Doctoral theses at NTNU, 2022:20

Edlira Martiri

Honey Templates: a Protection Mechanism for Biometric Systems

Doctoral thesis

NTNU
Norwegian University of Science and Technology
Thesis for the Degree of Philosophiae Doctor
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

NTNU
Norwegian University of Science and Technology

Edlira Martiri

Honey Templates: a Protection Mechanism for Biometric Systems

Thesis for the Degree of Philosophiae Doctor

Gjøvik, January 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the Degree of Philosophiae Doctor

Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

ISBN 978-82-326-6107-7 (printed ver.)
ISBN 978-82-326-6970-7 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)

Doctoral theses at NTNU, 2022:20

Printed by NTNU Grafisk senter

Contents

List of Tables **xi**

List of Figures **xvii**

0 Preface **3**

0.1 Introduction 3

0.2 The need for security 4

0.3 List of papers 6

1 Thesis Description **13**

1.1 Topic motivation 13

1.2 Thesis description 14

1.2.1 Enrolment and verification 15

1.2.2 State-of-art 17

1.2.3 Masquerade attack and honeywords 18

1.2.4 Proposed template generation scheme 19

1.3 Research Questions 21

1.4 Contribution of this thesis 23

1.5 Structure of the Dissertation 24

2 Honey Objects in System Security: the property of Indistinguishability 29

2.1 Introduction 29

2.2 A generic honey-objects system design 30

2.3 Honeywords 31

2.4 The property of Indistinguishability in a cryptographic context . . 32

2.5 Indistinguishability as a security concept 35

2.6 Brief chapter summary 38

3 Honey Templates as a Biometric Protection Mechanism 41

3.1 Introduction to biometric systems: threats and vulnerabilities . . . 41

3.2 Biometric templates protection mechanisms 43

3.2.1 Privacy risks and standard security requirements 45

3.2.2 Categories of template protection schemes 47

3.2.3 Protection criteria 50

3.3 The concept of honeywords in the biometric context 51

3.3.1 Architecture design of a biometric honey templates system 53

3.4 Honey Templates first proof of concept: **Algorithm I** 55

3.5 Security and Irreversibility evaluations 59

3.5.1 Irreversibility evaluation tests and results 61

3.5.2 Performance comparison between protected and unprotected case 63

3.5.3 Recognition performance evaluation 65

3.6 Conclusions and further work 67

3.7 Brief chapter summary 67

4 Biometric Honey Templates Adoption in Feature Transformation Mechanisms 69

4.1 Problem statement 69

4.2 Definitions for a honey-based Defender-Attacker game 70

4.3 Experimental setup for mimicking D: **Algorithm II** 71

4.3.1 Subjects database creation (DB^*) 71

4.3.2 Feature extraction and template protection 72

4.4 Recognition accuracy results 73

4.5 A classification challenge for honey templates generation 76

4.6 Testing the Defender/Attacker game 77

4.6.1 Classification algorithms 78

4.7 Results and algorithm analysis 80

4.8 Brief chapter summary 83

5 Biometric Honey Templates Adoption with Bloom Filters 85

5.1 Introduction 85

5.1.1 Masquerade Attacks and Honey Templates 86

5.1.2 Biometric Template Protection and Bloom Filters 86

5.2 Proposed System: Honey Bloom Filter Template Generation . . . 87

5.2.1 Honey Templates 87

5.2.2 Bloom Filter Based Protected Templates 87

5.3 Experimental Framework: **Algorithm III** 90

5.3.1 Database and Verification System 90

5.3.2 Experimental protocol 91

5.4 Experimental Evaluation 92

5.4.1 Performance Evaluation 92

5.4.2 Indistinguishability in the Protected Domain 94

5.4.3 Indistinguishability and Irreversibility Evaluation in the Unprotected Domain 97

5.5 Conclusions 98

5.6 Brief chapter summary 98

6 Biometric Honey Templates Adoption with Fuzzy Commitment in a Hybrid Scheme 101

6.1 Introduction 102

6.2 Fuzzy Commitment and binarization conditions 102

6.3 Theorem of Augmentation 105

6.4 Hybrid scheme description for template protection 107

6.5 Framework description of template testing in a face verification honey-based system: **Algorithm IV** 109

6.5.1 Database properties 109

6.5.2 Testing framework 110

6.6 Tests results for classification and system performance 113

6.7 Security analysis and attack scenarios of the hybrid honey-based protection scheme 114

6.7.1 Scheme components security 114

6.7.2 Attack scenarios 115

6.8 Conclusions 117

6.9 Brief chapter summary 118

7 Automatic and Human Classifiers Comparison 119

7.1 Introduction 119

7.2 Honey-based biometric system 120

7.2.1 Architecture design 120

7.3 Attack scenarios on Honey-based system 122

7.3.1 Attacker's perspective on existing honey-based biometric system . . . 122

7.3.2 PCA for feature vectors generation . . . 123

7.3.3 Template protection on honey-based system: **Algorithm V** 123

7.3.4 Secret-based Random Projection as a protection mechanism on feature vectors . . . 125

7.4 Template distinguishability tests . . . 128

7.4.1 Pre-images classification . . . 128

7.4.2 Pre-images classification with human testers . . . 131

7.5 Conclusions . . . 133

7.6 Brief chapter summary . . . 133

8 Measuring the Attacker's Confidence on a Honey-based System 135

8.1 Introduction . . . 135

8.2 Bayes inference for templates predictability . . . 136

8.3 Designing attacker's confidence . . . 140

8.3.1 CASE A: Attacker's confidence for tuned template generation . . . 140

8.3.2 CASE B: Attacker's confidence considering real-life classifiers . . . 146

8.4 Experimental setup and results: **Algorithm VI** . . . 148

8.4.1 Defender/Attacker simulation . . . 148

8.4.2 Systems performance evaluation and results . . . 150

8.4.3 Classification tests . . . 152

8.4.4 Experimental attacker's confidence . . . 154

8.5 Conclusions . . . 155

8.6 Brief chapter summary . . . 155

9 Thesis Conclusions 159

9.1 Thesis overview 159
9.2 Research Questions Verification 163
9.3 Future work 169

A PassGAN for Artificial Passwords Generation 171

A.1 Introduction 171
A.2 Background and Related Works 173
A.2.1 PassGAN 173
A.3 Our Work: **Algorithm VII** 174
A.3.1 Game Situation 174
A.3.2 Dataset 175
A.3.3 Experiment Setup 175
A.3.4 Performance Evaluation 176
A.4 Experiment results and conclusions 177
A.5 Brief chapter summary 179

List of Tables

3.1 Threats of a biometric system 44

3.2 Advantages and limitations of BTP schemes 50

3.3 Classification of protected templates by SVM for different values of w_1, w_2, w_3. 62

3.4 Correlation values between plain feature vectors; selected and unselected vectors; and the protected sugar template. 63

3.5 Correlation values between selected and unselected vectors and protected honey template 63

3.6 EER for sugar and honey templates for different values of w_1, w_2, w_3. 64

3.7 EER for unprotected and protected case for different values of w_1, w_2, w_3. 65

4.1 EER after feature extraction of Defender and Attacker systems for three **different auxiliary databases**. 74

4.2 EER after feature extraction of Defender and Attacker systems for **different biometric records**. 75

4.3 EER after feature extraction of Defender and Attacker systems in the protected domain for **different auxiliary matrices** AM, without honey templates. 76

4.4 EER after feature extraction of Defender and Attacker systems in the protected domain for **different honey features**. 76

4.5 Classification accuracy, and training time for $K = 12$ different categories of classifiers, for $q = 100$ in the training set. 81

4.6 Classification rates (FSTR and FHTR) of a complex Tree Decision classifier for different training set sizes q. Two sides of the table relate to Defender (D) and Attacker (A). 82

4.7 Average classification rates (FSTR and FHTR) of an Attacker System for discrimination of sugar and honey templates in a Defender database (considering x_1). 83

5.1 **Performance Evaluation**: EER and FNMRs at FMR = 0.1% for the baseline unprotected system, the Bloom filter based BTP (BF system) and the proposed scheme (BF + HT system). 93

5.2 **Template Classification.** Accuracy of classifiers' algorithms, and their processing time (corresponding to patterned templates of 808 features in the interval 401-450). 96

5.3 **Template classification.** FMR and FNMR rates of two classification algorithms for different intervals and training sets. 96

6.1 Excerpt from the accepted combinations between m (length of s-values) and t (number of error corrections) for a codeword length $n = 1023$. 104

6.2 Performance results after Random Projection (Level II, left side), and after Binarization (Level III, right side) for 5 rounds. 114

6.3 Classification rates for tests 6, 7, and 9. (Level II and III) 115

6.4 Attacker options and actions in a biometric system when honey templates are implemented 116

7.1 Statistics on the error rates and average time of pre-image classifiers. 132

8.1 Theoretical evaluation of probabilities for two scenarios: straightforward generation of sugar and honey templates (Example 1), and tuned generation of templates having $q = 30\%$ (Example 2). . . . 141

8.2 Theoretical evaluation of probabilities for two scenarios: straightforward generation of sugar and honey templates (Example 3), and tuned generation of templates having $q = 30\%$ (Example 4), and $q = 70\%$ (Example 5). The classification rate of the classifier is $c^{HT} = 0.2$. 147

8.3 EER comparison after PCA feature extraction. 152

8.4 EER comparison after Random Projection. 152

8.5 EER comparison after Random Projection with honey templates. . 153

8.6 Classification rates of sugar and honey templates after Random Projection. 153

8.7 Classification rates of sugar and honey templates after Random Projection. 153

8.8 Simulated system probabilities showing attacker's confidence with c_A^{ST}. 154

8.9 Simulated attacker's confidences on sugar and honey templates, calculated for different portions of ST. 154

9.1 Performance results after Random Projection (Level II, left side), and after Binarization (Level III, right side) for 5 rounds. 165

9.2 Classification rates for tests 6, 7, and 9. (Level II and III) 165

List of Figures

1.1 Subjects enrolment (left) and verification (right) in a biometric verification system based on honey templates. 16

1.2 Classifier training using labeled ground-truth sugar protected templates and honey protected templates. 20

1.3 Sugar and honey template generation in an iterative way. 21

1.4 Honey templates based fuzzy commitment to thwart a hash-cracking based masquerade attack. 22

2.1 A generalized Honey Objects database architecture design. 30

3.1 Architecture for a honey templates based biometric system: enrolment (up) and verification (down) 54

3.2 Samples in DBaux which is used for eigen faces training and used as AD_i by BTPS (up), Samples in DB^t which is used for recognition performance (down). 56

3.3 Standard deviation values for PCA coefficiens of the testing database DB^t. 58

3.4 Dynamic range of protected Sugar (a), and protected Honey Templates (b), with the mask vector mechanism. 60

3.5 Honey templates generation diagram in the mask vector BTP. . . 61

3.6 Recognition performance comparison: (a) Unprotected templates: EER = 0.1223 (b) BTPS with honey templates: EER = 0.1285. . . 66

4.1 Samples of a subject in *faces94* database. 72

4.2 *Eigenfaces* from Defender (a) and Attacker (b). 73

4.3 An inclusive multi-classification tool for biometric templates distinguishability. 77

5.1 Architecture for a honey templates based biometric system (113): enrolment (left) and verification (right) 88

5.2 Database generation: in order to ensure a higher security, templates and indices will be stored in databases which should be handled by independent systems: the set PT_i in the Biometric Database and the index L_i in the Honey Checker Database 89

5.3 Bloom filter templates generation. 89

5.4 **Performance Evaluation**. Detection Error Trade-Off (DET) curves for the baseline unprotected system, the Bloom filter based BTP (BF system) and the proposed scheme (BF + HT system). 92

5.5 **Scores Analysis**. Score distributions for sugar (S_{Sugar}, purple) and honey (blue, S_{Honey}) templates, when compared to the probe template (left). The difference between the $K = 10$ honey scores and their corresponding sugar scores ($S^i_{Honey} - S_{Sugar}$ for $i = 1, \ldots, K$) is depicted on the right. 93

5.6 **Feature Selection**: graph showing number of features in each interval of activated bits. For example, while 9,610 features contain 1-50 activated bits, 516 features contain 601-650 activated bits. Therefore, the length of the bars also represents the number of features which would be selected should the interval be chosen. Larger number of features yield larger patterned templates, and vice-versa. 95

5.7 **Irreversibility and Indistinguishability Evaluation**. Genuine (dashed green) and impostor (dashed red) HDs between real unprotected templates, compared to the HDs between reconstructed sugar templates (solid purple) and their corresponding real template, and between reconstructed honey templates (solid blue) and their corresponding real template. 98

6.1 Hybrid scheme for sugar and honey template protection: (a) Generation of sugar templates (SB^{FT}) and honey templates (HB^{FT}) after Feature Transformation (Scheme A and buffer b); (b) Generation of sugar templates (ST^{FC}) and honey templates (HT^{FC}) after Fuzzy Commitment (Scheme B). 108

6.2 Classifier training using the set CTrain from labeled ground-truth sugar protected templates (SB_i^{FT} labeled as *1*) and honey protected templates (HB_{ij}^{FT} labeled as *0*). 110

6.3 Classification and performance test framework for a hybrid honey-based protection mechanism. 112

7.1 (a) Samples in training database which is used for eigenfaces generation. (b) Samples in testing database which is used for recognition performance (72) 124

7.2 Mean face reconstruction of the set of images in the training database. 125

7.3 Protection mechanism on sugar and honey templates 125

7.4 Example of first sample of user u1: (a) raw image at enrollment; (b) image of constructed sugar template; and (c) images of a subset of the corresponding honey templates 126

7.5 Secret-based random projection scheme for template protection . . 127

7.6 EER of sugar templates with secret-based random projection as a protection mechanism. 128

7.7 Mean face of PCA_{Train} set. 129

7.8 Eigenfaces: generated from PCA_{Train} set. 130

7.9 System GUI for the volunteer classification testing. 132

8.1 Comparison of probabilities $P\left(ST_o|HT_s\right)$ and $P\left(ST_o|ST_s\right)$ for a constant classification rate $c = 0.5$ and different sugar templates portions r^{ST}. 141

8.2 Comparison of probabilities $P\left(HT_o|HT_s\right)$ and $P\left(HT_o|ST_s\right)$ for a constant classification rate $c = 0.5$ and different sugar templates portions r^{ST}. 142

8.3 Comparison of probabilities $P\left(ST_o|ST_s\right)$ and $P\left(ST_o|HT_s\right)$ for a constant portion of sugar templates $r^{ST} = 0.3$ and different classification rates c^{ST}. 147

8.4 Comparison of probabilities $P\left(HT_o|ST_s\right)$ and $P\left(HT_o|HT_s\right)$ for a constant portion of sugar templates $r^{HT} = 0.7$ and different classification rates c^{HT}. 148

8.5 Samples of a subject in *faces94* database. 149

8.6 *Eigenfaces* from Defender (a) and Attacker (b). 150

8.7 System recognition performance for Defender (up) and Attacker (down), after PCA feature extraction. 151

8.8 System recognition performance for Defender (up) and Attacker (down), after Random Projection. 156

8.9 System recognition performance for Defender (up) and Attacker (down), after Random Projection, with honey templates. 157

A.1 The attacker and the defender game flow. 174

A.2 The attacker's success rate given only 1 opportunity to guess the correct password in scenario 1. In this figure, the attacker and the defender use several iteration variations. 178

A.3 The attacker's success rate given more than 1 opportunity to guess the correct password in scenario 1. In this figure, the defender and the attacker use 195000 iterations. 178

A.4 The attacker's success rate given only 1 opportunity to guess the correct password in scenario 2. In this figure, the attacker uses a fixed number of iterations (100000), while the defender uses several numbers of iterations. 179

A.5 The attacker and the defender use different datasets. 180

A.6 The attacker and the defender use the same datasets. 180

Dedicated to my family.

