tr>
<tr>
<td>Education</td>
<td>1</td>
<td>16</td>
<td>Categorical</td>
<td>–</td>
<td>16</td>
</tr>
<tr>
<td>Industry</td>
<td>1</td>
<td>18</td>
<td>Categorical</td>
<td>–</td>
<td>18</td>
</tr>
<tr>
<td>Occupation</td>
<td>1</td>
<td>25</td>
<td>Categorical</td>
<td>–</td>
<td>25</td>
</tr>
<tr>
<td>Work Travel Time</td>
<td>1</td>
<td>177</td>
<td>Numerical – Integer</td>
<td>–</td>
<td>177</td>
</tr>
</tbody>
</table>
6.2 The Used Datasets

6.2.2 WBC dataset

The Wisconsin Breast Cancer (WBC) dataset [113] is offered by UCI Machine Learning Repository. It consists of 600 records. Each record is described by 10 attributes. The attribute information is shown in Table 6.2. We consider the first attribute, “ClumpThickness”, as the sensitive attribute. Notice that the last attribute is a class attribute that has 2 possible values (2 and 4) and it is treated as a categorical attribute.

Table 6.2: The selected attributes from WBC dataset

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Min.</th>
<th>Max.</th>
<th>Data Type</th>
<th>Rounding</th>
<th>No of Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clump Thickness</td>
<td>1</td>
<td>10</td>
<td>Numerical – Integer</td>
<td>–</td>
<td>10</td>
</tr>
<tr>
<td>Uniformity of Cell Size</td>
<td>1</td>
<td>10</td>
<td>Numerical – Integer</td>
<td>–</td>
<td>10</td>
</tr>
<tr>
<td>Uniformity of Cell Shape</td>
<td>1</td>
<td>10</td>
<td>Numerical – Integer</td>
<td>–</td>
<td>10</td>
</tr>
<tr>
<td>Marginal Adhesion</td>
<td>1</td>
<td>10</td>
<td>Numerical – Integer</td>
<td>–</td>
<td>10</td>
</tr>
<tr>
<td>S. E. Cell Size</td>
<td>1</td>
<td>10</td>
<td>Numerical – Integer</td>
<td>–</td>
<td>10</td>
</tr>
<tr>
<td>Bare Nuclei</td>
<td>1</td>
<td>10</td>
<td>Numerical – Integer</td>
<td>–</td>
<td>10</td>
</tr>
<tr>
<td>Bland Chromatin</td>
<td>1</td>
<td>10</td>
<td>Numerical – Integer</td>
<td>–</td>
<td>10</td>
</tr>
<tr>
<td>Normal Nucleoli</td>
<td>1</td>
<td>10</td>
<td>Numerical – Integer</td>
<td>–</td>
<td>10</td>
</tr>
<tr>
<td>Mitoses</td>
<td>1</td>
<td>10</td>
<td>Numerical – Integer</td>
<td>–</td>
<td>10</td>
</tr>
<tr>
<td>Class</td>
<td>2</td>
<td>4</td>
<td>Categorical</td>
<td>–</td>
<td>2</td>
</tr>
</tbody>
</table>

6.2.3 WINE dataset

The Wine Recognition data set (WINE) [10] is offered by UCI Machine Learning Repository. It consists of 178 records. Each record is described by 14 attributes. The attribute information is described in Table 6.3. The first attribute, “Magnesium”, is the sensitive attribute. It is important to note that we “round” attributes with real values by multiplying the attribute values by a multiple of 10, for example, 10, (moves the decimal point one digit to the right), or 100, (moves the decimal point two digits to the right). For example, the attribute “ColorIntensity” takes real values from 10.6 to 30. After rounding it becomes 106 to 300 and hence the number of values is 195.
6.3 The Experiments: Description and Implementation

6.3.1 Sampling

In this experiment, we release a subset of records (sample) from the original microdata file (population). We use a simple random sample without replacement where each record in the original is equally likely to be included in the produced sample and duplicates are not allowed. The produced sample has a constant size specified as a percentage of the original dataset total size and referred to as “sampling factor”. In deciding on the structure of the sampling experiment, we follow work by Truta et al. [105] on disclosure risk measures for sampling. Their measures are not linked to a certain individual but compute the overall disclosure risk for the database.

We consider a scenario where an intruder obtains a copy of the released sample file. The intruder has supplementary knowledge (SK) about an individual whose corresponding record is stored in the original dataset. They do not necessarily know all attribute values for that individual. Their supplementary knowledge can be as limited as one attribute or can be as extensive as all attributes except the confidential one. The intruder performs an anal-

Table 6.3: The selected attributes from WINE dataset

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Min.</th>
<th>Max.</th>
<th>Data Type</th>
<th>Rounding</th>
<th>No of Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Magnesium</td>
<td>70</td>
<td>160</td>
<td>Numerical – Integer</td>
<td>5</td>
<td>19</td>
</tr>
<tr>
<td>Proline</td>
<td>280</td>
<td>1680</td>
<td>Numerical – Integer</td>
<td>5</td>
<td>281</td>
</tr>
<tr>
<td>Alcohol</td>
<td>11.03</td>
<td>14.83</td>
<td>Numerical – Real</td>
<td>100</td>
<td>381</td>
</tr>
<tr>
<td>Malic Acid</td>
<td>0.74</td>
<td>5.80</td>
<td>Numerical – Real</td>
<td>100</td>
<td>506</td>
</tr>
<tr>
<td>Ash</td>
<td>1.36</td>
<td>3.23</td>
<td>Numerical – Real</td>
<td>100</td>
<td>188</td>
</tr>
<tr>
<td>Alcalinity of Ash</td>
<td>10.6</td>
<td>30.0</td>
<td>Numerical – Real</td>
<td>10</td>
<td>195</td>
</tr>
<tr>
<td>Total Phenols</td>
<td>0.98</td>
<td>3.88</td>
<td>Numerical – Real</td>
<td>100</td>
<td>291</td>
</tr>
<tr>
<td>Flavanoids</td>
<td>0.34</td>
<td>5.08</td>
<td>Numerical – Real</td>
<td>100</td>
<td>475</td>
</tr>
<tr>
<td>Nonflavanoid Phenols</td>
<td>0.13</td>
<td>0.66</td>
<td>Numerical – Real</td>
<td>100</td>
<td>54</td>
</tr>
<tr>
<td>Proanthocyanins</td>
<td>0.41</td>
<td>3.58</td>
<td>Numerical – Real</td>
<td>100</td>
<td>318</td>
</tr>
<tr>
<td>Color Intensity</td>
<td>1.3</td>
<td>13.0</td>
<td>Numerical – Real</td>
<td>10</td>
<td>118</td>
</tr>
<tr>
<td>Hue</td>
<td>0.48</td>
<td>1.71</td>
<td>Numerical – Real</td>
<td>100</td>
<td>124</td>
</tr>
<tr>
<td>OD280/OD315</td>
<td>1.27</td>
<td>4.00</td>
<td>Numerical – Real</td>
<td>100</td>
<td>274</td>
</tr>
<tr>
<td>Class</td>
<td>1</td>
<td>3</td>
<td>Categorical</td>
<td>–</td>
<td>3</td>
</tr>
</tbody>
</table>
ysis using the released sample together with the supplementary knowledge with an aim to infer a confidential attribute value corresponding to the individual in question, e.g., salary in PUMS dataset.

Table 6.4: Original Dataset

<table>
<thead>
<tr>
<th>Record No.</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
</tr>
</thead>
<tbody>
<tr>
<td>or1</td>
<td>10</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>or2</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>or3</td>
<td>5</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>or4</td>
<td>1</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>or5</td>
<td>8</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>or6</td>
<td>10</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>or7</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>or8</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>or9</td>
<td>3</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>or10</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>or11</td>
<td>10</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>or12</td>
<td>5</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 6.5: Sample Dataset

<table>
<thead>
<tr>
<th>Record No.</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
</tr>
</thead>
<tbody>
<tr>
<td>sr1</td>
<td>8</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>sr2</td>
<td>1</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>sr3</td>
<td>5</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>sr4</td>
<td>10</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>sr5</td>
<td>10</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>sr6</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

Let us take a look at a working example to clarify the notation that we use to explain our experiment. Table 6.4 shows the original dataset consisting of 12 records representing different individuals. We add the first column (Record No.) to the table to make it easy to refer to different records. We call them original record 1 (or1) to original record 12 (or12). Table 6.5 represents a sample released from the original dataset with a sampling factor “0.5”. Records in the sample are referred to as sample record 1 (sr1) to sample record 6 (sr6). Suppose the third attribute (A3) is the confidential attribute that we need to protect. The intruder is interested in
inferring the third attribute value of an individual whose data is stored in
the original dataset. Suppose they know that the value of the first attribute
\((A1)\) for this individual is “1”. We now present a notation that will be used
throughout our experiment and apply it to this example to make it clear for
the reader:

- \(C\) is the confidential attribute. In our example \(C = A3\).
- \(D\) is the domain of the confidential attribute \(C\) in the original database.
  In our example, \(|D| = 5\).
- \(d_i\) is the \(i^{th}\) value of the confidential attribute in the domain \(D\), where
  \(1 \leq i \leq |D|\). In our example \(1 \leq i \leq 5\). Note that we arbitrarily order
  the values of the confidential attribute in the domain \(D\).
- \(p_i\) is the probability that the confidential attribute value is \(d_i\).
- \(M_s\) is the set of records in the released sample dataset that matches the
  intruder’s supplementary knowledge. In our example \(M_s = \{sr2, sr6\}\)
  as they are the only two records in the sample with \(A1 = 1\).
- \(M_o\) is the set of records in the original dataset that matches the in-
  truder supplementary knowledge; \(M_o = \{or4, or8, or10\}\).
- \(f_{d_i}\) is the frequency of \(d_i\) in \(M_s\), that is, how many times \(d_i\) appears
  in records that belong to \(M_s\). In our example, only the values 3 and 5
  appear once each in \(M_s\) and the pairs \((d_i, f_{d_i})\) are:

  \[
  (1, 0), (2, 0), (3, 1), (4, 0), (5, 1)
  \]

We use our security measure to evaluate how secure is releasing a sample
such as Table 6.5. To be able to do so, we need a set of \(d_i\)’s and their
Corresponding \(p_i\)’s. The following steps are done to obtain \((d_i, p_i)\):

- Finding the number \(|M_s|\) of records in the sample dataset that matches
  the intruder’s SK of some attribute values. In our example, both \(sr1\)
  and \(sr2\) have \(A1 = 1\) so \(|M_s| = 2\).
6.3 The Experiments: Description and Implementation

- Finding the number $|M_o|$ of records in the original dataset that matches the intruder’s SK. In our example, $A1 = 1$ in the records $or4, or8$, and $or10$ so $|M_o| = 3$.