Declaration of Authorship

I, Edlira Martiri, hereby declare that this thesis and the work presented in it is entirely my own. Where I have consulted the work of others, this is always clearly stated.

Signed:

Date:

Chapter 0

Preface

0.1 Introduction

This thesis is presented to the Department of Information Security and Communication Technology at the Faculty of Information Technology and Electrical Engineering, Norwegian University of Science and Technology, Gjovik, as partial fulfilment of the requirements for the degree of Philosophiae Doctor, Ph.D. The work presented here is mainly based on the extensive experimental work that can be considered as a novel protection mechanism for the biometric templates stored in a biometric system for access control. The idea is inspired by the work of Ari Juels, Cornell University, NY, USA, and Ronald L. Rivest from MIT. In their paper "Honeywords: Making password-cracking detectable", presented at the *ACM SIGSAG* conference in 2013, they propose a mechanism of generating fake passwords and storing them together with the real one. This mechanism camouflages a genuine password in order to make it difficult to distinguish from possible leakage of stealth.

This thesis has started from the honeyword idea and was further elaborated on and applied in the biometric context, specifically the templates. Considering that biometric data have their pecularities and user authentication is different from that of passwords we found ourselves in front of a new challenge and a new research path. How can biometric templates be generated? Can they be distinguished from real ones if an attacker possesses them? These were only the first questions we worked in this thesis, which layed the way to many new interesting ones to come.

0.2 The need for security

Deception based techniques are increasingly gaining interest within the information security community [7]. Even if we protect our systems and data to a big extent, attackers still penetrate the system and operate without being detected, hence being able to steal sensitive data. Therefore, when such an attacker overcomes traditional detection mechanisms, we would like to have the ability to lead him astray by deceiving him and drawing his attention to non-sensitive data, which are false or misleading.

In cybersecurity, deception and decoy-based mechanisms have been used for more than two decades to detect data leakage and intrusions. Among other examples, *honeypot* servers appear to be a legitimate part of a system, but are actually isolated and monitored, in order to lure and block attackers [101]. Other approaches include placing *honeyfiles* [119] in the system to attract hackers and detect them when they access those files, and the most recent *honeywords* scheme [59] to trace leaked hashed passwords. Person authentication has become of utmost importance. In contrast to traditional technologies, which relied on PINs or passwords (something that you know or have), biometrics has emerged as an automatic and reliable alternative, in which behavioural or physiological characteristics of the subjects (something that you are) allow his identification. Among other advantages, biometric traits (e.g., face, iris or fingerprint) cannot be lost or forgotten. On the other hand, should a biometric template be stolen, it cannot be replaced.

Given the high sensitivity of biometric data, we need to protect the information handled and stored by biometric systems, so that no biometric information is leaked. In that context, *Biometric template protection* (BTP) technologies [22, 53, 93] offer solutions to privacy preserving biometric authentication. They are commonly categorized as *biometric cryptosystems* [109], where a key is either extracted or bound to a biometric sample, and *cancelable biometrics* [86], where biometric samples are obscured in a non-reversible manner, or hybrid schemes that integrate at least two of the existing techniques.

However, even if biometric references for the enrolled subjects are constructed using BTP technologies, a template can still be stolen from a database and used to impersonate a particular subject without the BTP scheme noticing it. To prevent such kind of attacks, the system is augmented with synthetic templates, an idea which stems from the *Honeywords* approach. It is proposed in [59] for the protection of traditional passwords where several password hashes are stored for each subject: hashes for the real password and for other false passwords or honeywords.

A general framework for the application of the honeywords concept to biometric authentication system, known as *Honey Templates*, was for the first time proposed by the author of this thesis in [73], and a particular case study on face verification based on Eigenfaces was applied on a small scale database [116], showing its effectiveness in terms of both irreversibility and recognition performance (see Sect. 5.1.1).

In all those systems, the real data (*sugar* object) is hidden among the false *honey* objects in the system. The latter should fulfill two requirements [58]:

- *Indistinguishability*: to deceive an attacker, honey objects must be hard to distinguish from sugar objects.
- *Secrecy*: the sugar object should remain secret among the honey objects.

As a consequence, if any of the honey objects were used, the attacker will gain no information about the sugar object and the system will detect the intrusion and trigger an alarm.

In the context of access control, and in particular, biometric authentication systems, one further concern may rise: masquerade attacks and identity theft. Should an attacker steal the reference (protected) template, he could use it to fraudulently access the system until the leakage is detected. To prevent such impersonation, the *Honey Templates* scheme adapts the *honeywords* methodology to the secure storage of biometric templates [73].

In this thesis we tailor the "honeywords" idea, which was proposed to detect the hashed password cracking, to enable the detectability of biometric template database leakage. However, unlike passwords, biometric features encoded in a template cannot be renewed after being cracked and thus not straightforwardly able to be protected by the honeyword idea. To enable the honeyword idea on biometrics, diversifiability (and thus renewability) is required on the biometric features. We propose to use BTPS for his purpose in this paper and present a machine learning based protected template generation protocol to ensure the best anonymity of the generated sugar template (from a user's genuine biometric feature) among other honey ones (from synthesized biometric features).

While masquerade attacks are possible to cope with by better anti-spoofing technologies, they are not very possible to be completely prevented. To discourage both the physical and the digital masquerade attacks, we do this by empowering the system with detectability of the leakage of protected templates. Juels in [59] proposes the idea of *honeywords* used on passwords and we extend this idea to the biometric templates. We elaborate further the architecture design for a biometric system using BTPS-based honey templates and apply this idea on faces. We also evaluate the security, biometric recognition performance and irreversibility of honey face templates.

The honeywords method provides us a systematic way to counter the masquerade attack against protected biometric templates. It resorts to probability (i.e. information-theoretic security) instead of computational complexity based security to cope with the crackable-hash assumption. In the biometric context, most databases are facing the same challenges.

Firstly, templates must be constructed in such a way that an adversary is not able to distinguish a sugar from a honey one, even if he: breaks the protection mechanisms; uses automatic tools such as classifiers; or tries to visually capture differences of honey and sugar templates pre-images to differentiate them. Secondly, the sugar template must be placed in a random position in the user database entry, or user data file, among the honey templates and this specific index must be known only to the honeychecker.

We note that the aim of our approach on biometric templates, as well as the honeywords method, is not to lure the intruders with fake data, but to provide a means to alert the system that an internal or external adversary had access to the users' data and used them back: in other words that there have been system attack, information leakage, and user impersonation (masquerade attack).

Deception techniques used in Information Security are recently applied in a biometric context, offering not only protection mechanisms on the templates, but also they are capable of notifying if database leakage and user impersonation occurred. The real challenge in designing a honey-based system is *indistinguishability* between real and synthetic templates. This is in fact also a security requirement of the system.

In this thesis we present six algorithms: one is applied on passwords, and five are protection schemes which generate honey templates by preserving the indistinguishability property. Algorithms' implementation and experiments show very promising results on the final sets of *sweet templates*, meaning that they offer high level of dissimilarity, by preserving the system recognition performance.

0.3 List of papers

The thesis is presented as a monography, but it is built upon ten papers that comprise the main body of research. Six of these papers contribute directly to the work of this thesis, and four papers that paved the way to new directions and works. Moreover, results and analysis from this thesis are also published in four more national conferences in Albania and presented in different activities, symposiums, and security academies where the author was invited.

- CHAPTER 3

  Bian YANG, Edlira MARTIRI, “Using Honey Templates to augment Hashed based Biometric Template Protection”, 39th IEEE International Conference on Computers, Software and Applications, COMPSAC, July, 2015.

- CHAPTER 4

  Edlira MARTIRI, Bian YANG, Christoph BUSCH, “Protected Honey Face Templates”, 14th IEEE International Conference of the Biometrics Special Interest Group (BIOSIG), September, 2015.

- CHAPTER 5

  Edlira Martiri, Marta Gomez-Barrero, Bian Yang, Christoph Busch, "Biometric template protection based on Bloom filters and honey templates", IET Biometrics Journal, DOI: 10.1049/iet-bmt.2015.0111, ISSN 2047-4938, 2016.

- CHAPTER 6

  Edlira Martiri, Bian Yang, “A cryptographic evaluation of Biometric Honey Templates’ space and their implementation in a Hybrid Protection Scheme”, IET Biometrics Journal, December 2021 (under review).

- CHAPTER 7

  Edlira Martiri, Bian Yang, Muhammad Ali Fauzi, “Indistinguishability of Biometric Honey Templates: Comparing Human Testers and SVM Classifiers”, IEEE, International Conference on Computational Science and Computational Intelligence, CSCI-ISCW track: Cyber Warfare, Cyber Defense, Cyber Security, Las Vegas, USA, December, 2020.

- CHAPTER 8

  Edlira Martiri, Bian Yang, “On the predictability of biometric honey templates, based on Bayesian inference”,10th International Conference on Communication and Network Security, University of Tokyo, Japan, November, 2020.

- APPENDIX A

  Muhammad Ali Fauzi, Bian Yang, Edlira Martiri, “PassGAN-Based Honeywords System”, International Conference on Computational Science and Computational Intelligence, IEEE Research Track/Symposium on CSCI-ISCW: Cyber Warfare, Cyber Defense, and Cyber Security, December, 2019, Las Vegas, USA.

- More papers on PassGan

  Muhammad Ali Fauzi, Bian Yang, Edlira Martiri, “Password Guessing-Based Legacy-UI Honeywords Generation Strategies for Achieving Flatness”, 44th IEEE Annual Computers, Software, and Applications Conference (COMPSAC), July, 2020, Madrid, Spain.

  Muhammad Ali Fauzi, Bian Yang, Edlira Martiri, “PassGAN Based Honeywords System for Machine-Generated Passwords Database”, 5th IEEE International Conference on Intelligent Data and Security (IEEE IDS 2020), May 2020, Baltimore, USA.

  Muhammad Ali Fauzi, Bian Yang, Edlira Martiri, “PassGAN for Honeywords: Evaluating the Defender and the Attacker Strategies”, International Conference of Advanced Computing and Informatics (Information Security track), Springer, April, 2020, Casablanca, Morocco.

Acknowledgments

From all people I encountered during this PhD thesis work at NTNU, I would especially acknowledge and thank my main supervisor, Bian Yang, for guiding and supervising me, supporting and encouraging, for being a great motivator for this topic and positively critical while shaping together this novel idea. Our discussions and brainstorming sessions are now part of my work experience, passing them to my students too. Having the scientific aparatus is a must in this journey, but being a teacher is a gift that not everyone could possess. Secondly, my sincere thanks go to Christoph Busch, for believing in me, and introducing me to our great team at the Norwegian Biometrics Laboratory, where I met some of the best professors and new researchers in Security and Biometrics. Christoph, I learned a lot from you too, you were a role model and a great team leader. You both welcomed me to our department and really made this work come to light.

I would like to express my respect and gratitude to two very well-known authors, upon which papers this topic was inspired. The first one is Ari Juels from Cornell University, New York, former Chief Scientist of RSA, Director of RSA Laboratories, and a Distinguished Engineer at EMC (now Dell EMC). And the second is Ronald L. Rivest, at MIT, member of the lab's Theory of Computation Group and a founder of its Cryptography and Information Security Group. He is also a founder of RSA Data Security, now named RSA Security (the security division of EMC), Verisign, and Peppercoin.

Moreover, I would like to thank all the people working at the administration and staff of the department of Statistics and Applied Informatics, University of Tirana, Albania, who first introduced me the possibility of a PhD in Norway, this amazing and snowy country, so different from the sunny little Albania, but now the place I call my second home. Thank you to the Academic Exchange for Progress project

coordinators from Albania, Kosovo, Macedonia and Norway, and the Albanian IT Association (AITA), especially prof. Kozeta Sevrani for giving me this opportunity and supporting us as new generation of researchers, like few people do. All the friends I met during my stay in Gjovik, starting from my office colleagues Gaute and Vivek, together with Zenun, Kushtrim, Vasilis, and Dimitra, were the ones I would always stay and discuss, have lots of coffee, eat our quick lunches, get back to our computers, exchange opinions on science, culture, life, and of course laugh a lot. Indeed, this thesis has been a long and exciting graph of exploration paths of new scientific concepts and topics; a set of sleepless hours into creativity, thinking, writing, and coding; a great cultural experience of getting to know a new country, language and friends with whom we could understand that North and South are only two sides of the horizon.

It is a pleasure to thank all of them who I encountered and helped me out, since without their support, patience, and sometimes sacrifice, it would not have been possible, but these few paragraphs unfortunately can not fill all their names and contributions. From all of them, my beloved husband Ergys is the one who never stopped cheering me up, and walked with me side by side in every step, together with our two daughters Lejla and Hana. Our families were always the greatest support to all our endeavours, with their understanding, positivity and encouragement. Mom, dad, Joni, I can never thank you enough. Even though we were distant, you always reminded me that life is beautiful, full of colors, that every challenge and difficulty is finally rewarded. The same did my friends from University of Tirana, at THPM group, with their energy and dedication, positivity and good friendship.

Finally, the opportunity to undertake and perform this study has been possible because of the financial support from AEP, ORIGINS, Fujitsu, and PIDaaS projects. I am really grateful and honored for getting this opportunity.

Edlira Martiri
From Tirana to Gjovik

January, 2022

Chapter 1

Thesis Description

1.1 Topic motivation

Nowadays almost every aspect of our life is information driven. To achieve protection, data integration, or secure communication, there exists a plethora of systems and tools, at the heart of which stand cryptographic mechanisms. Nevertheless, strong cryptographic tools do not imply strong security. From principles of security, *the security of a system depends on the weakest link*. Similarly, *the security of a cryptographic mechanism* depends on the most vulnerable part of its implementation. In this regard, achieving as strong as possible cryptographic algorithms might not be the aim. These mechanisms lie in many security services, such as access controls.

Among the different access control mechanisms, essential part of any secure system, to gain access in a specific area, cards and tokens can be provided, passwords and PINs can be used, or subjects can expose their biometric traits. Defined in (49), *biometric systems enable automated recognition of individuals based on their behavioural or physical characteristics*. After *enrolment*, samples are processed in the form of a *template*. They are then stored in a protected way in the database, and used for *identification*, or *verification* of subjects. Their improper use may be detrimental for the system, and if they fall in the wrong hands, they could be used to impersonate a subject, or could be used by an adversary for further attacks sometimes at later times, making the scenario (from the point of view of a security officer) full of ambiguities. In a generic biometric system, according to ISO/IEC 24745 standard (50), biometric templates contain mainly two parts: (1) *a pseudonymous identifier* (*PI*), it represents an individual as a protected identity, e.g. a transformed template; and (2) *auxiliary data* (*AD*), used to reconstruct pseudonymous identifiers during verification, e.g. transformation parameters or keys.

As it can be implied, *PI* contains sensitive information, and even though there have been many efforts in protecting it, breaches still occur. This means that when thinking of a template protection mechanism, a more global solution must be considered. These solutions should be easily adoptable to real-life systems, fast, and cost-effective. From the scientific community there have been many efforts in designing, testing, and implementing such template protection schemes. But very few schemes provide protection and information leakage detection capabilities.

Hash based biometric template protection schemes (BTPS), such as fuzzy commitment, fuzzy vault, and secure sketch, address the privacy leakage concern on the plain biometric template storage in a database through using cryptographic hash calculation for template verification. However, cryptographic hashes have only computational security whose being cracked shall leak the biometric feature in these BTPS; and furthermore, existing BTPS are rarely able to detect during a verification process whether a probe template has been leaked from the database or not (i.e., being used by an imposter or a genuine user).

1.2 Thesis description

In information security, data hiding and deception mechanisms are extensively used to protect information from possible adversaries (8), such as: 1) *Masking*: attackers may hide scripts in the background of a webpage; 2) *Repackaging*: from the system perspective, data can be repackaged as something else; 3) *Dazzling*: real objects are confused with synthetic objects; 4) *Mimicking*: as an example, phishing is the action of mimicking a real website; 5) *Inventing*: if mimicking is not possible, invention can be used to add new systems or components (e.g. honeypots) ; 6) *Decoying*: when system parts are added to lure attackers and lead them astray from more valuable system parts.

Specifically, *honey objects* are for more than a decade offering a wide range of solutions applied at system or data level. A perfect *honey object* (H_{Ob})is completely indistinguishable from the real or *sugar object* (S_{Ob}) that it is trying to hide, meaning that the level of "mimicry" of the mechanism must be high enough to dazzle attackers. If a set of k-synthetic objects is added, the probability of guessing that a generic object G_{Ob} from the whole set is $P(G_{Ob} \equiv S_{Ob}) = 1/(k+1)$. This means that the security of the scheme is probability-based. The higher the number of added honey objects, the lower the probability of guessing the sugar one. As Shpilrain puts it in (95) (where decoy objects in a public-key setting are applied) we can tolerate leakage of information as long as the probability of recovering the secret information is lower than a tolerate non-zero probability of having a security breach, which always exists.

In general, honey objects have to satisfy two properties (56). First, a honey object must be *indistinguishable* if compared to a real object. In (94), from the idea of perfect secrecy, a perfectly believable decoy is defined a honey object which is ideally chosen with probability $p = 0.5$ over all trials. The second property of honey objects is *secrecy*. After generating a set of honey objects the real one must be randomly placed among them, in such a way that the possible attacker should not be able to define/predict it.

One of the most well-known examples are *honeypots* (machines inserted in a network to attract intruders' attention from more important machines (100)). There are also *honeyfarms*, a network of honeypots, or *honeyfiles* which are synthetic files carrying fake data to dazzle attackers and lead them astray from real files (116). A similar work was published in 2013 by Juels, which involves synthetic passwords (58). According to this technique, named *honeywords*, n-generated passwords are hashed and stored in the same storage space with the real user's hashed password. If an attacker will posses the database content it should be impossible for him/her to guess the real one. This is the main idea from which the *biometric honey template* stems.

The main challenge of *honeywords* was to generate passwords as much look-alike as human generated passwords, but not only. Since many tests can be performed on passwords, another challenge to be solved was the fact that false passwords should match their user's profile (vocabulary words, important dates, gender, etc.), otherwise a knowledgeable attacker might be able to differentiate real passwords from machine generated passwords. The same challenge remains for *Honey Templates* too. An attacker should not be able to differentiate the two categories of templates. Even if he has full knowledge on the system (i.e. the fact that Honey Templates are deployed, algorithm parameters, classifier rates, etc.), and possesses sophisticated classification tools, his chances in distinguishing templates should be infinitesimally small.

1.2.1 Enrolment and verification

To better understand how the honey-based system is applied in the biometrics context let's consider the enrolment and verification phases presented schematically in Fig. 1.1.

During **enrolment** a user u_i presents its biometric trait and the feature vector T_i is extracted. On it is applied the protection mechanism and as a result a set of

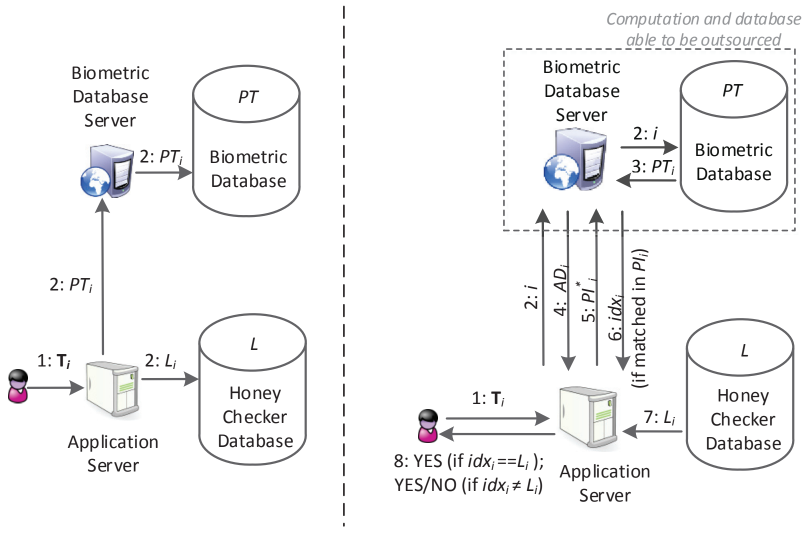


Figure 1.1: Subjects enrolment (left) and verification (right) in a biometric verification system based on honey templates.

protected templates PT_i is stored in the biometric database in a randomized order. At the same time, in the Honey Checker Database will be stored the index L_i of the genuine template.

During **verification** user u_i presents the plain features T_i and personal information claiming to be u_i. The system finds his/her identity, the set PT_i of templates, and checks if T_i decodes with success with one of the stored templates. If YES, meaning that there exists a $j \in \{1, ..., k+1\}$ where the decoding is successful, the system requests from the honey checker the index L_i. If the index L_i is equal to j, the user is authenticated, otherwise, if we have a successful decoding with a $j \neq L_i$, this means that the user is trying to authenticate with one of the honey templates, which can be considered as database information leakage and user impersonation.

In this thesis we tailor the honeywords idea, which was proposed to detect the hashed password cracking to enable the detectability of biometric template database leakage. However, unlike passwords, biometric features encoded in a template cannot be renewed after being cracked and thus not straightforwardly able to be proteted by the honeyword idea.

To enable the honeyword idea on biometrics, diversifiability (and thus renewability) is required on the biometric features. In this research we propose to use, together with the honey objects mechanism, a machine learning based template generation framework to ensure best anonymity of sugar templates (from a users genuine biometric feature) among other honey ones (from synthetized biometric features).

1.2.2 State-of-art

Biometric template protection schemes (BTPS) (51, 92) can provide privacy protection, i.e., irreversibility and unlinkability (5), to biometric templates stored in a biometric database, which ideally makes a leaked protected template much less concerned in terms of biometric information leakage from the user's perspective. However, from the system perspective, a template leaked in its protected form from the biometric database can still pose serious threats to the system's security. The exploitation can be made in both the standalone biometric system scenario (e.g., personal device access control) and the remote biometric authentication scenario (e.g., online services, etc.).

In the former case, an adversary can find a pre-image of the protected template via an off-line brute-force attack and use the pre-image to make a faked physical biometric characteristic (e.g., a gummy finger) in order to spoof the biometric sensor. Note the pre-image found to match the protected template may not be a "look-alike" mate to the genuine plain template, i.e., the pseudo-authorized-leakage case described by Simoens et. al. (98). This can be regarded as a physical masquerade attack. In the latter case, it can be possible to hijack a live-generated protected template in the client end and replace it with the leaked template before sending it to the server end for a comparison.

This can be regarded as a digital masquerade attack which reminds us of the Augmented Password-Authenticated Key Agreement (e.g., SRP (46)) used for the similar purpose in password remote authentication. However, unlike passwords which are exact data and thus suitable for discrete logarithm calculations, protected biometric templates could be fuzzy data, e.g., cancelable biometrics (85) and biohashing (103), and thus difficult to directly avail such passwords authentication protocols against the masquerade attack. While masquerade attacks may be thwarted by better anti-spoofing technologies and trusted computing infrastructure, they are impossible to be completely prevented. Instead, in this thesis we resort to the idea of "honeywords" (58) to discourage both the masquerade attacks by enabling the system to detect the leakage of protected templates in the first time their honey peers are presented to the system. However, due to biometrics' permanence, the generated templates cannot be renewed like a password and thus may need BTPS to achieve the renewability. On the other hand, many BTPS, especially the fuzzy schemes (fuzzy commitment (61), fuzzy vault (60), secure sketch (102)), rely on cryptographic hashes which has however only security in the computational sense and is subjected to a brute-force attack. Fortunately, this security concern on fuzzy schemes can be addressed by the honeywords concept.

Note that the essence of a honeyword system is to separate the password authentication data management from the authentication decision making. This data separation enables the data outsourcing to a third party (e.g., a cloud service provider) without too much privacy concern since the third party has no information to identify a sugar word. This kind of data separation could be more useful to outsourcing biometric reference data since biometric data are more heterogeneous in type and size and could thus be more difficulty or with higher cost to manage by the authentication decision making party than by the third party.

1.2.3 Masquerade attack and honeywords

As mentioned, templates leakage from the identity impersonation. The identity database could be filled with passwords, usually stored in their salted-hash form, such as a shadow password file used in UNIX for user authentication, or could be a biometric database with protected templates created by BTPS. For the hashed passwords, an adversary who got a leaked password hash h can employ the brute-force attack to find a pre-image p with its hash value $H(p)$ equal to h. The adversary does not care whether p equals to the legitimate user's genuine password as long as it can be used to impersonate the legitimate user to spoof the system.

Based on this security assumption, the honeywords method was proposed in (58) to hide the genuine password's hash (called *sugar word*) among $(K1)$ randomly generated passwords' hashes (called *honey words*) without storing the labels to distinguish it from other $(K\breve{}1)$ honeywords in the same database. The dynamic range of k could be from 2 to $1,000$ as reasonable choices depending on applications. Thus an attacker who cracked such a honeywords database while also having enough computational power to crack the hashes has but a probability of $1/K$ to probe a sugar word to the system succeeding in launching a masquerade attack. Note that the attacker has only one chance to try his / her luck since the system will be immediately alerted of the fact of the password database leakage if the first try fails (a honey word is probed to the system).

Considering the case of biometric *protected templates* (PT) created by BTPS, for most BTPSs PT databases are facing the same masquerade attack challenge as the passwords database case. For those hash based BTPSs such as fuzzy commitment, fuzzy vault, and secure sketch, if hash cracking is assumed possible, a pre-image of the biometric feature can be estimated according to the cracked pre-image of a hash. Note that this estimation could be an easy task for fuzzy commitment, fuzzy vault, and secure sketch because in all these schemes the biometric features can be reconstructed by simple exclusive-OR and addition operations. Be aware that the Augment PAKE such as SRP, which can be used to prevent the masquerade attack caused by password database leakage, cannot thwart the masquerade attack since the assumption of attacker's ability in hash cracking undermines such password authentication protocols when they are used for those hash based BTPSs.

The honeywords method, however, provides us a systematic level approach to counter the masquerade attack towards protected biometric templates. It resorts to probability (i.e. *information-theoretic security*) instead of *computational complexity based security* to cope with the crackable-hash assumption.