- Determining the probabilities for all $d_i$’s:
  
  - Find $f_{d_i}$. The pairs $(d_i, f_{d_i})$ in our example are:
    
    $$(1, 0), (2, 0), (3, 1), (4, 0), (5, 1)$$

  - Calculate $p_i$ based on the following formula:
    
    $$p_i = \frac{f_{d_i}}{|M_s|} \cdot \frac{|M_s| - |M_o|}{|M_o|} \cdot |D|$$

    where $p_i$ is computed as the probability that the record in question is in the sample and $d_i$ is its confidential value or that the record does not appear in the sample and $d_i$ is its confidential value. Note that for those records that do not appear in the sample the equal probability of all $|D|$ values is assumed. We also assume that $|M_o|$ is a part of the intruder’s supplementary knowledge.

    The pairs $(d_i, p_i)$ in our example are:

    $$(1, \frac{1}{15}), (2, \frac{1}{15}), (3, \frac{6}{15}), (4, \frac{1}{15}), (5, \frac{6}{15})$$

    In our experiment, we use four different sampling factors:

    $$\{0.05, 0.10, 0.20, 0.50\}$$

This is done in order to study the affect of sample size on the security. For each sample size, we generate 30 different sample files. Additionally, we study the affect of the intruder’s supplementary knowledge. We start with supplementary knowledge as little as one attribute and extended it to reach all attributes except the confidential one. For each attribute we performed experiments for all possible values. The results we present in this chapter are the average results, over all 30 samples, all attributes and all values.
6.3.2 Query Restriction

In this experiment, we consider a scenario where an intruder submits a set of range queries to a DBMS. Just like in the case of sampling, we assume that the intruder has supplementary knowledge (SK) about a particular record in the original dataset, which can range from one attribute to all attributes except the confidential one. The intruder performs an analysis using the answers to the submitted queries as well as the supplementary knowledge with an aim to infer a confidential attribute value for the given record, e.g., salary in PUMS dataset.

We assume that the intruder has build a system of linear equations out of the responses to range queries. We use similar notation as in the case of sampling and we give it here in its totality for the convenience of the reader:

- $n$ is a number of records in the original database.
- $C$ is the confidential attribute.
- $c_i$, $1 \leq i \leq n$, is the value of the confidential attribute $C$ in record $i$.
- $D$ is the domain of the confidential attribute $C$ in the database.
- $d_i$ is the $i^{th}$ value of the confidential attribute in the domain $D$, where $1 \leq i \leq |D|$. Note that ordering of the values in the domain $D$ is arbitrary.
- $p_i$ is the probability that the confidential attribute value is $d_i$.
- $M_o$ is the set of records in the original dataset that matches the intruder’s supplementary knowledge.
- $Q = 2l$ is the query set size for the queries a user (i.e., an intruder) is permitted to ask; note that, for simplicity, we only consider even query set sizes;
- $k$ is the number of queries and thus also the number of equations in a system of linear equations and it is given by

$$k = \left\lfloor \frac{2n}{Q} \right\rfloor - 1$$
For example, for a query set size of 2, \( k = n - 1 \). In a dataset of \( n \) records and for the query set size \( Q = 2l \), the range queries available to an intruder are of the form \([1, 2l], [l+1.3l], [2l+1.4l], \ldots, [(m-2)l+1.6l]\), where \( n = m \times l + s \).

For example, the system of linear equations for \( Q = 2 \) is given by:

\[
\begin{align*}
  c_1 + c_2 &= q_1 \\
  c_2 + c_3 &= q_2 \\
  &\vdots \\
  c_{n-1} + c_n &= q_{n-1}
\end{align*}
\]

where \( c_i \) is the value of confidential attribute in record \( i \). Thus the intruder obtains a system of \( k \) linearly independent equations in \( n \) unknowns. To be able to solve the system and completely compromise the database, the intruder needs \( n \) linearly independent equations. Nevertheless, with \( k < n \) linearly independent equations, the intruder is able to find the upper and lower bounds (min, max) for the confidential attribute value in each record in the dataset.

We follow the evaluation of an entropy based measure of disclosure risk presented in [76] and solve two linear programming problems, maximisation and minimisation, to find \( L \) and \( U \), the upper and the lower bound for \( r_i \), the value of the confidential attribute \( C \) in record \( i \) that matches the intruder’s SK. The constraints for the linear programming problems consist of the given system of \( k \) linearly independent equations in \( n \) unknowns, plus a system of inequalities of the form \( r_i \geq d_{\text{min}} \) and \( r_i \leq d_{\text{max}} \), where \( d_{\text{min}} \) and \( d_{\text{max}} \) are the minimum and the maximum value in the domain \( D \). The linear function to be maximised (minimised) is the confidential value in the record \( i \). We use \( L \) and \( U \) for each record that matches the intruder’s SK to find the probability \( p_i \) for each value \( d_i \) in the domain \( D \). We need these probabilities in order to use our security measure to evaluate how secure is answering a set of range queries.

We apply “Algorithm 2” to obtain \( d_i, p_i \):
Output: $d[], p[]$

// Find the set of records in the original dataset that matches the intruder’s SK.

Find $M_o$:

foreach $i$ in $[1, |D|]$ do

| $p_i \leftarrow 0$; |

end

foreach record belonging to $M_o$ do

// Find $L$ and $U$, that is, the lower and the upper bound for the confidential attribute value in the current record.

Find $L$ and $U$:

foreach $i$ such that $d_i$ in $[L, U]$ do

| $p_i \leftarrow p_i + \left( \frac{1}{U-L+1} \cdot \frac{1}{|M_o|} \right)$; |

end

end

Display: $d[], p[]$

Algorithm 2: Finding $d_i$ and their corresponding $p_i$

Let us use the dataset in Table 6.4 as a working example. Assume that the third attribute ($A_3$) is the confidential attribute and the intruder has a supplementary knowledge that $A_1 = 1$ for an individual of concern. Assume further that the intruder submitted a set of range queries and succeeded to build the following $k = 11$ linearly independent equations:

\[
\begin{align*}
  c_1 + c_2 &= (2 + 3) = 5 \\
  c_2 + c_3 &= (3 + 2) = 5 \\
  c_3 + c_4 &= (2 + 5) = 7 \\
  c_4 + c_5 &= (5 + 1) = 6 \\
  c_5 + c_6 &= (1 + 5) = 6 \\
  c_6 + c_7 &= (5 + 2) = 7 \\
  c_7 + c_8 &= (2 + 4) = 6 \\
  c_8 + c_9 &= (4 + 4) = 8 \\
  c_9 + c_{10} &= (4 + 3) = 7 \\
  c_{10} + c_{11} &= (3 + 3) = 6 \\
  c_{11} + c_{12} &= (3 + 2) = 5
\end{align*}
\]
The intruder solves the above system of linear equations. For each record in Table 6.4, the intruder finds the minimum and maximum values ($L$ and $U$) for the confidential attribute ($A_3$) as follows:

- $or1 : A_3 \in [2, 3]$.  
- $or2 : A_3 \in [2, 3]$.  
- $or3 : A_3 \in [2, 3]$.  
- $or4 : A_3 \in [4, 5]$.  
- $or5 : A_3 \in [1, 2]$.  
- $or6 : A_3 \in [4, 5]$.  
- $or7 : A_3 \in [2, 3]$.  
- $or8 : A_3 \in [3, 4]$.  
- $or9 : A_3 \in [4, 5]$.  
- $or10 : A_3 \in [2, 3]$.  
- $or11 : A_3 \in [3, 4]$.  
- $or12 : A_3 \in [1, 2]$.  

The intruder then applies “Algorithm 2” to obtain $d_i$, $p_i$:

$$(1, 0), (2, \frac{1}{6}), (3, \frac{2}{6}), (4, \frac{2}{6}), (5, \frac{1}{6})$$

We run the experiment for 5 different query set size {2, 4, 8, 16, 32}. For each query set size, we shuffle the records in the original dataset to get different systems of linear equations. For each query set size, we produce randomly 30 different systems of linear equations. Just like in the case of sampling, for each attribute we performed experiments for all possible values. The results we present in this chapter are the average results, over all 30 systems of linear equations, all SK attributes and all values.

### 6.3.3 Noise Addition

We consider a scenario where a DBMS alters an original dataset by adding certain level of noise. The noise is added to all attributes in the dataset, sensitive and non-sensitive, categorical and numerical. Additive noise [59,
is used and the amount of noise is drawn randomly from binomial probability distribution as the nature of attributes in our dataset is discrete. The DBMS then releases the perturbed version of the dataset and an intruder obtains a copy of it. Similarly to sampling and query restriction scenarios, the intruder has a supplementary knowledge (SK) about an individual whose corresponding record is stored in the original dataset. We assume that the intruder knows how the noise was added to the original dataset. The intruder performs an analysis using the released perturbed dataset together with the supplementary knowledge with an aim to infer a confidential attribute value, e.g. salary in PUMS dataset, corresponding to the individual of concern. We assume there is only one confidential attribute; it is straightforward to generalize our experiments to cover more than one confidential attribute. We next show how we add noise to a dataset in our experiment. Then we evaluate the security of the perturbed dataset by using our proposed security measure.

Adding noise

We add noise “e” to every single attribute \( X \) in the used datasets. The produced perturbed version value is expressed by \( X' \) where:

\[
X' = X + e
\]

The added noise \( (e) \) is drawn randomly from a binomial probability distribution. The amount of noise is set to be a parameter. We used four different levels of noise: \{25%, 50%, 75%, 100%\}. Let us use the attribute “salary” in PUMS dataset as an example to show how we add noise. The steps are as follows:

1. Before adding noise to the salary, we study the nature of this attribute including: its data type (numerical or categorical), its actual domain, and possible number of values in the actual domain. Salary in PUMS dataset is a “Numerical − Integer” attribute and is rounded to the nearest 10 thousand. The actual data has minimum salary \( d_{\text{min}} = 10000 \) and maximum salary \( d_{\text{max}} = 250000 \). The number of possible
values for salary is computed as follows:

$$|D| = \frac{d_{\text{max}} - d_{\text{min}}}{R} + 1 = \frac{250 - 10}{10} + 1 = 25$$

where $R$ is the Rounding (see Section 6.2.1).

2. We can add different amount of noise. By $k\%$ noise, we mean that the maximum noise $M$ is $k\%$ of $|D| - 1$, rounded to the nearest 2. Since we also include 0 noise, this implies that the total number of values for noise must be odd. In this example we add noise of 100%, maximum noise is 24 and there are 25 different values of noise in $[0 - 24]$. If there were 24 values in the domain of the attribute "salary", the 100% noise would mean that the maximum noise is 22.

![Figure 6.1: Binomial PDF for $n = 24$ and $p = 0.5$](attachment:figure.png)

3. We use binomial probability distribution, see Figure 6.1, to draw random number $G$ in the range $[0 - 24]$ to generate the amount of noise that is introduced to one instance of salary. Since we wish the noise to be in the range

$$[-\frac{\text{maximum noise}}{2} \times R, +\frac{\text{maximum noise}}{2} \times R]$$
we calculate the noise as:

\[ e = \left( G - \frac{M}{2} \right) \times R \]

With a maximum noise of 24 and Rounding of 10000, assume that the random number \( G \) drawn from the binomial distribution is 13, and we obtain \( e = 10000 \). If the value of the original salary is 40000, then the perturbed value will be \( 40000 + 10000 = 50000 \).

4. After adding noise to the original value, we need to guarantee that the perturbed value is located within the attribute domain. For example, when the original salary is 240 and the added noise is 60, the produced perturbed salary value will be 300. This value of salary is currently outside the attribute range ([10, 250]). To avoid that, we calculate the perturbed value \( X' \) as:

\[ X' = (X + e - d_{\min}) \mod (|D| \times R) + d_{\min} \]

For example, for the original value of 240 and noise 60 we have

\[
\begin{align*}
X' &= (240 + 60 - 10) \mod (25 \times 10) + 10 \\
X' &= 50
\end{align*}
\]

**Analysing noise**

We need a set of \( d_i \) and their corresponding \( p_i \) in order to use our security measure to evaluate how secure is releasing a perturbed dataset. “Algorithm 3” shows the steps that are followed in order to analyse noise and obtain a set of \( d_i \) and their corresponding \( p_i \).
Input: Original dataset, Perturbed dataset, and the intruder’s SK
Output: \( d[ ], p[ ] \)

// Initialize probSUM
\( \text{probSUM} \leftarrow 0; \)

// For each perturbed record in the perturbed dataset.
foreach \( \text{ptr} \) do
  // Initialise the probability of the perturbed record.
  \( \text{probPTR} \leftarrow 1; \)
  foreach attribute that belongs to the intruder’s SK do
    // Reconstructing \( G \), the random number drawn from the
    // binomial probability distribution, from the perturbed
    // and the original value of the current attribute
    \( r \leftarrow \left( X - X' \right) + \frac{\text{maximum noise}}{2} \text{ mod } |\text{Domain}|; \)
    // Computing the probability of \( G \), i.e., the probability
    // for obtaining \( G \) successes in \( N \) trials, where \( N \) is the
    // number of values for the current attribute and \( p = 0.5 \)
    \( P(G) \leftarrow \frac{N!}{G!(N-G)!} \cdot p^G (1-p)^{N-G}; \)
    // Contribution of current attribute towards current
    // perturbed record’s probability
    \( \text{probPTR} \leftarrow \text{probPTR} \cdot P(r); \)
  end
  foreach \( d_i \) in the confidential attribute Domain do
    // Reconstructing \( G \), the random number drawn from the
    // binomial probability distribution, from \( C' \), the
    // perturbed value of the confidential attribute
    \( r \leftarrow \left( C' - d_i \right) + \frac{\text{maximum noise}}{2} \text{ mod } |\text{Domain}|; \)
    // Computing the probability of \( G \), i.e., the probability
    // for obtaining \( 'r' \) successes in \( 'N' \) trials, where \( N \)
    // is \( 'number of values' \) for the current attribute and
    // \( p = 0.5 \)
    \( P(G) \leftarrow \frac{N!}{G!(N-G)!} \cdot p^G (1-p)^{N-G}; \)
    // Contribution of current perturbed record towards \( p_i \)
    \( p_i \leftarrow p_i + (\text{probPTR} \cdot P(r)); \)
  end
  \( \text{probSUM} \leftarrow \text{probSUM} + \text{probPTR}; \)
end

foreach \( p_i \) in the confidential attribute Domain do
  // dividing \( p_i \) by probSUM, to ensure that all probabilities
  // add up to 1.
  \( p_i \leftarrow \frac{p_i}{\text{probSUM}}; \)
end

Display: \( d[ ], p[ ] \)

Algorithm 3: Analysing noise
6.4 The Experiments: PUMS Dataset Results

6.4.1 Sampling

We present here the results of using our security measure to evaluate how secure sampling is as a disclosure control method for the PUMS dataset. We organise the results into three different categories:

1. **Result variation in 30 samples:**
   This part presents results of running the experiment on 30 different randomly generated samples from the same original dataset. The results presented here are for sampling factor of 0.1. Results for other sampling factors, 0.05, 0.2 and 0.5, can be found in Appendix B. Figure 6.2 shows the density of the average initial entropy with different intruder’s Supplementary Knowledge (SK). For example, Figure 6.2b is for an intruder’s SK of 2 attributes. We notice that the average initial entropy values for different samples are distributed in the range \([4.530 - 4.545]\). This is a very small range and it shows that our results do not differ significantly from sample to sample. This conclusion also holds for the average entropy. Recall from Section 5.5 that we obtained it as an area divided by the domain. Strictly speaking, it is the entropy \(H(\epsilon)\) averaged over \(\epsilon\). We also refer to ”average entropy” as ”area” in the figures in this chapter. Figure 6.3c shows that, with an intruder’s SK of 3 attributes, the average entropy values are in the range \([1.465 - 1.470]\).
Figure 6.2: SDC: Sampling, Dataset: PUMS, Sampling Factor: 0.1. Distribution of average initial entropy for 30 samples when SK is 1, 2, 3, or 6 attributes.
2. The impact of supplementary knowledge on the results:

In this part of Sampling experiment, we generate 30 samples for every sample size and compute, over the 30 samples, the average initial entropy and the average entropy for each size of the intruder’s SK. Figure 6.4 shows the relationship between the average initial entropy and the average entropy on one hand and the intruder’s SK (the number of known attributes, on the other hand.

Figure 6.4a shows that, with a small sampling factor such as 0.05 and 0.1, the average initial entropy does not largely decrease as the size of the intruder’s SK increases. Number of known attributes in the intruder’s SK has more affect on the average initial entropy when we have large sample size such as 50%. Figure 6.4b shows that with sampling factor of 0.50, average entropy starts to decrease sharply when the intruder knows 2 attributes and does not differ much when the intruder’s SK consists of 5 or 6 attributes. We conclude that the intruder’s SK has bigger impact on security as we have larger size of
the released sample.