1.2.4 Proposed template generation scheme

Now we consider how to generate a honey biometric template. Creation of honey biometric features can be done either by finding real biometric samples other than from the user's body characteristics or by biometric sample (or feature) synthesis. Taking real biometric samples as honey templates could be not only ethic sensitive but also costly in operation. Biometric sample or feature synthesis can be varied in technical difficulty depending on modalities and feature extraction methods. Here the difficulty should be understood as the difficulty in distinguishing the honey templates generated from synthesized samples (or features) from the sugar template generated from the real sample (or feature).

In this thesis, we try evaluating this "difficulty" from a pattern recognition perspective and propose such a general scheme to obtain such *indistinguishable* sugar templates and honey templates. First, as shown in Figure 1.2, ground-truth sugar and honey protected templates are generated using the same AD, which is generated by a pseudo-random number generator ($PRNG$), from the ground-truth sugar plain biometric features SB and random plain biometric features HB. FT is the feature transformation used to generate protected templates that are able to compare with a distance thresholds (e.g., cancelable biometrics and biohashing).

With the ground-truth protected templates, a machine learning approach (e.g., $SupportVectorMachine$, SVM) can be used to train a classifier C by labelling those sugar templates as '$1'$ and those honey ones as '$0'$. Note that in addition to those honey templates, any random templates, which could be obtained as a pre-image from a brute-force hash cracking process, should be labelled as '$0'$ as well. Obviously, the higher is C's classification error rates, higher is the nondistinguishability of the sugar template among those honey ones.

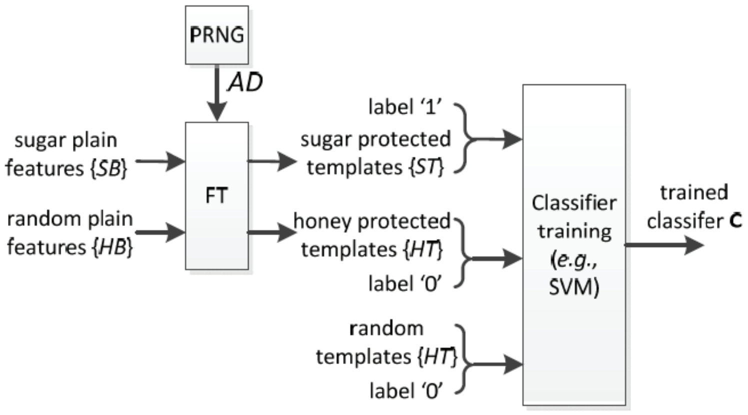

PRNG
AD
sugar plain features {SB}
random plain features {HB}
FT
label '1'
sugar protected templates {ST}
honey protected templates {HT}
label '0'
random templates {HT}
label '0'
Classifier training (e.g., SVM)
trained classifer C

Figure 1.2: Classifier training using labeled ground-truth sugar protected templates and honey protected templates.

While its training is ready, C can be used to test a sugar transformed template ST, as shown in Figure 1.3, to see it can be classified as a honey one (labelled as $'0'$). If yes, this ST can be saved in the biometric database as one of the KPIs associated with the user. If no, a new random AD is adopted to generate a new ST for being tested by C again until the new ST can be labeled as $'0'$. This last AD will be used to generate all honey transformed templates HT as well. In this way, the sugar template is hided among honey templates with a high non-distinguishability in the sense of machine learning based classification. While this is the generic description of the template generation, in our work we have also analyzed and evaluated the attacker's confidence in labelling stolen templates.

Figure 1.4 presents an example using honey biometric protected templates (ST or HT) as the input to the fuzzy commitment scheme. Suppose the secret s's hash$h(s)$ used in fuzzy commitment can be reversed to a pre-image of s, denoted as $p(s)$, an attacker can derive a pre-image $p(BT)$ of the biometric template BT. Now the only challenge to the attacker is to determine whether $p(BT)$ is derived from a sugar or honey protected template?

The attacker may use a classifier C_{mal} to assist in making this decision. If we assume C_{mal} is equivalent in error rates to C (i.e., no more powerful than C in

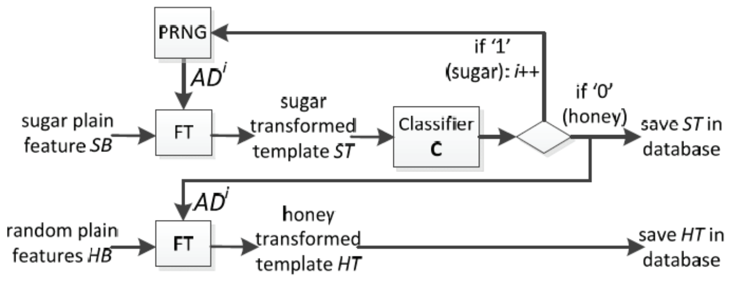


Figure 1.3: Sugar and honey template generation in an iterative way.

distinguishability), C_{mal} may have a high probability to infer a wrong decision since the classification error rate is high. This indicates a low probability to identify the sugar template from out of the K PIs if K is a large number.

To conclude, in this thesis we borrowed the honeywords concept which was designed for detecting leaked passwords and propose a biometric database construction design and architecture design for a biometric system using such a honey templates idea. To get rid of the permanence of biometric features, BTPS was proposed to diversify the plain templates to renewable ones. To make the sugar template non-distinguishable from those honey ones, we proposed a machine learning based template generation scheme.

While use of honey templates in biometrics is explored for the first time in this thesis, we believe both the BTPS method and the honey template construction method have wide room to dwelve into more complex and inclusive aspects as well as improve in both security and recognition performance aspects in the future. We hope this thesis can provoke thoughts and discussions in this field.

1.3 Research Questions

During the path of our research we faced many new challenges, which were then converted to hypotheses and gradually were extensively tested and thoroughly analysed. This work aims to explore the possibility of augmenting a biometric system with synthetic templates in order to camouflage the real ones. To achieve this, the idea of honewords was adopted in this new system setup considering most of the possible dimensions for a secure implementation.

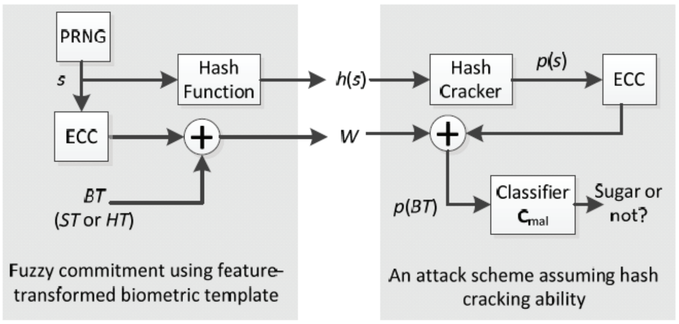

PRNG
s
Hash Function
ECC
BT (ST or HT)
Fuzzy commitment using feature-transformed biometric template
h(s)
W
Hash Cracker
p(s)
ECC
p(BT)
Classifier C_{mal}
Sugar or not?
An attack scheme assuming hash cracking ability

Figure 1.4: Honey templates based fuzzy commitment to thwart a hash-cracking based masquerade attack.

All our work was implemented on face verification biometric systems. Not only we aimed in proving the soundness of the newly presented technique, but we also included new protection functions, hybrid schemes, two-level security analysis, and mainly a new concept on synthetic templates generation, and how to make the system learn from the process. We can truthfully say this work presents an interdisciplinary attempt in the long scientific endeavor which hopefully improves the security of biometric systems and contributes to other topics within information security too.

In fact, this thesis revolves around one main and important idea: how to augment existing or new biometric template protection schemes with honey templates, without deteriorating the system performance. Along this journey we came across other ideas, we extended these questions to different schemes, and we tried to have a complete and inclusive point of view on the topic. Below are listed the main research questions:

- **RQ1.** Can Biometric Template Protection schemes be augmented with Honey Templates, that are indistinguishable from real ones and that has information leakage capabilities?

  RQ1.1. Does this augmentation deteriorate the system performance?

  RQ1.2. Are Equal Error Rates influenced in the same way for different schemes?

  RQ1.3. Can the indistinguishing property between templates be measured with reliable metrics?

RQ1.4. Is there an upper limit in the number of Honey Templates that can be augmented in different schemes?

RQ1.5. Is it possible for an attacker to mimic a honey-based system in terms of its parameters, and use it to classify leaked templates?

- **RQ2.** Can we provide a mechanism for the Honey Templates generation that learns and improves the templates indistinguishability property?
- **RQ3.** If face biometric templates, sugar or honey, were reconstructed and their images were tested to be distinguished by automatic classifiers or human testers, who would be more accurate?
- **RQ4.** Can we assert that Honey Templates are applicable in real-life scenarios?
- **RQ5*.** Can the mechanisms used in this thesis on biomtric templates be extended to other access control mechanisms: such as passwords, from where it emerged?

For the verification of the research questions, our main research methodology were experiments, where we had first to design our algorithms, prove the accuracy of the new mechanisms, and test system performances. The results of different algorithms, but which fell under the same category, were then compared. Other methodologies include literature review, and in one of the Research Questions (RQ4) we performed a small-sample survey. Research Question 5 is labelled with (*) because the answer for their verification is given in Appendix A at the end of this thesis. The reason of this decision lies in the fact that this questions might serve as future research work, requires more academic pursuit, their exploration requires more interdisciplinarity within the field, which would go beyond the aims of this work. Nevertheless we have started working in this direction too, contributing with several papers.

1.4 Contribution of this thesis

The main contributions of this thesis are as follows:
1. A new protection mechanism for Biometric Template Protection.
Honey objects are not novel in the research domain and in information security specifically. There have been different implementations: honeywords, honeypots, honeyfiles, etc, but biometric honey templates is a completely new implementation in a biometric system scenario. In our endevour we could notice that every honey object implementation has its own specifics. It resulted that biometric honey templates should be designed in such a way that a possible attacker can not distinguish

them from real ones, and, on the other side, these templates should be different enough not to be considered as acceptable by a verification module. This was one of the challenges we faced.
2. System Architecture designs.
In this thesis we designed a biometric system architecure augmented with honey templates. Except from offering protection, this mechanism has information leakage capabilities.
3. Introduction of new metrics.
To measure how synthetic templates are designed and how they could be acceptable to be included in a biometric database, we introduced the new "indistinguishability" metrics. Together with the metric we defined all the new concepts necessary to complete the whole methodology, together with a set of definitions, algorithms, and theorem.
4. Machine Learning mechanism for template generation.
An important contribution to our thesis is the "iterative classifier", a tool that by means of Machine Learning tools is continuously learning and generating improved honey templates in terms of indistinguishability. This tool was used in our other research regarding honey-passwords, included in this thesis.
5. More than six protection schemes are tested in combination with honey templates.
Finally, together with the qualitative generation of biometric synthetic templates by means of the iterative classifier, we added a second security layer to the system by combining honey templates with a protection biometric scheme. There are six algorithms that are experimented in this work: (I) a generic feature transformation proof-of-concept algorithm; (II) PCA for faces; (III) Bloom Filters (IV) Fuzzy Commitment; (V) a hybrid augmented protection scheme; and (VI) an algorithm for the measurement of the attacker's confidence in using honey templates. All algorithms are tested following the same protocol, then the system performance and security are evaluated. In addition, in Appendix A, we have implemented PassGAN for passwords for synthetic password generation via an interative classifier (Algorithm VII).

1.5 Structure of the Dissertation

In this chapter we offered to the reader a generic overview on what the main idea of this work is, our motivation, a short but detailed description, main research questions and contributions. Following, the thesis is organized in seven more chapters for RQ1 to RQ4, a separate chapter dedicated to thesis conclusions, and one appendix for RQ5.

In **Chapter 2** we give an overview of what honey objects are, the cryptographic apparatus necessary to understand the *property of indistinguishability* and *distinguishing attacks*. From the literature review, we could understand that the Indistinguishability property is a very important concept in modern cryptographic security. The simple case of distinguishing processes is on two variables (probability distributions), but it can be generalized for systems. We could further explore in this chapter what we considered necessary to lay a stable basis in terms of cryptography, for a clear and sound understanding of the new concepts and metrics that we need to introduce/adopt in a biometric setting. The chapter starts explaining in detail the concept of honey objects in the context of information security in general. Different mechanisms are reviewed, yielding the way to a better understanding of the database augmentation with synthetic data in order to lead attackers astray. The generic Honey Objects database architecture design is given and the specific honey templates system is described. This architecture is followed in all the template protection mechanisms contributed in this thesis.

In **Chapter 3** we define what biometric system and templates are, and explore the existing protection mechanisms. A focus is given to the generic biometric system threats and vulnerabilities, together with biometric template protection criteria. We can understand in this chapter the importance of robust protection schemes and the fact that nowadays two - or more - security levels are needed in information systems containing, processing and storing sensitive data.

To have a quick proof of the possibility of the implementation of honey templates in a biometric system, we here provide the first implementation of this idea. The authentication system is based on face recognition, and is the augmented with honey templates, (***Algorithm I***). The chapter ends with all the necessary recognition accuracy results by providing and comparing error rates in different setups, before and after protection and honey templates addition. This chapter helps us give hope for our first research question whether honey templates are applicable in biometric systems and how they affect the system performance. Anyway, to verify RQ1, there is still a lot more to experiment. To have the answer to this research question, in chapters 4, 5, and 6, Honey Templates are augmented in three different Biometric Template Protection Schemes: Random Projection, Bloom Filters, Fuzzy Commitment and hibrid scheme. We chose these mechanisms because they are each representatives from the main BTP categories: Biometric Cryptosystems, Feature Transformation, and Hybrid schemes. In Chapter 3, we answer RQ1.1, that honey templates influence the system performance.

Chapter 4 is dedicated to an important contribution we provide in our thesis: the iterative generation of templates by means of Machine Learning classification tools. In this approach, a trained classifier is used to test a real template, and conclude if it can be classified as a honey one (***Algorithm II***). If yes, the sugar template can be saved in the biometric database, otherwise a new template is generated for being tested by the classifier again.