Figure 6.4: SDC: Sampling, Dataset: PUMS. (average $H_0$ and average entropy) vs intruder’s SK (number of known attributes)

3. The impact of sample size on the results:
This category of the experiment is actually the same as the previous one but the results are presented differently. For each number of attributes in the intruder’s SK, we study the impact of sample size on the average initial entropy and the average entropy. Figure 6.5 shows that sample size does not have big impact when the intruder’s SK consists of 1 attribute. With the growth of SK, the impact of sample size on the average entropy becomes bigger. We notice that with small sampling factor (0.05), regardless of how big is the intruder’s SK, the average entropy values are close. However, the difference in average entropy values is getting bigger among different SK sizes as the sample size increases. We notice that for this dataset (PUMS), knowing 5 or 6 attributes by the intruder yields almost same results of the average entropy. This figure shows that our security measure can be used by releasing organizations in order to decide what is the most suitable and secure sample size to release for users.
4. The most revealing attribute:

This part of the experiment sheds the light on the intruder’s SK. Our security measure is able to produce results that answer the question
of what is the most revealing attribute, or combinations of attributes, in quasi-identifiers. For a supplementary knowledge size of 1 we identify the most revealing attribute and when SK consists of 2 attributes, we identify the most revealing couple of attribute and we call it a “combination”. The results introduced here, Figure 6.6, are for sampling factor of “0.5” and are computed as average over 30 different samples from the same original dataset. Results for other sampling factors (0.05, 0.1and0.2) can be found in Appendix B.

Figure 6.6b shows the density of the average entropy when the intruder’s SK consists of 1 attribute. The figure shows the distribution of the area values for different attributes. For example, it shows that age is the most revealing attribute, with average area value of “1.272”, while sex is the least revealing attribute, with average area value of “1.292”. Actually, this finding is not surprising as the actual domain of age, according to Table 6.1, has “69” possible values while sex has only “2” values. Figure 6.6d shows the density of the average entropy when the intruder’s SK size is 2. The figure shows the distribution of the area values for different combinations of 2 attributes. The combination that consists of attributes \{age, occupation\} is the most revealing combination. The least revealing combination is \{Sex, education\}. average entropy values are distributed in the range [1.088 – 1.277]. This is a wider range than the one for SK of size 1 attribute which means that knowing certain combination of 2 attributes, such as \{age, occupation\}, reduces the value of average area in a noticeable way.

Figure 6.6f shows the density of the average entropy when the intruder’s SK size is 3. The combination that consists of attributes \{age, occupation, work travel time\} is the most revealing combination. The least revealing combination is \{Sex, education, industry\}. average entropy values are distributed in the range [0.846 – 1.204].
6.4 The Experiments: PUMS Dataset Results

Figure 6.6: SDC: Sampling, Dataset: PUMS, Sampling Factor: 0.5. Distribution of average initial entropy and average entropy for different 1, 2, and 3 attribute combinations

6.4.2 Query restriction

We present here the results of evaluating query restriction as a disclosure control method for the PUMS dataset. We organise the results into three different categories:

1. **Result variation in 30 sample files:**
   We randomly permute the records in the original dataset in order
to get different systems of linear equations. We do this 30 times in order to produce results with confidence intervals. This is done for each query set size. The results presented here are for a query set size of 2. Results for other query set sizes (4, 8, 16 and 32) can be found in Appendix B. Figure 6.8 shows the density of the average entropy with different intruder’s SK. For example, Figure 6.8c is for an intruder’s SK of 3 attributes. We notice that the average entropy’s values are effectively the same from one systems of linear equations to another. This conclusion also holds for the calculations of average initial entropy, Figure 6.7.

Figure 6.7: SDC: Query Restriction, Dataset: PUMS, Query Set Size: 2. Distribution of average initial entropy for 30 shuffled datasets when SK is 1, 2, 3, or 6 attributes.
2. The impact of supplementary knowledge and query set size on the results:

In this category of Query Restriction experiment, we generate, out of the same dataset, 30 different systems of linear equations for every query set size and compute, over the 30 systems of linear equations, the average initial entropy and the average entropy for each size of the intruder’s SK. Figure 6.9 shows the relationship between the average initial entropy and the average entropy on one hand and the intruder’s SK, that is the number of known attributes, on the other hand.

Figure 6.9a shows that the average initial entropy as a function of the intruder’s SK. For a query set size of 2, there is almost no uncertainty about the confidential value when the intruder knows at least 5 or more attributes. Our measure did assess more precisely the amount of risk that is involved from answering same set of range queries. Figure 6.9b shows that, for a query set size of 2, the intruder needs only 4 attributes to infer the confidential value. Both figures, Figure 6.9a and Figure 6.9b, agree on the fact that restricting the
query set size to be 32 is too restrictive as it does not improve the security. A query set size of 16 gives the same result as query set size of 32. We conclude that our security measure can be used as an assessment tool by DBMS to judge what is the most suitable query set size that is not too restrictive and balance between the data quality and data confidentiality. Figure 6.10 supports this conclusion.
6.4 The Experiments: PUMS Dataset Results

Figure 6.9: SDC: Query Restriction, Dataset: PUMS. ($H_0$, Area) vs Number of Known Attributes
3. The most revealing attribute:
The results introduced in Figure 6.11, are for query set size of “2” and are computed as average over 30 different systems of linear equations from the same original dataset. Results for other query set sizes, (4, 8, 16, 32), can be found in Appendix B.

Figure 6.11a shows the density of the average initial entropy when the intruder’s SK consists of 1 attribute. The figure shows the distribution of the initial entropy for different attributes. It shows that age is the most revealing attribute, with average initial entropy value of “2.995”. Our security measure, Figure 6.11c, shows that age is the second most revealing attribute, with average area value of “0.532”, while occupation is the most revealing attribute, with average area value of “0.520”. Both initial entropy and our measure agree that Sex is the least revealing attribute, with average initial entropy value of “3.302” and average area value of “0.588”.

Figure 6.11d shows the density of the average entropy when the intruder’s SK size is 2. The figure shows the distribution of the area values for different combinations of 2 attributes. The combination that consists of attributes \{age, occupation\} is the most revealing combination, with average area value of “0.184”. The least revealing combination is \{sex, industry\}, with average area value of “0.538”. Average entropy values are distributed in the range [0.184 – 0.538]. This is considered a wide range and means that the identity of the attributes in the intruder’s SK does affect how much the intruder knows about a confidential value.
6.4 The Experiments: PUMS Dataset Results

![Initial Entropy](a) $H_0$, 1 attribute
![Initial Entropy](b) $H_0$, 2 attribute combinations
![Area](c) Area, 1 attribute
![Area](d) Area, 2 attribute combinations

Figure 6.11: SDC: Query Restriction, Dataset: PUMS, Query Set Size: 2. Distribution of the average initial entropy and the average entropy for different 1, 2 attribute combinations

6.4.3 Noise addition

We present here the results of using our security measure to evaluate how secure is Noise Addition as a disclosure control method for the PUMS dataset. The results are also organised into three different categories:

1. **Result variation in 30 perturbed files:**
   This part presents results of running the experiment on 30 different perturbed files from the same original dataset. The results for this category are for noise amount of “100%”. Results for other noise amounts, (25%, 50%, 75%), can be found in Appendix B. Figure 6.13 shows the density of the average entropy with different intruder’s SK. For example, Figure 6.13c is for an intruder’s SK of 3 attributes. We notice that average entropy’s values, from one perturbed file to another, are distributed in the range [1.250 – 1.278]. This is a small range and it shows that generating different perturbed files with the same amount
of noise does not yield big difference in the average entropy values. The difference is reasonable as the added noise is generated randomly. This conclusion holds for the values of the average initial entropy. Figure 6.12c shows that, with an intruder’s SK of 3 attributes, the average initial entropy values are in the range $[4.122 \text{ – 4.165}]$.

![Figure 6.12](image)

Figure 6.12: SDC: Noise Addition, Dataset: PUMS, Noise Amount: 100%. Distribution of average initial entropy for 30 perturbed files when SK is 1, 2, 3, or 6 attributes.
2. The impact of the intruder’s SK and the added noise amount on the results:

In this category of Noise Addition experiment, we generate 30 perturbed files for every noise amount and compute, over the 30 perturbed files, the average initial entropy and the average entropy for each size of the intruder’s SK. Figure 6.14 shows the relationship between the average initial entropy and the average entropy on one hand and the intruder’s SK, that is the number of known attributes, on the other hand.

Figure 6.14a shows that, with a large amount of noise such as “100%”, the average initial entropy does not largely decrease as the size of the intruder’s SK increases. Number of known attributes in the intruder’s SK has more affect on the average initial entropy when we have small amount of noise such as 25%. Figure 6.14b shows that with amount of noise of 25%, average entropy starts to drop in a large value when the
intruder knows 2 or 3 attributes and it is the worst when the intruder’s SK consists of 5 or 6 attributes. We conclude that the intruder’s SK has bigger impact on security as we have smaller amount of added noise in the perturbed files.

The results in Figure 6.15 are presented differently from Figure 6.14. For each number of attributes in the intruder’s SK, we study the impact of noise amount on the average initial entropy and the average entropy. Figure 6.15 shows that noise amount does not have huge impact when the intruder’s SK consists of 1 attribute. With the growth of SK, the impact of noise amount on the average entropy becomes bigger. We notice that with large amount of noise “100%”, regardless of how big is the intruder’s SK, the average entropy values are relatively close. However, the difference in average entropy values is getting bigger among different SK sizes as the amount of added noise decreases. This figure shows that our security measure can be used by releasing organizations in order to adjust the added amount of noise and to find the right balance between data quality and data confidentiality.
6.4 The Experiments: PUMS Dataset Results

Figure 6.14: SDC: Noise Addition, Dataset: PUMS. (average $H_0$ and average entropy) vs intruder’s SK (number of known attributes)
6.4 The Experiments: PUMS Dataset Results

Figure 6.15: SDC: Noise Addition, Dataset: PUMS. (average $H_0$ and average entropy) vs Noise Amount

(a) $H_0$ vs Noise Amount

(b) Area vs Noise Amount

3. The most revealing attribute:

The results introduced in Figure 6.16, are for noise amount of “100%” and are computed as average over 30 different perturbed datasets
from the same original dataset. Results for other noise amounts, (25%, 50%, 75%), can be found in Appendix B.