In this chapter different classification tools are examined, and an inclusive multi-classification tool architecture is provided for biometric templates indistinguishability. Results from this work help us answer to our research question RQ1 (can a BTP scheme be augmented with Honey Templates), and to RQ1.5, as it is possible for an attacker to mimic a honey-based system in terms of its parameters, and use it to classify leaked templates. The level of confidence of the attacker in the classification framework depends on the training sets, and algorithms. Another contribution in this Chapter are the definitions that add to the metrics apparatus regarding honey templates, making the answer to the research question RQ1.3 more complete.

In ***Chapter 5*** we implement ***Algorithm III*** which merges an existing and robust template protection scheme - Bloom Filters, with the mechanism of honey templates. In this scheme the recognition performances are improved compared to previous ones, contributing to our research question RQ1 regarding the implementation to honey templates. In RQ1.1, Honey Templates do not deteriorate the system performance when the template protection mechanism are Bloom Filters. For RQ1.3, regarding honey templates metrics, with the definitions provided in this chapter we complete and verify this research question.

Algorithm IV in ***Chapter 6*** merges a hybrid scheme with honey templates. Both algorithms (III and IV) follow again the same protocol, they are all based on face authentication and in all chapters we can find the necessary metrics for template indistinguishability and system performance. This chapter completes research question RQ1, as we have now applied honey templates in different template protection schemes. Regarding RQ1.4, we observe that there is an upper limit in the number of Honey Templates that can be augmented in this scheme. In this chapter we provide the answer to RQ3, regarding who would be more accurate between automatic classifiers or human testers, if face biometric templates -real or honey- were reconstructed.

In ***Algorithm V*** described in ***Chapter 7***, we provide the possibility of pre-image classification for improved security and add another contribution: to test the indistinguishability of templates we compared automatic classifiers with human testers. The resulting rates might yield us to an interesting new research area, *the psychological perception of human visual and memory to detect minor changes of reconstructed pre-images*. This is one of the directions we can aim in the future.

Even if we assume that all the mechanisms we provided to this point are ideally generating honey templates with perfect indistinguishability rates, there is another important security analysis we have to provide. Supposing a high-level attacker having full knowledge on the system, we propose in ***Chapter 8*** a methodology for the theoretical evaluation of template predictability based on Bayesian inference (***Algorithm 6***).

We simulate a honey-based system generating and storing K synthetic templates for each subject. By following the same protocol of the previous chapters, a trained classifier is used to test a real template, to see if it can be classified as a honey one. If yes, this template will be saved in the biometric database, otherwise a new template is generated to be tested again. Templates testing and storing will be even controlled by the classifier by this condition: sugar templates are classified and stored as honey templates to a certain percentage q (i.e., the iteratively generated sugar templates can be all considered as honey, or only a portion q of them). We suppose the attacker knows even this percentage.

Finally, in ***Appendix A***, we have included one of our recent research works that stemmed from the different disciplines and topics like Honey Templates, Machine Learning, iterative classification, access controls, defender/attacker game, etc, that we have included in this monography. It is about the generation of synthetic passwords as according to the PassGAN algorithm with the adoption of the itterative classifier used in a honey templates scenario.

The final content, ***Chapter 9*** is one of the most important ones of this work. It summarizes, compares and analyses all the results from Chapters 1 through 8, by giving a full and clear overview of the contributions. It also adds more clarification, with more data, for every Research Question we proposed in the previous section. Finally, the chapter concludes with new ideas that we intend to further develop in the future, or that can serve as starting points and interesting topics for young researchers, but not only, that find the idea interesting and important. We hope these ideas will contribute to the advance of new knowledge in this field, and the practical improvement of secure access control mechanisms.

Chapter 2

Honey Objects in System Security: the property of Indistinguishability

Being in an unideal setup, information systems suffer from different threats. Adversaries continuously try to posses database contents, break communication channels, even impersonate users. To this end, protection mechanisms will be too prone to different attacks, and if adversaries are successful the protection mechanism is considered broken. One of the mechanisms that adds a security layer to systems are *honey objects*, which mimic the appearance and behaviour of other sensitive entities of an information system: servers, files, data, etc. Fact is that an adversary is sometimes contended even with less achievement, but if they are faced with these different entities (real and fake) they have to not only break their protection mechanism, but also distinguish, process, classify, or even guess which the real data is among them all.

2.1 Introduction

Honey objects are used in various aspects of system security to deceive internal threats or external intruders (be they people or machines) against unauthorized data access. An example are *honeypots* mostly used for the detection of outer intruders, which are network machines used to distract adversaries from other more important machines, and *honey farms* (a network of honeypots) (47) enabling deep research into server-side attacks. Subsequently, *honeytokens* (100) are mostly implemented against internal threats and another development we find at system level are *honeyclients* (80), the complementary of honeypots, designed to mimic the behavior of a web browser.

On the data level we find solutions such as the *honeywords* and *honeydocuments*. The honeyword method (58) hides the password of a user between k hash values of random passwords, and honeydocuments (15) is again a trap-based mechanism which uses decoy documents. All these mechanisms serve as a safeguard against adversaries who try to get unauthorized data access.

2.2 A generic honey-objects system design

The design of a honey objects database is shown in Fig. 2.1 and it can be applied to passwords, biometric templates and other objects which have similar properties with them, in terms of usage and storage. As in (114) for the i–th user, during ***registration***, the sugar object S_i will be created from the user data and *K-1* honey objects H_{ij} $(j = 1, 2, ..., K-1)$ will be generated. These objects will pass through a protection mechanism (like hashing in passwords or one of the BTPS in biometric systems) having as a result a set of K protected objects: one SP_i from the sugar object and *K-1* protected honey objects HP_{ij} $(j = 1, 2, ..., K-1)$.

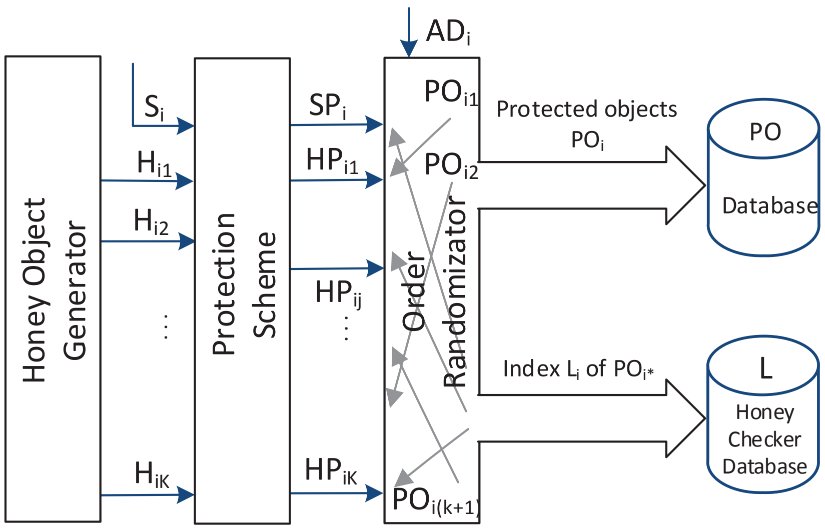


Figure 2.1: A generalized Honey Objects database architecture design.

To hide SP_i among the other honey objects, its memory address or index is randomly allocated and to the other objects are assigned the remaining addresses or indices. This process is handled by the *Order Randomization block*, which uses the auxiliary data AD_i to generate the SP_i-s index and the indices of the protected honey objects. We define the protected objects as PI_{ij} with $(j = 1, 2, ..., K)$ and the set $PO_i = \{AD_i, PI_{i1}, PI_{i2}, ..., PI_{iK}\}$ of user i, containing the auxiliary data and the randomized protected objects. This set is stored in the database whereas the index L_i is stored in the *Honey Checker Database*.

A generic honey objects architecture design is presented in Fig. 2.1. This architecture is generalized for every system augmented with honey objects applied at data level (i.e. it excludes system solutions like honeypots, or their networks, but is applicable on files, passwords, templates, etc). It comprises of three main modules: *i)* honey objects generator; *ii)* protection scheme module; and *iii)* order randomization module. First, a plain object S_i is created, which is a representation of a genuine subject (e.g. a password, a template, a record, a file, etc). From the first module a set of k-honey objects H_{ij}, with $j = 1, 2, ..., k$ is generated. On S_i and H_{ij} is applied a protection mechanism, then their order randomized, and a final set of (k+1)-protected objects $PO_i = \{S_i, H_{i1}, H_{i2}, ..., H_{ik}\}$, named *sweet objects*, is stored in the database. The index of the genuine object after order randomization only will be stored separately in the *Honey Checker Database*.

In this scheme, the *indistinguishability* property of honey objects, has to be assured by the first block, the Honey Object Generator, and the Order Randomization block handles the fulfilment of the second property, *secrecy*. It uses the auxiliary data AD_i to shuffle the $(k+1)$-indices of the objects by using a pseudo-random number generator ($PRNG$) with a shuffling algorithm (e.g. Fisher-Yates (10)). If, as supposed in the architecture design of the honey objects database, the sugar object is generated first, and the honey objects are positioned in the remaining k-indices, then the randomization block technique for index shuffling should not reveal this input sequence. So, if an attacker possesses the list of objects, it should be hard for him to reconstruct the original sequence. In fact this is an important requirement of cryptographic shuffling algorithms and a solution to this can be found in (27).

2.3 Honeywords

The Honeywords project, is presented by Juels and Rivest (58) in 2013. It is a method used to improve the security of hashed passwords, by adding false passwords for each user. The aim of honeywords was to generate passwords as much look-alike as human generated passwords. This method faced a first challenge: false passwords should match their user's profile (vocabulary words, important dates, gender, etc.), otherwise a knowledgeable attacker might be able to differentiate real passwords from machine generated passwords. The real password is called a *sugarword* while the decoys are called *honeywords*. The combination of the two is often referred to as *sweetwords*. When the attacker succeeds in cracking passwords from the leaked shadow password file, she or he still has to be able to choose which password is the genuine one out of all the cracked sweetwords. But, once the attacker logs in using a honeyword, an alarm will be triggered to notify the defender that there is a malicious password attempt to the system and high confidence that the password file was leaked from the system and cracked.

Generally, the system works as follows. During the registration process, the user will choose a username and a password. Following that, $k - 1$ honeywords associated with the password will be generated so that it has k sweetwords. The sweetwords, together with the username and the other information about the user, are then stored in the login server after the order of the passwords is shuffled. Subsequently, the information about the user index and the index of the real password will be saved into a checker-sever called honeychecker. Since the honeychecker only has information about user index and real password index, the honeychecker can be used only if we have access to user information in the login server. In this system, it is assumed that the attacker does not manage to steal both the login server and the honeychecker data in the same time period.

During the login phase, the user enters his username and password. The system then looks for the matching record in the login server. If the username exists, after performing the hashing calculation to the entered password, the system then checks whether the password is in the sweetwords stored for the corresponding user. If the password is not found in the sweetwords, then the password entered is incorrect and login is denied. Otherwise, if the password is found, the system sends information about the user index and index of the entered password into the honeychecker. Honeychecker then makes a comparison between the entered password index and the real password index for the corresponding user in the database. If the two have the same index, then the login is successful. On the contrary, the honeychecker sends an alert to the system administrator about a password data breach or does other procedures in accordance with the policy that has been determined.

2.4 The property of Indistinguishability in a cryptographic context

From the literature, and in the given honey objects architecturene, we can state that one of the main threats to honey objects implementations are *Distinguishing attacks*. They are a form of cryptanalysis which allow an attacker to differentiate encrypted data from random data, i.e. in certain situations honey objects from real objects, for example, codewords from random binary strings. In this case, a ciphertext is considered secure, if it cannot be distinguished by a random permutation having the same domain and range as the cipher, -this is mainly the case in block ciphers. Indistinguishability is a very important concept in modern cryptographic security. The simple case of distinguishing processes is on two variables (probability distributions), but it can be generalized for systems.

In this section we provide the cryptographic apparatus that is neccessary for our work in building a robust system in terms of security. It will also help us in shaping the measuring mechanism and evaluating the indistinguishability property of honey templates. Based on what literature has proposed for variables, distributions, or ciphers, we will adopt these concepts and provide our ideas. In the following definitions, for our purposes, *functions* will be interpreted as *template protection schemes*, *messages* as *plain feature vectors*, and *codewords* or ciphertexts as *protected templates*. The first definition is that of a *cryptosystem*.

Definition 1 (Cryptosystem) The cryptosystem C is comprised of a set $\{E_e : e \in K\}$ of enciphering algorithms, and a corresponding set $\{E_e^{-1} : e \in K\} = \{D_d : d \in K\}$ of deciphering algorithms. □

Let's recall from Cryptography: the pair (e, d) are the keys used for *encryption* (e) and *decryption* (d). For each $e \in K$ there exists a $d \in K$ such that $D_d(E_e(m)) = m$. Messages m are called *plaintexts*. The output of the encipherment are called *ciphertexts* (76). In case the same key is used for encryption and decription ($e = d$), the cryptosystem is called *symmetric*. Otherwise, we have *asymmetric* cryptosystems. In this case, there is no need to share a key, but d can be deducted by e. Symmetric cryptosystems are considered *private-key*, whereas the later are *public-key* (74).

In symmetric cryptosystems the two parties sharing the same secret key want to exchange messages over an insecure channel, i.e. an adversary can see all sent messages but cannot modify them, or insert other messages. According the Kerchkhoff' principle an adversary has full knowledge on the system (encryption and decryption algorithms) except the shared key. Recalling from Definition 1 a cryptosystem is the set of Encryption and Decryption algorithms (*E* and *D*). To generate the shared key a third algorithm is added: *Gen*. It outputs a key *k* from the predefined finite set *K*. Formally, the encrytion algorithm $E_k(m) = c$ takes as input a key $k \in K$ and a message $m \in M$ and its output is a ciphertext $c \in C$. Similarly, $D_k(c) = m$ has as inputs the same key $k \in K$, and ciphertext $c \in C$. The output is the same message $m \in M$.

In real life scenarios, regardless of the *a priori* analysis, there will always exist the necessity to construct a model for the evaluation of a system security, that will strongly depend on the cryptographic apparatus. In a honey-objects setup, this becomes of even more utmost importance. Theoretically, the main models are for evaluating system security are: *unconditional security*; *provable security*; *computational security*; and *complexity-theoretic security*.