Figure 6.16c shows the density of the average entropy when the intruder’s SK consists of 1 attribute. The figure shows the distribution of the area values for different attributes. For example, it shows that \textbf{occupation} is the most revealing attribute, with average area value of “1.282”, while \textbf{age} was the most revealing attribute in Sampling experiment. It is worth to notice that \textbf{age} is still considered the second revealing attribute and it has average area value of “1.283”. \textbf{Sex} is the least revealing attribute, with average area value of “1.291”. However, average area values are distributed in the range [1.282 – 1.291] and we notice that it is very narrow range.

Figure 6.16d shows the density of the average entropy when the intruder’s SK size is 2. The figure shows the distribution of the area values for different combinations of 2 attributes. The combination that consists of attributes \{\textbf{industry, occupation}\} is the most revealing combination, with average area value of “1.267”. The least revealing combination is \{\textbf{sex, industry}\}, with average area value of “1.289”. Average entropy values are distributed in the range [1.267 – 1.289]. This is still considered a narrow range and recall from the results in the previous category that with a large amount of noise such as “100%”, the average initial entropy does not largely decrease as the size of the intruder’s SK increases. Number of known attributes in the intruder’s SK has more affect on the average initial entropy when we have small amount of noise such as 25%.
6.5 The Experiments: Other Dataset Results

6.5.1 Sampling: WBC dataset

The impact of the intruder’s SK and the sample size on the results: For the WBC dataset, we generate 30 samples for every sample size and compute, over the 30 samples, the average initial entropy and the average entropy for each size of the intruder’s SK. Figure 6.17 shows the relationship between the average initial entropy and the average entropy on one hand and the intruder’s SK, that is the number of known attributes, on the other hand.

Figure 6.17 shows that, with a small sampling factor such as 0.05 and 0.1, the average initial entropy does not largely decrease as the size of the intruder’s SK increases. The confidential attribute in WBC dataset has 10
values in its actual domain and thus $H_0$ according to Shannon’s entropy is $\log_2(10) = 3.322$. We notice from Figure 6.17a that for sampling factors of 0.05, 0.1, and 0.2 and with 1 attribute in the intruder’s SK, the average initial entropy is almost 3.322. It is stayed almost the same for sampling factor of 0.05 regardless of the increase in the intruder’s SK size while it drops a little in value for the sampling factor of 0.1. Figure 6.17b shows that with sampling factor of 0.50, the average entropy starts to drop in value when the intruder knows 1 attributes or more and it is the lowest when the intruder’s SK consists of 7, 8, or 9 attributes. We conclude that the intruder’s SK has bigger impact on security as we have larger size of the released sample.

The results in Figure 6.18 reveals same information as in presented differently from Figure 6.17 but we choose to represent the average initial entropy and the average entropy as a function of the sample size. Other results and diagrams can be found in Appendix B.2.
Figure 6.17: SDC: Sampling, Dataset: WBC. (average $H_0$ and average entropy) vs intruder’s SK (number of known attributes)
Figure 6.18: SDC: Sampling, Dataset: WBC. (average $H_0$ and average entropy) vs sample size
6.5.2 Sampling: WINE dataset

The impact of the intruder’s SK and the sample size on the results: For the WINE dataset, we generate 30 samples for every sample size and compute, over the 30 samples, the average initial entropy and the average entropy for each size of the intruder’s SK. Figure 6.19 shows the relationship between the average initial entropy and the average entropy on one hand and the intruder’s SK, that is the number of known attributes, on the other hand.

Figure 6.19 and Figure 6.20 show an interesting result and that is for all sampling factors when the size of the intruder’s SK reach to 2 attributes, the average initial entropy and the average entropy values stay stable. That means the intruder’s knowledge about the confidential value does not increase as the size of SK increase. We believe this is due to the different unique combinations in this dataset. Other results and diagrams can be found in Appendix B.3.
Figure 6.19: SDC: Sampling, Dataset: WINE. (average $H_0$ and average entropy) vs intruder's SK (number of known attributes)
Figure 6.20: SDC: Sampling, Dataset: WINE. (average $H_0$ and average entropy) vs sample size
6.5.3 Query restriction: WBC dataset

The impact of the intruder’s SK and the sample size on the results:
For the WBC dataset, we generate, out of the same dataset, 30 different systems of linear equations for every query set size and compute, over the 30 systems of linear equations, the average initial entropy and the average entropy for each size of the intruder’s SK. Figure 6.21 shows the relationship between the average initial entropy and the average entropy on one hand and the intruder’s SK, that is the number of known attributes, on the other hand.

Figure 6.21a represents the average initial entropy as a function of the intruder’s SK. It shows that the intruder’s knowledge about the confidential value is identical for query set sizes of 8, 16, and 32. Figure 6.21b represents the average initial entropy as a function of the intruder’s SK. It shows that how strong our novel measure in precisely capturing the intruder’s knowledge. It show that the intruder’s knowledge about the confidential value is identical only for query set sizes of 16, and 32 while it is slightly more for the query set size of 8. Other results and diagrams can be found in Appendix B.5.
Figure 6.21: SDC: Query Restriction, Dataset: WBC. (average $H_0$ and average entropy) vs intruder’s SK (number of known attributes)
Figure 6.22: SDC: Query Restriction, Dataset: WBC. (average $H_0$ and average entropy) vs Query Set Size
6.5.4 Query restriction: WINE dataset

The impact of the intruder’s SK and the sample size on the results:
For the WINE dataset, we generate, out of the same dataset, 30 different systems of linear equations for every query set size and compute, over the 30 systems of linear equations, the average initial entropy and the average entropy for each size of the intruder’s SK. Figure 6.23 shows the relationship between the average initial entropy and the average entropy on one hand and the intruder’s SK, that is the number of known attributes, on the other hand.

Figure 6.23a represents the average initial entropy as a function of the intruder’s SK. It shows that the intruder’s knowledge about the confidential value is identical for query set size of 8, 16, and 32. Figure 6.21b represents the average initial entropy as a function of the intruder’s SK. It shows that the intruder’s knowledge about the confidential value is identical for query set size of 8, 16, and 32. We conclude that if a releasing organization permits a query set size of 16, or 32 it will consider overly restrictive as query set size of 8 will lead to same results but give more usefulness of the dataset to the users. Other results and diagrams can be found in Appendix B.6.
Figure 6.23: SDC: Query Restriction, Dataset: WINE. (average $H_0$ and average entropy) vs intruder’s SK (number of known attributes)
Figure 6.24: SDC: Query Restriction, Dataset: WINE. (average $H_0$ and average entropy) vs Query Set Size
6.5.5 Noise addition: WBC dataset

The impact of the intruder’s SK and the sample size on the results:
For the WBC dataset, we generate 30 perturbed files for every noise amount and compute, over the 30 perturbed files, the average initial entropy and the average entropy for each size of the intruder’s SK. Figure 6.25 shows the relationship between the average initial entropy and the average entropy on one hand and the intruder’s SK, that is the number of known attributes, on the other hand.

Figure 6.25 shows that, with a large amount of noise such as “100%”, the average initial entropy and the average entropy slightly decrease as the size of the intruder’s SK increases. Number of known attributes in the intruder’s SK has more affect on the average initial entropy and the average entropy when the amount of noise such is smaller as 25%. Figure 6.14b shows that with amount of noise of 25%, average entropy starts to drop in value when the intruder knows 1 or more attributes and it is the worst when the intruder’s SK consists of 9 attributes. We conclude that the intruder’s SK has larger impact on security when a smaller amount of noise is added to the perturbed files. Other results and diagrams can be found in Appendix B.8.
6.5 The Experiments: Other Dataset Results

Figure 6.25: SDC: Noise Addition, Dataset: WBC. (average $H_0$ and average entropy) vs intruder's SK (number of known attributes)
Figure 6.26: SDC: Noise Addition, Dataset: WBC. (average $H_0$ and average entropy) vs Noise Amount
6.5.6 Noise addition: WINE dataset

The impact of the intruder’s SK and the sample size on the results:
For the WINE dataset, we generate 30 perturbed files for every noise amount and compute, over the 30 perturbed files, the average initial entropy and the average entropy for each size of the intruder’s SK. Figure 6.27 shows the relationship between the average initial entropy and the average entropy on one hand and the intruder’s SK, that is the number of known attributes, on the other hand.

Figure 6.27 shows that, unlike the other datasets (PUMS and WBC), with a large amount of noise such as “100%”, the average initial entropy and the average entropy decrease as the size of the intruder’s SK increases. Number of known attributes in the intruder’s SK has the biggest affect on the average initial entropy and the average entropy when the amount of noise is 25%. Figure 6.27b shows that with amount of noise of 25%, the average entropy drops almost to minimum when we have when the intruder knows 4 or more attributes. Figure 6.28b shows that with the intruder’s SK consists of 7 attributes, any increase in the SK size after that does not lead to an increase in the intruder’s knowledge about the confidential value. Other results and diagrams can be found in Appendix B.9.
Figure 6.27: SDC: Noise Addition, Dataset: WINE. (average $H_0$ and average entropy) vs intruder’s SK (number of known attributes)
Figure 6.28: SDC: Noise Addition, Dataset: WINE. (average $H_0$ and average entropy) vs Noise Amount
Chapter 7

Conclusion

Mobile agents are programs that travel autonomously through a computer network in order to perform some computation or gather information on behalf of a human user or an application [58]. The mobile agent system is a very promising paradigm that has already established its presence in many applications including e-commerce and distributed information search and retrieval. At the same time, this technology has introduced some very serious security problems and emphasized some existing security issues. It is more difficult to ensure security in the mobile agent paradigm than in some other technologies where hardware solutions are practical.