Unconditional security in the context of encryption is called *perfect secrecy*. It is assumed that an attacker has unlimited computational power. From the definition of perfect secrecy, the encryption of two messages m_a and m_b from M are identically distributed regardless of the key k from K.

Definition 2 (Perfect secrecy) A symmetric cryptosystem with message space *M*, key space *K*, and ciphertext space *C* is *perfectly secret* if for all message pairs $m_a, m_b \in M$ and $c \in C$:

$$P[E_k(m_a) = c] = P[E_k(m_b) = c] \tag{2.1}$$ □

The assumption made on the attacker's resources makes unconditional security unpractical, since at least computing power or space are limited in a real-life scenario. The same holds true for honey objects. If we consider that *Practical security* is the so-called *provable security*. In this case the potential of breaking a given cryptosystem is compared to well-known difficult problems. It is shown that a successful attack is not possible in systems with provable security.

Computational security is based on known attacks and measured on number of operations *n*. Most systems belong to this class. A successful attacker can break a securely known system, or solve an unproved mathematical problem. Proofs in computationally secure systems have to be refined more concretely, e.g. key size in terms of attacker's resources and capability. To better describe computational security we define here *key protection* and *message protection*, supposing an algorithm *T* can run in time $t^T \leq t^E$ of the encryption algorithm (*E*). *T* is also called a *statistical test.*

Definition 3 (Key protection) A scheme is secure if the probability:

$$P[T(M, C) = K] \leq \varepsilon \tag{2.2}$$ □

From this definition we assume that there exists a very small probability which allows a key to be recovered from a given message and its cyphertext. Also, it does not provide any protection for the message.

Definition 4 (Message protection) A scheme is secure if the probability:

$$P[T(E_K(M)) = M] \leq \varepsilon \tag{2.3}$$ □

These definitions seems to protect the message from its recovery, and are helpful in defining the scheme security. And finally, *information-theoretic security* considers an attacker has unlimited computing power, but the security does not rely on unproven assumptions. This means that security will be kept even with the development of new machines and methods. The first such system is the *One-Time pad* (OTP). In our honey-based system, we can say that the difficulty of the attacker to crack the algorithm resorts to probability (i.e. *information-theoretic security*) instead of computational complexity based security.

2.5 Indistinguishability as a security concept

Indistinguishability, as we mentioned in the previous section, is a very important concept in modern cryptographic security. To better understand this property in a cryptographic context, let's suppose we have a message space with two elements only. Given an encryption E of a message m, by definition:

Definition 5 (Indistinguishability-secure cryptosystem) A cryptosystem C is considered secure in terms of indistinguishability if any adversary cannot distinguish the chosen ciphertext with probability greater than 0.5. □

This means that the adversary should not gain any information from the ciphertext, but guessing randomly, or flipping a coin. Indistinguishability attacks are considered as an important security property of many encryption schemes, and sometimes as a requirement. Informally speaking, the indistinguishability process in security terms is presented as a game. If the attacker's probability in guessing ciphertexts is approximate to 0.5 then the cryptosystem wins, otherwise it's the attacker [1]. To this point, we can consider other data representations, e.g. starting from *synthetic data* generated to be as much look-alike as real ones, to *honeywords* and *honey templates*, which are put in the same comparison with their respective real one. Becoming part of the same game, and potential victims of the same indistinguishability attacks, we can apply the same definitions and theorems we will further describe in this section. We can now define what an indistinguishability secure honey-based system is:

Definition 6 (Indistinguishability-secure honey-based system) A honey-based system H is considered secure in terms of indistinguishability if any adversary cannot distinguish the chosen honey-object with probability greater than 0.5. □

[1]Definitions and theorems in this section result from lectures in Cryptography and related topics: (I) Theoretical Foundations of Cryptography, by Chris Peikert; (II) Cryptography by David Wagner; (III) Cryptography, by Paul Beame.

Theory further defines, in a more relaxed manner, what Statistically ε-indistinguishability of random variables is:

Definition 7 (Statistically ε-indistinguishability) Let X and Y be random variables taking values in a set *S*. X and Y are called statistically ε-indistinguishable if for every event $T \subseteq S$

$$|P(X \in T) - P(Y \in T)| \leqslant \varepsilon \tag{2.4} \quad \square$$

If $\varepsilon = 0$ the two probabilities are equal. This is known as *perfect indistinguishability*. The left side of Eq. 2.4 is the advantage of the test T, Adv*T*. This can be considered as a game of two black boxes. One black box outputs $x \leftarrow D$, the other $x \leftarrow D'$). The advantage of guessing which box you were given measures the difference with the ideal case of *0.5*.This can be the guessing game that an attacker tries if he/she possesses real and fake data in the same set. If no other means can discriminate the two type of values, then guessing is the final ace under the attacker's sleeve. In the world of honeywords, or honey templates, an attacker might own classification tools, or could adopt other techniques to separate real data from synthetically generated passwords or biometric templates. Regardless of their classification method, the mentioned honey objects should have been generated with strong indistinguishability property, such that the attacker's probability is infinitesimally small. Let's further introduce two more definitions: statistical secrecy, and computational indistinguishability.

Definition 8 (Statistical secrecy) The encryption scheme satisfies the *statistical ε-indistinguishability*, if for every two $m_1, m_2 \in M$, the random variables $E_K(m_1)$ and $E_K(m_2)$ are statistically ε-indistinguishable.

Intuitively, the probability for the adversary to get information about the plaintext *m* in at most ε.

Definition 9 (Computational indistinguishability) Distributions D and D' are computationally indistinguishable (or (t, ε) *indistinguishable*) if for every strategy *T*, it takes no longer time *t*, then the advantage ε.

$$AdvT \leq \varepsilon \tag{2.5} \quad \square$$

Distributions *D* and *D'* are computationally indistinguishable if no efficient algorithm can tell the difference between them except with small probability.

These definitions help in defining the following theorems.

Theorem 1 (Theorem of Indistinguishability) *X is indistinguishable from Y (written $X \sim Y$) if X and Y are (t, ε)-indistinguishable for t and ε being infinitesimally small.* □

The security definitions we provided measure the indistinguishability property in a cryptographic domain, between a ciphertext and an enciphered random plaintext. This is considered as *real-or-random (rr) security*. In a biometric templates scenario, there are different protection schemes which are based on cryptographic primitives, meaning that when augmenting these systems with synthetic templates we will adopt the same concepts, definitions, and theorems to measure their indistinguishability property. Except from real-or-random security, there are some more definitions of security under the *indistinguishability* approach (24).

Find-then-guess security

In the algorithm of find-then-guess (*fg*) the attacker acts in two phases. First he makes queries to the oracle with two messages (find). Then, he picks one of the messages at random, encrypts it, and then makes more oracle queries in order to output a guess. The adversary A in find-then-guess security is a two-phase algorithm. It should be noted that if the probability of guessing the right one is 0.5 this does not mean that every pair of encrypted messages is indistinguishable, but at least those the adversary comes up with.

Left-or-right security

Let us now suppose the adversary has two selector functions S_1 and S_2. They take two arguments (m, m_0) and each return the first or the second: $S_1(m, m_0) = m$ and $S_2(m, m_0) = m_0$. These two functions feed two oracles $E(k, S_1)$ and $E(k, S_2)$. One of the oracles is given to the attacker, but without knowing which. The oracle may then make a number of queries, trying to understand if the result came from the *right* or from the *left* part. This approach is known as *left-or-right security* (*lr*) and can be considered as a generalization of *find-then-guess* security.

It is proven that left-or-right security implies real-or-random security ($lr \Rightarrow rr$). Even the contrary should be considered: *rr insecurity implies lr insecurity*, i.e. real-or-random attacks can be transformed in left-or-right attacks.

Randomness of ciphertexts security

In the case of real-or-random security the plaintext was randomized then encrytpted, whereas now the ciphertext is randomized. This is the security requirement of *randomness of ciphertexts* (*rc*). It measures the indistinguishability of a ciphertext from a random string having the same length.

Semantic security

Semantic security (*ss*) implies that an attacker may send a distribution over messages, and not merely one message to the oracle. This can be considered a highly knowledgeable attacker who owns a function that encapsulates the partial information the attacker already has, a sequence of samples ($m_1, ..., m_q$) from *M*.

These security definitions will help us in modelling different attacker profiles for the honey templates schemes.

2.6 Brief chapter summary

In this chapter we provided an overview of honey objects and their application, offering a generalized architecture, applicable at data level (files, passwords, templates, etc). The system was explained for both modes: registration, and operation. The main property of this architecture is that honey objects should not be indistinguishable from real ones. We review this property under a cryptographic context, but drawing parallel lines between cryptosystems and honey-based systems. In fact, honey objects must comply to two main properties: (1) **indistinguishability**, honey and real objects must be hard to distinguish from each other (e.g. a real password from a generated password or a database entry of a real patient in a health system from a fake one); (2) **secrecy**, the real object must be secret and camouflaged among the honey objects. In the case of honeywords, if an intruder by some means gets access to the user' set of passwords, he can use/guess only one of them. The system intercepts that a honeyword is used, it will consider this as information leakage and proceed with further steps (set off an alarm and/or update the passwords set).

This is a basis for the coming chapter, where biometric systems are discussed, their protections mechanisms, and there is given the first proof-of-concept of a honey templates biometric system.

Regarding our Research Questions, with this chapter we set the basis for the verification of **RQ1.3**, if the indistinguishability property can be measured with reliable metrics. To our work, generating honey templates that are indistinguishable from real ones, means that the probability of guessing a real template from a synthetic one is equal to 0.5. Or, it means that classification tools are not able to differentiate them. So, if in other classifications in general researchers tend to have clear class separations, in this work classifiers should ideally provide a value close to 0.5. These metrics will be presented in *Chapter 4*.

Chapter 3

Honey Objects as a Biometric Template Protection mechanism

Biometric systems can be found nowadays implemented in a variety of situations, for the purpose to recognize individuals based on their characteristics. Either physiological (fingerprints, face, iris, etc.) or behavioural (voice, gait recognition, keyboard dynamics, etc.) these characteristics can be detected, and biometric features extracted to be further processed in order to be used for comparison. Biometric features are numerical representations of these traits. They are stored in the form of *templates* in the biometric database, which can help individuals enrol, identify or verify, and after a successful recognition process let them gain access in other specific parts of an information system. In this chapter we will provide a state of art and basis for the application of honey templates in biometric systems.

3.1 Introduction to biometric systems: threats and vulnerabilities

In a biometric system, during registration, verification or identification, the subject can present his/her biometric characteristic (face, fingerprint, signature, iris, voice, etc.) and the system extracts a set of features from this content. These features, used for comparison to recognize or identify a subject, are protected and stored in a data storage system (such as a database) in the form of a template. By definition, *a template is a set of stored biometric features comparable directly to probe biometric features* (5). It represents sensitive information and should be protected without losing the capability to identify or verify a person (97). This is important since templates, if not properly protected, can reveal much of the subject's information.

As every Information System, even biometric systems can be prone to attacks. An adversary who has somehow stolen the biometric database content, can break template's protection mechanism, generate spoof biometrics and impersonate a user. In fact, in September 2015 a severe biometric breach happened to the Office of Personel Management in Washington after a cyberattack, where 5,6 million fingerprints were stolen by a group of hackers[1]. Federal experts claimed that misuse of stolen fingerprints was limited, and intelligence community proposed they would find ways to prevent fingerprint data abuse.

In late March 2016, the largest government data breach to date happened, where 55 million voters in the Philippines were affected. Data comprised of 228,605 email addresses, and 15,8 million fingerprints[2]. Again, security experts told biometric data might be useless without a computer system that can interpret it, since these type of systems are developed for specific countries and situations. But, it can be assumed that in the future attackers might develop tools and systems such that their ability to break template protection mechanisms and impersonate users enhances significantly.

Biometric Authentication Systems nowadays use fingerprints, faces, signatures, iris and other biometric characteristic types of individuals in order to identify them. The whole process, from data capture subsystem to decision subsystem, represents a "chain" of building-blocks. In every link of this chain, one of the main concerns is information protection [16]. The main components of a biometric system in general include: the data capture module; the feature extraction module; the comparison module; the storage module; and the decision module [8], [16]. A biometrics-enabled identification system can perform these actions on a biometric subject: enrollment, verification, identification.

Using biometrics as a means to verify or identify a subject has also raised a lot of concerns mainly because an adversary attack can impose real threats to the overall system. Jain et al. consider three main attacks (52) as part of the biometric system, which are considered as global vulnerabilities:

1. **administration attack**: these attacks are present as a result of improper administration from the inside.

[1]Washington Post, "OPM says 5.6 million fingerprints stolen in cyberattack, five times as many as previously thought", *https://www.washingtonpost.com/news/the-switch/wp/2015/09/23/opm-now-says-more-than-five-million-fingerprints-compromised-in-breaches/*

[2]Wired, "The Philippines election hack is 'freaking huge'", *http://www.wired.co.uk/news/archive/2016-04/14/philippines-data-breach-fingerprint-data*

2. **non-secure infrastructure**: these attacks can be found in hardware, software or communication channels of the system.

3. **biometric overtness**: an attacker can circumvent the system by presenting a fake artefact of a biometric characteristic.

Other biometric systems threats, attacks, and vulnerabilities are described in (17, 79, 48) and are presented in Table 3.1. We briefly describe these threats.

In **T1**, an attacker presents a fake biometric characteristic at the *Reader*. The fabrication of fake characteristics is called *Synthetic Biometric Feature Attack*. It can be applied on different biometrics: fingerprints, iris, face, voice, but some are more resistant than others.

Threat **T2** involves the continuous presentation of biometric characteristics previously recorded to perturb the *Comparator* (biometric system module which compares a reference stored in the database and a newly presented one) score (output of the *Comparator*) until the threshold is reached. A *Trojan Horse* is a program that acts on entities by disguising itself and operating to the attacker interest. This attack in general is executed against biometric system modules such as Reader, Feature extractor, Comparator, *Decision module* (outputs a decision, based on the Comparator score), etc.