In this thesis we surveyed the main issues in the security of mobile agents. We considered both the mobile agent and the agent platform points of view, and reconfirmed that it is much more difficult to ensure the security of mobile agents than the security of agent platforms. We discussed the security threats and requirements that need to be met in order to alleviate those threats.

We presented the most important techniques for providing security in mobile agent systems. Some of those techniques, for example Sandboxing, have been used for a long time and are well understood. On the other hand, some other techniques, such as Computing with Encrypted Function are still at the theoretical level and are not yet widely used in practice. None of the existing techniques provides an optimal solution for all scenarios. For ex-
ample, Sandboxing provides a high level of security but is overly restrictive as only a very few applications can operate in such a constrained environment. However, a combination of various techniques may yield powerful solutions. For example, in Java 2 Sandboxing has been used in combination with fine-grained access control and Code Signing. In any case, more research is needed in the future to warrant sufficient trust in mobile agent technology by a wide range of users.

There is a need for new security solutions that can accommodate different aspects of open system architecture including the social behavior of its entities and can work side by side with traditional security mechanisms [85]. Social control is considered a soft security solution [83]. We proposed coupling technique as a security solution that is based on the trust as a social control to work together with existing traditional security mechanisms. Coupling technique aims to increase the probability that the agent successfully accomplishes its task by partitioning the itinerary into the pairs, or “couples” of platforms. The coupling radically increases the overall probability that the agent will not be harmed, especially if the itinerary contains servers with a very low level of trust.

It is difficult to find a general method that would be applicable for different mobile agent scenarios and applications. Our Scout and Routed agent approach dramatically increases the security of an agent, at the expense of somewhat slower execution of the agent and the increase in its size. In the very hostile environment, due to the increase in agent’s size the method could become unfeasible. However, in real life environments malicious platforms are rare. Additionally, the Routed agent’s itinerary is already relatively trusted, as obviously malicious host are previously filtered out. Thus Routed agent is expected to perform efficiently and provides an attractive solution for securing mobile agents for which high level of security is essential.

Routed agent has several advantages over a traditional single agent: Its itinerary is filtered and it contains only platforms with sufficient trust level; at the same time, mobility and autonomy are preserved through the Scout agent. Knowing itinerary in advance enables the owner to use various methods to increase security, e.g., public key encryption to ensure that only
platforms in the itinerary can read and execute the agent, or Cubaleska and Schneider method [28] for detection of DoS attack. Routed agent travels from “couple” to “couple” rather then from platform to platform which remarkably increases its security. The total time taken by the Routed agent is expected to be shorter than the time taken by an agent going sequentially through the same itinerary, as the platforms in a couple are processing the agent in parallel. Moreover, a limit can be imposed on the time each platform waits to receive a second copy of the agent, which limits the total time taken by a Routed agent.

The Scout/Routed agent technique is very suitable for e-commerce applications such as shopping mobile agent that collects prices and offers. Our Routed agent technique is also applicable independently of the Scout agent, whenever the itinerary and the trust values of the platforms in the itinerary are known.

We also proposed a Petrol Station as an entity that would provide a service to other entities, in the form of certifying mobile agents and equipping them with safe itinerary based on trust score and applying the Routed agent.

The above discussion focuses on the security of the mobile agent. In the second part of this thesis, we shed some light on the security of mobile agent platforms and in particular the Statistical Disclosure Control’s dilemma. On one hand, one must keep the risk of individual value disclosure as low as possible. On the other hand, the utility (usefulness) of the database must remain high. A good SDC measure aims at finding a right balance between the two. In order to achieve this balance, it is crucial to precisely measure both utility and disclosure risk. We proposed a disclosure risk measure that covers the shortcomings of the proposed discloser risk measures in the literature.

Our disclosure risk measure is based on Shannon’s entropy but covers both exact and approximate compromise. It gives careful consideration to the attribute values in addition to their probabilities and calculates the disclosure risk for various levels of approximate compromise by using the proposed dynamic programming algorithm. The measure’s strength point
comes from the fact that it is independent of the applied SDC technique.

Our proposed measure is easy to implement and we showed, by running extensive experiments, how this measure can be used to assess the security mechanisms for protecting privacy in statistical databases and data mining. The experiment produced strong results. For SDC such as Sampling, the results showed that our security measure can be implemented by releasing organizations in order to decide what is the most suitable and secure sample size to release for users. Additionally, for Query Restriction, the security measure can be used as an assessment tool by various DBMS to support decisions such as what is the most suitable query set size that is not overly restrictive and to find the right balance between the quality and confidentiality of the data.

Moreover, in Noise Addition context, the measure can be used, by releasing organizations, in order to adjust the amount of noise that needs to be added to the original dataset before releasing it to users. We conducted the experiments on three different datasets protected by three different SDC techniques, namely Sampling, Query Restriction, and Noise Addition. The novel measure is a very helpful tool in guiding the releasing organizations to the most revealing or sensitive identifying attribute. This will the organizations in implementing some techniques that pay more attention to this sensitive attribute. For example, in Sampling, there might be some action needs to be taken in order to get rid of some unique combinations of sensitive attributes. We assumed that there is only one confidential attribute; it is straightforward to generalize our experiments to cover more than one confidential attribute and to apply the measure on more SDC techniques and we leave that for future work.
Appendix A

A Numerical Example

We present further illustration to our proposed dynamic programming algorithm. We use Algorithm 1 to calculate the disclosure risk in the numerical example in Section 5.4.1:

Stage 1: $\epsilon_1 = 1$, we need to compute $H(\epsilon)$.

State 1: $i = 1$ or $X_1 = 1$, we need to calculate $H(i, \epsilon) = H(1, 1)$

\[
H(i, \epsilon) = \min[(H(i-1, \epsilon) + a_i), \ (H(i-2, \epsilon) + a_{i-1}), ..., \ (H(j-1, \epsilon) + a_j)]
\]

\[
H(1, 1) = \min[(H(0, 1) + a_1)]
\]

\[
H(1, 1) = \min[(0 + p_1 \cdot \log(\frac{1}{p_1}))]
\]

\[
H(1, 1) = \min[(0 + 0.411)]
\]

\[
H(1, 1) = 0.411
\]

State 2: $i = 2$ or $X_2 = 3$, we need to calculate $H(i, \epsilon) = H(2, 1)$

\[
H(2, 1) = \min[(H(1, 1) + a_2)]
\]

\[
H(2, 1) = \min[(H(1, 1) + p_2 \cdot \log(\frac{1}{p_2}))]
\]

\[
H(2, 1) = \min[(0.411 + 0.332)]
\]
\[ H(2,1) = 0.743 \]

State 3: \( i = 3 \) or \( X_3 = 8 \), we need to calculate \( H(i, \epsilon) = H(3,1) \)

\[
\begin{align*}
H(3,1) &= \min[(H(2,1) + a_3)] \\
H(3,1) &= \min[(H(2,1) + p_3 \cdot \log(\frac{1}{p_3}))] \\
H(3,1) &= \min[(0.743 + 0.360)] \\
H(3,1) &= 1.103
\end{align*}
\]

State 4: \( i = 4 \) or \( X_4 = 9 \), we need to calculate \( H(i, \epsilon) = H(4,1) \)

\[
\begin{align*}
H(4,1) &= \min[(H(3,1) + a_4), (H(2,1) + a_3)] \\
H(4,1) &= \min[(H(3,1) + 0.216), (H(2,1) + ((p_3 + p_4) \cdot \log(\frac{1}{(p_3 + p_4)})))] \\
H(4,1) &= \min[(1.103 + 0.216), (0.743 + 0.311)] \\
H(4,1) &= \min[(1.319), (1.054)] \\
H(4,1) &= 1.054
\end{align*}
\]

Stage 2: \( \epsilon_2 = 2 \), we need to compute \( H(\epsilon) \). State 1: \( i = 1 \) or \( X_1 = 1 \), we need to calculate \( H(i, \epsilon) = H(1,2) \)

\[
\begin{align*}
H(1,2) &= p_1 \cdot \log(\frac{1}{p_1}) \\
H(1,2) &= 0.411
\end{align*}
\]

State 2: \( i = 2 \) or \( X_2 = 3 \), we need to calculate \( H(i, \epsilon) = H(2,2) \)

\[
\begin{align*}
H(2,2) &= \min[(H(1,2) + a_2), (H(0,2) + a_1)] \\
H(2,2) &= \min[(H(1,2) + 0.332), (0 + ((p_1 + p_2) \cdot \log(\frac{1}{(p_1 + p_2)})))] \\
H(2,2) &= \min[(0.411 + 0.332), (0 + 0.5)] \\
H(2,2) &= \min[(0.743), (0.5)] \\
H(2,2) &= 0.5
\end{align*}
\]

State 3: \( i = 3 \) or \( X_3 = 8 \), we need to calculate \( H(i, \epsilon) = H(3,2) \)

\[
\begin{align*}
H(3,2) &= \min[(H(2,2) + a_3)]
\end{align*}
\]
\[
H(3, 2) = \min[(H(2, 2) + p_3 \cdot \log(\frac{1}{p_3}))]
H(3, 2) = \min[(0.5 + 0.360)]
H(3, 2) = \ 0.860
\]

State 4: \( i = 4 \) or \( X_4 = 9 \), we need to calculate \( H(i, \epsilon) = H(4, 2) \)
\[
H(4, 2) = \min[(H(3, 2) + a_4), (H(2, 2) + a_3)]
H(4, 2) = \min[(H(3, 2) + 0.216), (H(2, 2) + ((p_3 + p_4) \cdot \log(\frac{1}{p_3 + p_4})))]
H(4, 2) = \min[(0.860 + 0.216), (0.5 + 0.311)]
H(4, 2) = \min[(1.076), (0.811)]
H(4, 2) = \ 0.811
\]

Stage 3: \( \epsilon_3 = 5 \), we need to compute \( H(\epsilon) \). State 1: \( i = 1 \) or \( X_1 = 1 \), we need to calculate \( H(i, \epsilon) = H(1, 5) \)
\[
H(1, 5) = p_1 \cdot \log(\frac{1}{p_1})
H(1, 5) = \ 0.411
\]

State 2: \( i = 2 \) or \( X_2 = 3 \), we need to calculate \( H(i, \epsilon) = H(2, 5) \)
\[
H(2, 5) = \min[(H(1, 5) + a_2), (H(0, 5) + a_1)]
H(2, 5) = \min[(0.743), (0.5)]
H(2, 5) = \ 0.5
\]

State 3: \( i = 3 \) or \( X_3 = 8 \), we need to calculate \( H(i, \epsilon) = H(3, 5) \)
\[
H(3, 5) = \min[(H(2, 5) + a_3), (H(1, 5) + a_2)]
H(3, 5) = \min[(H(2, 5) + 0.360), (H(1, 5) + ((p_2 + p_3) \cdot \log(\frac{1}{p_2 + p_3})))]
H(3, 5) = \min[(0.5 + 0.360), (0.411 + 0.258)]
H(3, 5) = \min[(0.860), (0.668)]
H(3, 5) = \ 0.668
\]
State 4: $i = 4$ or $X_4 = 9$, we need to calculate $H(i, \epsilon) = H(4, 5)$

$$H(4, 5) = \min[(H(3, 5) + a_4), (H(2, 5) + a_3)]$$

$$H(4, 5) = \min[(H(3, 5) + 0.216), (H(2, 5) + ((p_3 + p_4) \cdot \log(\frac{1}{p_3 + p_4})))$$

$$H(4, 5) = \min[(0.668 + 0.216), (0.5 + 0.311)]$$

$$H(4, 5) = \min[(0.884), (0.811)]$$

$H(4, 5) = 0.811$

Stage 4: $\epsilon_4 = 6$, we need to compute $H(\epsilon)$. Stage 1: $i = 1$ or $X_1 = 1$, we need to calculate $H(i, \epsilon) = H(1, 6)$

$$H(1, 6) = p_1 \cdot \log(\frac{1}{p_1})$$

$H(1, 6) = 0.411$

State 2: $i = 2$ or $X_2 = 3$, we need to calculate $H(i, \epsilon) = H(2, 6)$

$$H(2, 6) = \min[(H(1, 6) + a_2), (H(0, 6) + a_1)]$$

$$H(2, 6) = \min[(0.743), (0.5)]$$

$H(2, 6) = 0.5$

State 3: $i = 3$ or $X_3 = 8$, we need to calculate $H(i, \epsilon) = H(3, 6)$

$$H(3, 6) = \min[(H(2, 6) + a_3), (H(1, 6) + a_2)]$$

$$H(3, 6) = \min[(H(2, 6) + 0.360), (H(1, 6) + ((p_2 + p_3) \cdot \log(\frac{1}{p_2 + p_3})))$$

$$H(3, 6) = \min[(0.5 + 0.360), (0.411 + 0.258)]$$

$$H(3, 6) = \min[(0.860), (0.668)]$$

$H(3, 6) = 0.668$

State 4: $i = 4$ or $X_4 = 9$, we need to calculate $H(i, \epsilon) = H(4, 6)$

$$H(4, 6) = \min[(H(3, 6) + a_4), (H(2, 6) + a_3), (H(1, 6) + a_2)]$$

$$H(4, 6) = \min[(0.884), (0.811), (H(1, 6) + ((p_2 + p_3 + p_4) \cdot \log(\frac{1}{p_2 + p_3 + p_4})))$$

$$H(4, 6) = \min[(0.884), (0.811), (0.411 + 0.199)]$$

$H(4, 6) = \min[(0.884), (0.811), (0.610)]$
\( H(4, 6) = 0.610 \)

Stage 5: \( \epsilon_5 = 7 \), we need to compute \( H(\epsilon) \). State 1: \( i = 1 \) or \( X_1 = 1 \), we need to calculate \( H(i, \epsilon) = H(1, 7) \)

\[
H(1, 7) = p_1 \cdot \log \left( \frac{1}{p_1} \right)
\]

\( H(1, 7) = 0.411 \)

State 2: \( i = 2 \) or \( X_2 = 3 \), we need to calculate \( H(i, \epsilon) = H(2, 6) \)

\[
H(2, 7) = \min[(H(1, 7) + a_2), (H(0, 7) + a_1)]
\]

\( H(2, 7) = \min[(0.743), (0.5)] \)

\( H(2, 7) = 0.5 \)

State 3: \( i = 3 \) or \( X_3 = 8 \), we need to calculate \( H(i, \epsilon) = H(3, 7) \)

\[
H(3, 7) = \min[(H(2, 7) + a_3), (H(1, 7) + a_2), (H(0, 7) + a_1)]
\]

\[
H(3, 7) = \min[(0.860, 0.668), (H(0, 7) + ((p_1 + p_2 + p_3) \cdot \log(\frac{1}{(p_1 + p_2 + p_3)})))]
\]

\( H(3, 7) = \min[(0.860, 0.668), (0 + 0.070)] \)

\( H(3, 7) = \min[(0.860, 0.668), (0.070)] \)

\( H(3, 7) = 0.070 \)

State 4: \( i = 4 \) or \( X_4 = 9 \), we need to calculate \( H(i, \epsilon) = H(4, 7) \)

\[
H(4, 7) = \min[(H(3, 7) + a_4), (H(2, 7) + a_3), (H(1, 7) + a_2)]
\]

\[
H(4, 7) = \min[(0.070 + 0.216), (0.5 + 0.311), (0.411 + 0.199)]
\]

\( H(4, 7) = \min[(0.286), (0.811), (0.610)] \)

\( H(4, 7) = 0.286 \)

Stage 6: \( \epsilon_6 = 8 \), we need to compute \( H(\epsilon) \). State 4: \( i = 4 \) or \( X_4 = 9 \), we need to calculate \( H(i, \epsilon) = H(4, 8) \)

\[
H(4, 8) = \min[(H(3, 8) + a_4), (H(2, 8) + a_3), (H(1, 8) + a_2), (H(0, 8) + a_1)]
\]

\[
H(4, 8) = (H(0, 8) + a_1)
\]

\[
H(4, 8) = (0 + ((p_1 + p_2 + p_3 + p_4) \cdot \log(\frac{1}{(p_1 + p_2 + p_3 + p_4)})))
\]
\[ H(4, 8) = 0 \]

<table>
<thead>
<tr>
<th>( \epsilon )</th>
<th>( X_1 = 1 )</th>
<th>( X_2 = 3 )</th>
<th>( X_3 = 8 )</th>
<th>( X_4 = 9 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \epsilon_1 = 1 )</td>
<td>0.411</td>
<td>0.743</td>
<td>1.103</td>
<td>1.054</td>
</tr>
<tr>
<td>( \epsilon_2 = 2 )</td>
<td>0.411</td>
<td>0.5</td>
<td>0.860</td>
<td>0.811</td>
</tr>
<tr>
<td>( \epsilon_3 = 5 )</td>
<td>0.411</td>
<td>0.5</td>
<td>0.668</td>
<td>0.811</td>
</tr>
<tr>
<td>( \epsilon_4 = 6 )</td>
<td>0.411</td>
<td>0.5</td>
<td>0.668</td>
<td>0.610</td>
</tr>
<tr>
<td>( \epsilon_5 = 7 )</td>
<td>0.411</td>
<td>0.5</td>
<td>0.070</td>
<td>0.286</td>
</tr>
<tr>
<td>( \epsilon_6 = 8 )</td>
<td>0.411</td>
<td>0.5</td>
<td>0.070</td>
<td>0.0</td>
</tr>
</tbody>
</table>
Appendix B

Extra Experiment Results

B.1 Sampling: PUMS Dataset

Figure B.1: SDC: Sampling, Dataset: PUMS, Sampling Factor: 0.05. Distribution of average initial entropy for 30 samples when SK is 1, 2, 3, 4, 5, or 6 attributes.
Figure B.2: SDC: Sampling, Dataset: PUMS, Sampling Factor: 0.05. Distribution of average entropy for 30 samples when SK is 1, 2, 3, 4, 5, or 6 attributes.
B.1 Sampling: PUMS Dataset