In **T3** the Trojan Horse can generate a set of features and a template based on them, in **T5** the attacker can control and send commands to the Trojan Horse to do other operations (e.g. changing the threshold), whereas in **T7** the output of the Decision Module is overridden. The authentication process is bypassed in this case. The output of this block could be forced to produce a YES/NO answer.

T4, **T6** and **T8** threats happen on the communication channel between two building blocks. These attacks are not possible if the two blocks (e.g. Feature extractor and Comparator) reside on the same machine.

Finally, in **T9** the attacker will try to compromise the security of the database. Different attempts could be: double enroll attack of a user with a different name in the database with different privileges, corrupted template injection, DoS to the subject associated with the corrupted template, etc.

3.2 Biometric templates protection mechanisms

In a generic biometric system, according to ISO/IEC 24745 standard (49), biometric references (samples or templates) contain mainly two parts: **(1)** ***a pseudonymous identifier*** (*PI*), it represents an individual as a protected identity, e.g. a

Table 3.1: Threats of a biometric system

No.	Threat	Type of attack
T1	*Fake biometrics.*	Synthetic Biometric Feature Attack (13).
T2	*Resubmission of a previous biometric signal.*	Replay attack and Hill Climbing attack (106, 71).
T3	*Feature extractor threat.*	Trojan Horse: bypasses the feature extractor sub-system (4).
T4	*Communication channel between Feature Extractor and Comparator.*	Channel attack: An attacker intercepts a subject's template and injects a malicious template to attack the system (67).
T5	*Comparator attack.*	Trojan Horse: forces the comparator to produce a higher or lower score.(67)
T6	*Communication channel attack between Comparator and Decision Module.*	Channel attack: attacker accesses the communication channel and injects a score (70).
T7	*Decision Module attack.*	Trojan Horse: forces the Decision Module to produce a YES/NO result. (4)
T8	*Communication channel attack between Storage and Comparison Module*	Channel attack: intercepts the communication between the two and changes, deletes or replays data (70).
T9	*Storage subsystem attack.*	Unauthorized modification of one or more templates (66).

transformed template; and **(2)** ***auxiliary data*** (*AD*), used to reconstruct pseudonymous identifiers during verification, e.g. transformation parameters or keys. The pseudonymous identifier is the element used for verification and it should not allow the retrieval of the original biometric features. Whereas the auxiliary data generated during enrolment, are not compared during verification.

From a security perspective, the protection of templates stored in the database of a biometric system is one of the main challenges. An adversary can try to carry out a modification of their contents or even an unauthorized transfer of templates from the database towards another system. He can then generate a pre-image of the template by hill-climbing or brute force attack and since the adversary can create a fake physical characteristic from the biometric template this leads to physical masquerade attacks. All these attacks may occur as in a biometric standalone system, such as an automated border control system, as well as in remote biometric authentication. To prevent the leakage of biometric information, in addition to a better control of database access, other techniques should be implemented to prevent attacks and, even better, warn if such a leak has occurred.

3.2.1 Privacy risks and standard security requirements

Using biometrics as a means to verify or identify a person has also raised a lot of concerns mainly because there exists a tight coupling between a method and the physical properties of a subject. If these data are not secure then an intruder can possess them and try to use them to his/her interest, risking the privacy (17). An adversary attack can be included into different categories. For example Jain et al. consider three main attacks: **administration attack**, **non-secure administration** and **biometric overtness** (51), as part of the biometric system vulnerability. Also Breebaart et al. divide privacy risk into four categories (17):

1. Unauthorized collection of biometric samples without the subject's knowledge
2. Unnecessary collection
3. Unauthorized use and disclosure (like forensic, cross-matching databases, individual monitoring, etc)
4. Function creep, where the system is expanded into areas not originally intended.

Other vulnerabilities mentioned in (77) include:

1. Replacement by an impostor

2. Creation of a physical spoof
3. Replay of the template at the comparator module
4. Cross-matching across different databases to track a person without his/her consent.
5. An attacker tries to modify one or more templates in the database
6. Channel attack between stored templates and the comparator.
7. A hacker can override the choice of result and even if the actual pattern
8. recognition system had excellent performance characteristics, it has been rendered useless.

A protected template must be such that:

- it is impossible to retrieve the original sample or features;
- it is impossible to link genuine subjects across different services or databases; and
- it is still useful for comparison;

When templates pass through communication channels between the different building blocks of a biometric system, or when they are stored in the database, an attacker, a malicious intruder, or even system faults might be the cause of system compromise. To this respect there are three security criteria for biometric systems in the ISO standard (49):

a) *Confidentiality*: protects templates against unauthorized access when stored or transmitted. To obtain confidentiality on biometric data different encryption algorithms can be applied as access control mechanisms.

b) *Integrity*: assures accuracy and trust of biometric templates. If the biometric reference is untrustworthy because of accidental corruption, or template modification by an attacker, so will be the authentication process, untrustworthy.

c) *Renewability*: revokes and renews compromised templates. Because of a database security breach an attacker may possess biometric templates, and try to impersonate an individual (e.g. spoofed fingerprints). In this case a new template should be generated and linked to that individual.

3.2.2 Categories of template protection schemes

Biometric data can be used to reliably authenticate individuals, but any leakage can pose severe security and privacy issues. The need thus raises to protect the privacy of the subjects not only with biometric template protection (BTP) approaches (92), either *biometric cryptosystems* (107) or *cancelable biometrics* (85), but more complementary solutions should be designed. In order to ensure the subject's privacy, the international standard ISO/IEC 24745 (49) on BTP establishes two main requirements for protected templates:

- *Irreversibility*: given a protected template, it should not be feasible for a computationally bound adversary to reconstruct a biometric signal which is positively authenticated by the system.
- *Unlinkability*: given two templates enrolled in different biometric systems, it should not be possible to determine whether they belong to the same subject.

Furthermore, other properties of the unprotected systems, such as authentication performance or speed, should be preserved (99).

The biometric community has directed serious efforts in the last decade to the development of efficient BTP schemes which fulfil those requirements. Among other examples, the fuzzy vault scheme (60), in which Error Correcting Codes and polynomial encoding are utilized to hide the underlying biometric data, has been applied to fingerprint (78) or face (109). Similarly, in the BioHashing approach (68) secret tokens are blended with biometric data to generate a distorted biometric template. The security provided can be further improved using random multispace quantization (104) or the application of irreversible transformations to the original biometric data (87).

There have been many efforts from the scientific community in the designing, implementing and testing of different template protection schemes. They can be categorized in two main groups:

1. Biometric cryptosystems: also referred to as helper data-based systems since some public information is stored in the storage module (51).

2. Feature transformation (or cancelable biometrics (92)): on the template is applied a transformation function and the result of this function is stored in the storage module

Both categories can be further divided in two more subcategories. Feature transformation schemes include: ***salting*** and non-invertible transform and biometric cryptosystem include: ***key binding*** and ***key generation***.

Key binding

The scheme is named as key binding since the helper data is obtained by binding it with a key which is independent of the biometric feature. This is then stored in the storage module. In general the helper data is an association of an ECC and the template. The main advantage of this scheme is that it is tolerant toward intra-user variations (87, 51, 23). The main approached to Key binding are: (a) *fuzzy commitment* (62), (b) *shielding functions* (93), and (c) *fuzzy vault* (64).

(a) *Fuzzy Commitment Scheme*

This technique combines Error Correcting Codes and cryptography to create a scheme. It uses a function F which is used to commit a codeword chosen from a set C. What is stored in the system is the helper data consisting of a hash value, $h(c)$, and a vector containing the difference between c and the vector of a subject x.

(b) *Shielding functions*

At enrollment the system takes a feature vector of fixed length. The helper data is generated by this vector and a key K. Also the hash value of the key is stored. There is also a function G that calculates the differences of each feature. In the authentication phase if the vector Y is presented, it is calculated the function $G(W, Y)$ where W is the vector of differences and if $h(K') = h(K)$, then Y is authentic.

(c) *Fuzzy Vault*

This is a very popular key binding scheme where a key K is locked in a vault, VA. The vault is created by polynomial interpolation where each symbol of the key can be a coefficient of the polynom. Additional points are also added to the vault and this set forms the template. During authentication of a subject, his/her set$B$ must overlap sufficiently with the stored set A.

Key generation

Differently from key binding in this scheme the helper data derives from dhe template and the key is generated from the helper data and the biometric features (101, 102). An advantage of it is the fact that the scheme can also be used for cryptographic applications but its main problem is the generation of the key because of the intra-user variability (86). The main approached to Key generation are (93):

a. **Private Template Scheme**: here the template serves as a secret key.

b. **Quantization Schemes:** under this category fall all those schemes where keys are obtained by quantization of biometric features.

Salting

In biometric salting the template is transformed according to a function F which is invertible and is defined by a key or password. Having to present a key during verification is an advantage since it has lower FAR. Also multiple templates can be generated (104). Approaches of Salting include *Bio-hashing*, and *non-invertible transform*. In this case the function F is a one-way function and as a result even if the key is possessed by an intruder, it is difficult to revoke the original template (86). This scheme is safer than salting but the transformation function is difficult to construct.

To overcome the aforementioned attacks, there have been many efforts from the scientific community in designing, implementing and testing different biometric template protection schemes. They can be categorized in two main groups:

1. **Biometric cryptosystems**: also referred to as helper data-based systems since some public information is stored in the storage module (52). They include two sub-categories: *key binding* and *key generation*. In *key binding* the helper data is obtained by binding it with a key which is independent of the biometric feature. In *key generation* the helper data derives from the template and the key is generated from the helper data and the biometric features (101). Representative mechanisms for both types of biometric cryptosystems include: fuzzy commitment (62), fuzzy vault (64), shielding functions (93); and quantization schemes (52), respectively.

2. **Feature transformation** (or cancelable biometrics): on the template is applied a transformation function and the result is stored in the storage module. This category of biometric template protection schemes includes: *salting* (where the template is transformed according to a function which is invertible (79)) and *non-invertible transform* (in this case the feature transformation function is a one-way function (86)).

Table 3.2: Advantages and limitations of BTP schemes

Scheme	Advantages	Limitations
Key binding	Tolerant to intra-user variations	a. Needs sophisticated comparators b. Don't offer diversity and revocability c. Helper data must be well designed
Key generation	Useful for cryptographic applications	Difficult to generate keys with high stability and entropy
Salting	a. Low False Accept Rate (FAR) b. Subject-specific key c. Easy to substitute if compromised	a. Not secure if the key is compromised b. Comparison happens in the transformation domain
Non-invertible transform	a. Higher security b. Application-specific transformation function c. Subject-specific transformation function	a. Needs to improve the ratio between discriminability and non-invertibility b. Transformation function is difficult to construct

These schemes have their advantages and limitations, which they are presented in Table 3.2.

3.2.3 Protection criteria

The main properties that the schemes must fulfill are: diversity with unlinkability (no cross-matching), revocability (easy to revoke a corrupted template), irreversibility (hard to obtain the original sample or features' information) and performance (no degradation of the biometric system) [21]. In the table above we present the main protection schemes, the advantages and limitations of each: Also in [17] we have a full view of the criteria of a protected biometric template. These criteria are grouped into three categories: (I) *technical performance*; (II) *biometric data protection*; and (III) *operational performance*. Below we list the main criteria definitions:

I. Accuracy: trustworthiness of the decisions made by a biometric system. Accuracy degradation: accuracy performance decrease caused by BTP algorithms Throughput: number of biometric transactions processed continuously by an individual processing unit Storage requirements: requirements imposed by biometric systems in different applications on the size of protected templates and BTP algorithms. Diversity: maximum number of independent protected templates that can be generated from the same biometric feature by a BTP algorithm.

II. Irreversibility (full-leakage): the difficulty of determining exactly or with tolerance margin from a protected template the biometric sample(s) or features used during enrolment to generate that template. Unlinkability: the difficulty of classifying the protected templates over time and across applications.

III. Characteristic types independence: the flexibility of dealing with different biometric characteristic types or data representations Interoperability: the degree to which standardized data interchange formats are supported by the BTP algorithm.

In hash based biometric template protection scheme, such as fuzzy commitment [8], and secure sketch [9], if the hash is cracked, then the adversary can estimate the pre-image of the biometric features. And for feature-transformation based BTPSs (in [10] and [11]), the masquerade attack is even more straightforward. This is because the protected templates, PTs, are compared directly with a distance threshold and the attacker can find a PT's pre-image (biometric feature) with normally less effort than the case finding a pre-image of a hash value. As a result, for every enrolled user in the database, we need to provide a protection mechanism which needs to be applied on all the sweet templates (sugar and honey).

3.3 The concept of honeywords in the biometric context

From a biometric perspective, there does not exists a similar honeywords challenge of cracked passwords' content. The generation of honey and real (*sugar*) templates can happen simultaneously, in a single-step process. This is not the case with passwords where the generation of a set of honeywords has to follow and adopt according to the real password of the user.

Biometric templates have another peculiarity compared to passwords. During comparison, when a user enters his/her password, it should match exactly with the one stored in the database to log in. In a biometric system, when a user newly presents the biometric characteristic the automated system comparison is based on statistical similarity tests only. This implies that in biometrics there should be more ways to generate honey templates as much look-alike as sugar templates.

For the reasons mentioned above we consider that the novice idea of honey templates is an approach with applicability benefits. It improves the three security criteria: *Confidentiality* (templates are protected against unauthorized access), *Integrity*assures trust of biometric templates, and *Renewability* (because of random process, sugar and honey templates are both updateable), but it needs to be further elaborated.

For the automatic recognition of individuals in a biometric system, characteristic features are stored in the form of *templates*. They contain sensitive information, and as a result strong protection mechanisms must be applied, preferably multi-level security schemes. One of these is the newly presented idea of *honey templates*. Except a protection algorithm (security level I), there are added synthetically generated templates in the same storage space with real ones to camouflage them (security level II). Their inclusion will make an impostor's attempts to impersonate a user harder and time consuming.

From a cryptanalytic perspective, the main requirement is *indistinguishability* between genuine and false (*honey*) templates. But, if an attacker has full knowledge on the system, in this work it will be shown that he/she is able to create a very similar one, risking this requirement. Will he be able to classify and distinguish templates with the system he built? To answer this we simulate a Defender/Attacker scenario implementing two face verification systems.