Figure B.3: SDC: Sampling, Dataset: PUMS, Sampling Factor: 0.2. Distribution of average initial entropy for 30 samples when SK is 1, 2, 3, 4, 5, or 6 attributes.
Figure B.4: SDC: Sampling, Dataset: PUMS, Sampling Factor: 0.2. Distribution of average entropy for 30 samples when SK is 1, 2, 3, 4, 5, or 6 attributes.
Figure B.5: SDC: Sampling, Dataset: PUMS, Sampling Factor: 0.5. Distribution of average initial entropy for 30 samples when SK is 1, 2, 3, 4, 5, or 6 attributes.
B.1 Sampling: PUMS Dataset

Figure B.6: SDC: Sampling, Dataset: PUMS, Sampling Factor: 0.5. Distribution of average entropy for 30 samples when SK is 1, 2, 3, 4, 5, or 6 attributes.
Figure B.7: SDC: Sampling, Dataset: PUMS, Sampling Factor: 0.05. Distribution of average initial entropy and average entropy for different 1, and 2 attribute combinations.
Figure B.8: SDC: Sampling, Dataset: PUMS, Sampling Factor: 0.1. Distribution of average initial entropy and average entropy for different 1, and 2 attribute combinations.
Figure B.9: SDC: Sampling, Dataset: PUMS, Sampling Factor: 0.2. Distribution of average initial entropy and average entropy for different 1, and 2 attribute combinations
B.2 Sampling: WBC Dataset

Figure B.10: SDC: Sampling, Dataset: WBC, Sampling Factor: 0.1. Distribution of average initial entropy for 30 samples when SK is 1, 2, 3, 5, 7, or 9 attributes.
Figure B.11: SDC: Sampling, Dataset: WBC, Sampling Factor: 0.1. Distribution of average entropy for 30 samples when SK is 1, 2, 3, 5, 7, or 9 attributes.
Figure B.12: SDC: Sampling, Dataset: WBC, Sampling Factor: 0.1. Distribution of average initial entropy and average entropy for different 1, and 2 attribute combinations.
B.3 Sampling: WINE Dataset

Figure B.13: SDC: Sampling, Dataset: WINE, Sampling Factor: 0.1. Distribution of average initial entropy for 30 samples when SK is 1, 2, 3, 5, 7, 9, 11, or 13 attributes.
(a) 1 attribute is known
(b) 2 attributes are known
(c) 3 attributes are known
(d) 5 attributes are known
(e) 7 attributes are known
(f) 9 attributes are known
(g) 7 attributes are known
(h) 9 attributes are known

Figure B.14: SDC: Sampling, Dataset: WINE, Sampling Factor: 0.1. Distribution of average entropy for 30 samples when SK is 1, 2, 3, 5, 7, 9, 11, or 13 attributes.
Figure B.15: SDC: Sampling, Dataset: WINE, Sampling Factor: 0.1. Distribution of average initial entropy and average entropy for different 1, and 2 attribute combinations.
B.4 Query Restriction: PUMS Dataset

Figure B.16: SDC: Query Restriction, Dataset: PUMS, Query Set Size: 4. Distribution of average initial entropy for 30 shuffled datasets when SK is 1, 2, 3, 4, 5, or 6 attributes.
Figure B.17: SDC: Query Restriction, Dataset: PUMS, Query Set Size: 4. Distribution of average entropy for 30 shuffled datasets when SK is 1, 2, 3, 4, 5, or 6 attributes.
Figure B.18: SDC: Query Restriction, Dataset: PUMS, Query Set Size: 8. Distribution of average initial entropy for for 30 shuffled datasets when SK is 1, 2, 3, 4, 5, or 6 attributes.
Figure B.19: SDC: Query Restriction, Dataset: PUMS, Query Set Size: 8. Distribution of average entropy for 30 shuffled datasets when SK is 1, 2, 3, 4, 5, or 6 attributes.
Figure B.20: SDC: Query Restriction, Dataset: PUMS, Query Set Size: 16. Distribution of average initial entropy for 30 shuffled datasets when SK is 1, 2, 3, 4, 5, or 6 attributes.
Figure B.21: SDC: Query Restriction, Dataset: PUMS, Query Set Size: 16. Distribution of average entropy for 30 shuffled datasets when SK is 1, 2, 3, 4, 5, or 6 attributes.
Figure B.22: SDC: Query Restriction, Dataset: PUMS, Query Set Size: 4. Distribution of average initial entropy and average entropy for different 1, and 2 attribute combinations.
Figure B.23: SDC: Query Restriction, Dataset: PUMS, Query Set Size: 8. Distribution of average initial entropy and average entropy for different 1, and 2 attribute combinations.
Figure B.24: SDC: Query Restriction, Dataset: PUMS, Query Set Size: 16. Distribution of average initial entropy and average entropy for different 1, and 2 attribute combinations
B.5 Query Restriction: WBC Dataset

Figure B.25: SDC: Query Restriction, Dataset: WBC, Query Set Size: 2. Distribution of average initial entropy for 30 shuffled datasets when SK is 1, 2, 3, 5, 7, or 9 attributes.
Figure B.26: SDC: Query Restriction, Dataset: WBC, Query Set Size: 2. Distribution of average entropy for 30 shuffled datasets when SK is 1, 2, 3, 5, 7, or 9 attributes.
B.5 Query Restriction: WBC Dataset

Figure B.27: SDC: Query Restriction, Dataset: WBC, Query Set Size: 2. Distribution of the average initial entropy and the average entropy for different 1, 2 attribute combinations.
B.6 Query Restriction: WINE Dataset

Figure B.28: SDC: Query Restriction, Dataset: WINE, Query Set Size: 2. Distribution of average initial entropy for 30 shuffled datasets when SK is 1, 2, 3, 5, 7, 9, 11, or 13 attributes.
B.6 Query Restriction: WINE Dataset

Figure B.29: SDC: Query Restriction, Dataset: WINE, Query Set Size: 2. Distribution of average entropy for 30 shuffled datasets when SK is 1, 2, 3, 5, 7, 9, 11, or 13 attributes.
Figure B.30: SDC: Query Restriction, Dataset: WINE, Query Set Size: 2. Distribution of the average initial entropy and the average entropy for different 1, 2 attribute combinations.
B.7 Noise Addition: PUMS Dataset

Figure B.31: SDC: Noise Addition, Dataset: PUMS, Noise Amount: 75%. Distribution of average initial entropy for 30 perturbed files when SK is 1, 2, 3, 4, 5, or 6 attributes.
Figure B.32: SDC: Noise Addition, Dataset: PUMS, Noise Amount: 75%. Distribution of average entropy for 30 perturbed files when SK is 1, 2, 3, 4, 5, or 6 attributes.
Figure B.33: SDC: Noise Addition, Dataset: PUMS, Noise Amount: 50%. Distribution of average initial entropy for 30 perturbed files when SK is 1, 2, 3, 4, 5, or 6 attributes.
Figure B.34: SDC: Noise Addition, Dataset: PUMS, Noise Amount: 50%. Distribution of average entropy for 30 perturbed files when SK is 1, 2, 3, 4, 5, or 6 attributes.
B.7 Noise Addition: PUMS Dataset

Figure B.35: SDC: Noise Addition, Dataset: PUMS, Noise Amount: 25%. Distribution of average initial entropy for 30 perturbed files when SK is 1, 2, 3, 4, 5, or 6 attributes.
B.7 Noise Addition: PUMS Dataset

Figure B.36: SDC: Noise Addition, Dataset: PUMS, Noise Amount: 25%. Distribution of average entropy for 30 perturbed files when SK is 1, 2, 3, 4, 5, or 6 attributes.
B.7 Noise Addition: PUMS Dataset

Figure B.37: SDC: Noise Addition, Dataset: PUMS, Noise Amount: 75%. Distribution of average initial entropy and average entropy for different 1, and 2 attribute combinations.
Figure B.38: SDC: Noise Addition, Dataset: PUMS, Noise Amount: 50%. Distribution of average initial entropy and average entropy for different 1, and 2 attribute combinations.
Figure B.39: SDC: Noise Addition, Dataset: PUMS, Noise Amount: 25%. Distribution of average initial entropy and average entropy for different 1, and 2 attribute combinations
B.8 Noise Addition: WBC Dataset

Figure B.40: SDC: Noise Addition, Dataset: WBC, Noise Amount: 100%. Distribution of average initial entropy for 30 perturbed files when SK is 1, 2, 3, 5, 7, or 9 attributes.
Figure B.41: SDC: Noise Addition, Dataset: WBC, Noise Amount: 100%. Distribution of average entropy for 30 perturbed files when SK is 1, 2, 3, 5, 7, or 9 attributes.
B.8 Noise Addition: WBC Dataset

Figure B.42: SDC: Noise Addition, Dataset: WBC, Noise Amount: 100%. Distribution of the average initial entropy and the average entropy for different 1, 2 attribute combinations.
B.9 Noise Addition: WINE Dataset

Figure B.43: SDC: Noise Addition, Dataset: WINE, Noise Amount: 100%. Distribution of average initial entropy for 30 perturbed files when SK is 1, 2, 3, 5, 7, 9, 11, or 13 attributes.
Figure B.44: SDC: Noise Addition, Dataset: WINE, Noise Amount: 100%. Distribution of average entropy for 30 perturbed files when SK is 1, 2, 3, 5, 7, 9, 11, or 13 attributes.
B.9 Noise Addition: WINE Dataset

Figure B.45: SDC: Noise Addition, Dataset: WINE, Noise Amount: 100%. Distribution of the average initial entropy and the average entropy for different 1, 2 attribute combinations.
Bibliography