The main challenge of *honeywords* was to generate passwords as much look-alike as human generated passwords, but not only. Since many tests can be performed on passwords, another challenge to be solved was the fact that false passwords should match their user's profile (vocabulary words, important dates, gender, etc.), otherwise a knowledgeable attacker might be able to differentiate real passwords from machine generated passwords. The same challenge remains for honey templates too. An attacker should not be able to differentiate the two categories of templates. Even if he has full knowledge on the system (i.e. the fact that honey templates are deployed, algorithm parameters, classifier rates, etc.), and possesses sophisticated classification tools, his chances in distinguishing templates should be infinitesimally small.

Databases, having the role of storing the biometric templates, or other auxiliary data related to users (such as personal identifiers), they happen to be the target of most attackers. In fact, as it is the case of the majority of information systems, databases still suffer severe attacks. For this reason, synthetic biometric templates are added, hence the name *honey templates*.

3.3.1 Architecture design of a biometric honey templates system

In Fig. 5.1 we introduce the architecture design of a biometric system using honey templates. In the adoption of the *honeywords* concept, we have used and defined the following terms:

Definition 10 (Biometric Honey Templates) A set of indistinguishable and secret synthetic templates placed in the templates file of an authentication server, to deceive attackers. □

Definition 11 (Biometric Sugar Templates) A real biometric template, owned by a specific subject in the authentication biometric system, secretly placed among the honey templates in the subject's file in the authentication server. □

Definition 12 (Biometric Sweet Templates) A set of biometric templates comprised of the subject's real template and k-generated honey templates. □

During ***enrolment*** the *Application Server* converts the plain biometric feature B_i of user i to a set of protected templates. This set is defined in the same way as we did with the protected objects PO, i.e. $PT_i = \{AD_i, PI_{i1}, PI_{i2}, ..., PI_{iK}\}$. PTi will be stored in the *Biometric Database* and the index L_i in the *Honey Checker Database*.

During ***verification*** the user i will show his biometric characteristic and the Application Server will retrieve the $PI{*}_i$ from the Biometric Database and send it to the *Biometric Database Server*. After comparing $PI{*}_i$ with all the $PI_{ij}(j = 1, 2, ..., K)$ of user i, it will send the best-matched template's index to the Application Server. If the index idx_i matches L_i the user will grant access to the system, otherwise an alarm will be set off and specific rules will follow, according to the defined security policies of the system. In the latter case, if the user personal identifiers match but the templates' indices do not, the system will consider this attempt as information leakage.

The properties mentioned in the previous section for the general case, should apply in the case of biometric systems too, meaning that: honey templates should be *indistinguishable* from sugar templates generated from real users' biometric samples; they should *secretly* hide the sugar template; the fact that they share the same storage space means that they are conspicuous; and there should be no correlation between them and the plain features from where they were generated.

Except from being indistinguishable from real objects and secretly stored, we find in (95) other more readily verified properties. To this regard, honey objects are *conspicuous*, meaning that they are visible and exposed similarly as sugar objects.

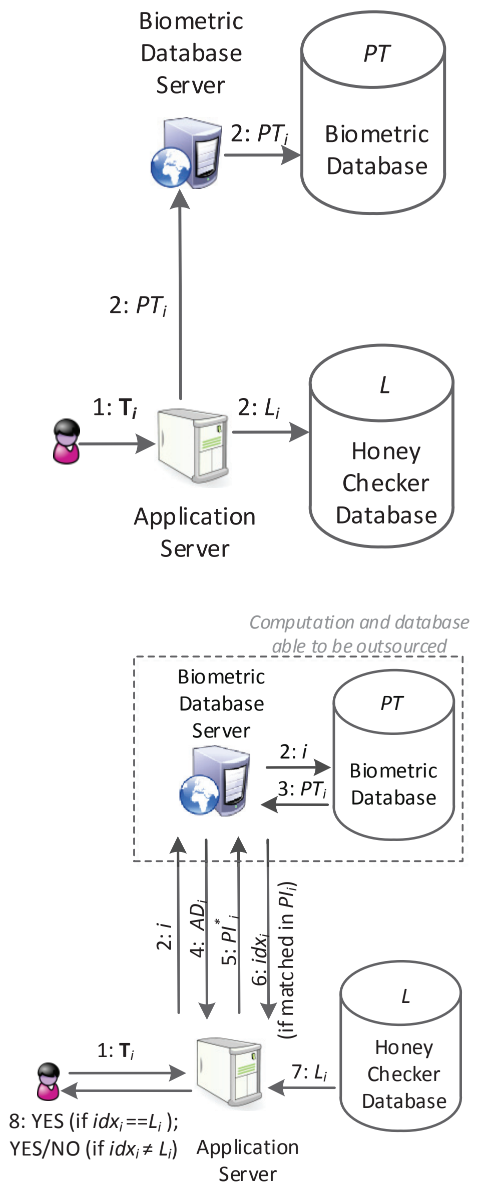


Figure 3.1: Architecture for a honey templates based biometric system: enrolment (up) and verification (down)

Another property of good honey objects is *non-interference*. There should be no correlation between sugar and honey objects, and of course between them and the plain objects from where they were generated.

3.4 Honey Templates first proof of concept: Algorithm I

The architecture of a honey templates based biometric system, proposed in this work, can be applied to different biometric characteristic. In this section we propose a first proof-of-concept for the generation of honey templates. Our first attempt is in the construction and protection of honey face templates. The protection mechanism we have adopted on the faces' feature vectors, is the plain-feature-defined sub-set selection borrowed by the idea in (111). In this work it is proposed a dynamic random projection scheme applied in fingerprints. The method alleviates security concerns due to the stolen token by increasing the computational complexity to search for the unprotected biometric features. This is achieved by a projection process which dynamically assembles a random projection matrix from a set of candidate projection vectors. The selection of projection vectors is decided by the biometric feature vector itself and thus forms a nonlinear projection process, without the need for external secret keys. In our application to faces we adopt this idea for the BTP and for the honey templates we collected a database of publicly available images and constructed a shuffling mechanism of their feature vectors. Principal Component Analysis (PCA) was the extraction algorithm used for the feature vectors with a dimensionality of n=20. PCA is a mathematical procedure that converts a set of correlated variables to a set of uncorrelated variables (53). It finds a reduced set of vectors which could serve as a basis from which all other vectors are generated as their linear combination. Face images are seen from the PCA technique under the same point of view. Their aim is to find the most informative base vectors, called *eigenfaces*. Recognition performance degradation can be anticipated from the adoption of BTPS as discussed in (99).

Experiment databases

To make a quick proof-of-concept evaluation of the proposed honey template based biometric system, we created two small-scale face databases denoted as DB^a and DB^t representing an *auxiliary database* (for purposes of PCA training and the construction of AD_i used by BTPS) and a *testing database*, respectively. DB^t is formed by 40 faces with 10 samples each as a sub-set of the ORL face database (81). To be fair in performance evaluation, DB^{aux} is formed by 40 faces from public websites with 1 sample for each face. All the 40 samples in DB^{aux} were cropped and normalized in their size to the same specification of those ORL samples (i.e., $92x112$ pixels, 256 gray scales).

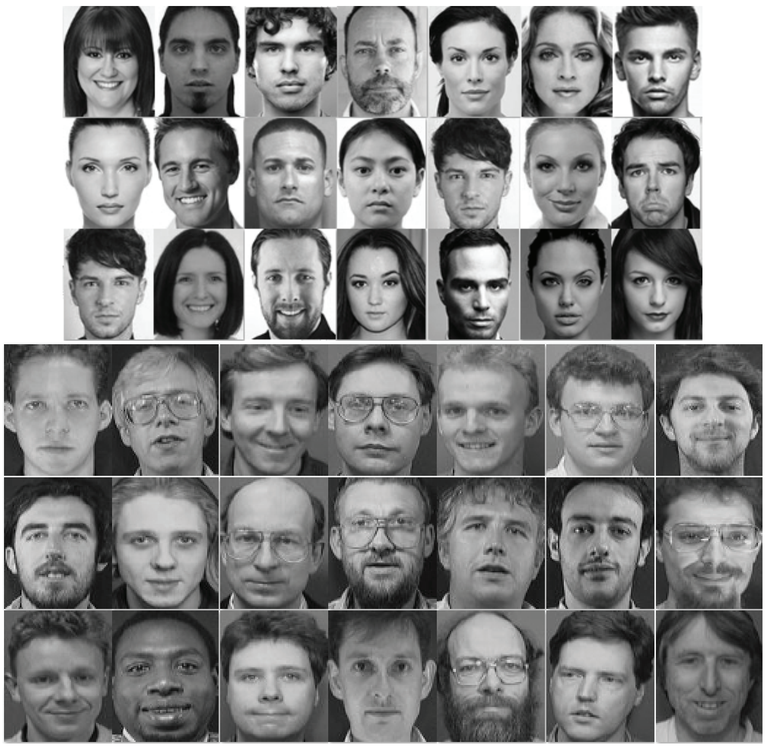


Figure 3.2: Samples in DBaux which is used for eigen faces training and used as AD_i by BTPS (up), Samples in DB^t which is used for recognition performance (down).

Feature vector construction and BTP

We took the first $N = 20$ PCA coefficients $c^t = (c^{t1}, c^{t2}, ..., c^{t20})$ (weights of eigenfaces that are trained from DB^{aux}) to decompose each of test sample in DB^t of a test sample as the sugar biometric feature vector $SB = c^t$. We define an indicative binary vector $v = (v_1, v_2, ..., v_{20})$ as:

$$v_j = \begin{cases} 0 & \text{if } c_{tj} \geq 0 \\ 1 & \text{if } c_{tj} < 0 \end{cases} \tag{3.1}$$

According to the indicative binary vectors we can randomly group DB^{aux} into two non-overlapped sub-sets DB_1^{aux} and DB_2^{aux} with 20 samples each. Then we generate 20 PCA coefficient feature vectors, denoted as $c\mathbf{1}_{auxq} = (c\mathbf{1}_{auxq1}, c\mathbf{1}_{auxq2}, ..., c\mathbf{1}_{auxq20})$ and $c\mathbf{2}_{auxq} = (c\mathbf{2}_{auxq1}, c\mathbf{2}_{auxq2}, ..., c\mathbf{2}_{auxq20})$, from each of the two sub-sets, respectively, with $1 \leq q \leq 20$. From the two groups of feature vectors $c\mathbf{1}_{auxq}$ and $c\mathbf{2}_{auxq}$ ($1 \leq q \leq 20$), a 20-dimensional mask vector $m = (m_1, m_2, ..., m_{20})$ is constructed as:

$$m_j = \frac{s_j}{20} \sum_{q=1}^{20} |(1 - v_q)c1_{auxqj} + v_q c2_{auxqj}| \tag{3.2}$$

where s_j is defined as a sign value (+1 or -1) in the equations below:

$$s_j = \begin{cases} +1 & \text{if } A_j \geq 0 \\ -1 & \text{if } A_j < 0 \end{cases} \tag{3.3}$$

$$A_j = \sum_{q=1}^{20} ((1 - v_q)c1_{auxqj} + v_q c2_{auxqj}) \tag{3.4}$$

Now with the plain feature vector c^t and the mask vector m, we can generate the protected sugar template $ST = (st_1, st_2, ..., st_{20})$ as:

$$st_j = rem(rem(w_1 m_j, 2\sigma) + w_2 c_{tj}, w_3 \sigma) \tag{3.5}$$

where σ is an estimated standard deviation of all feature vectors in DBt, and the three weights can be tuned to achieve the best trade-off between security and recognition performance. rem is a remainder operation, but that keeps the sign, differently from the mod operation. The values of the weights w_i with i in $\{1, 2, 3\}$ will be discussed later during the experimental evaluation.

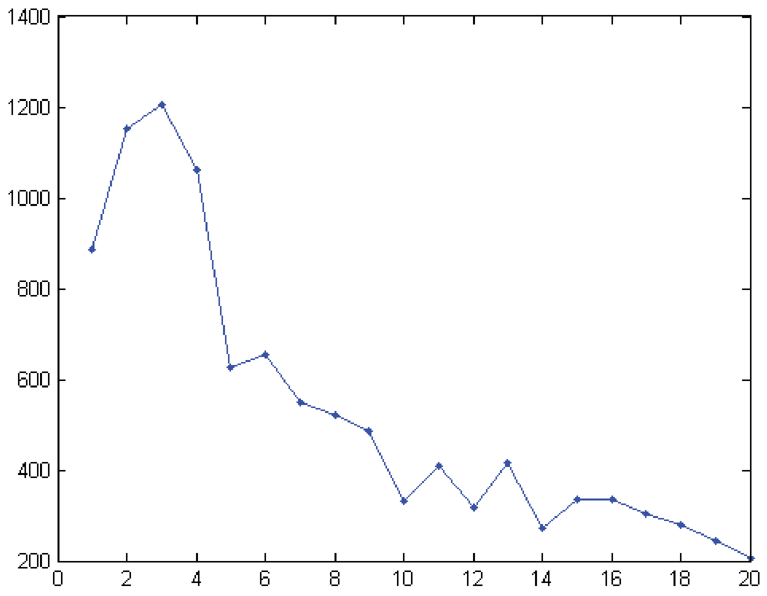


Figure 3.3: Standard deviation values for PCA coefficiens of the testing database DB^t.

In Fig. 3.3 is shown the standard deviation of PCA components in the testing database. In a brief, the above BTPS relates the mask feature vector generation to the plain feature vector's components' signs and thus obtains varied mask feature vectors for each different plain feature vectors.

Honey templates generation
For each sugar protected template, we generated 15 honey protected templates to make the total number of templates $K = 16$. The generation process of a honey template is shown in the diagram of Fig. 3.5. This process is the same as that of a sugar template, except that the plain feature c^t is replaced by a random feature vector with each component's dynamic range $[-0.5, +0.5]$. In Fig. 3.4 we can visually compare the distribution of the generated templates in both protected domains: honey and real.

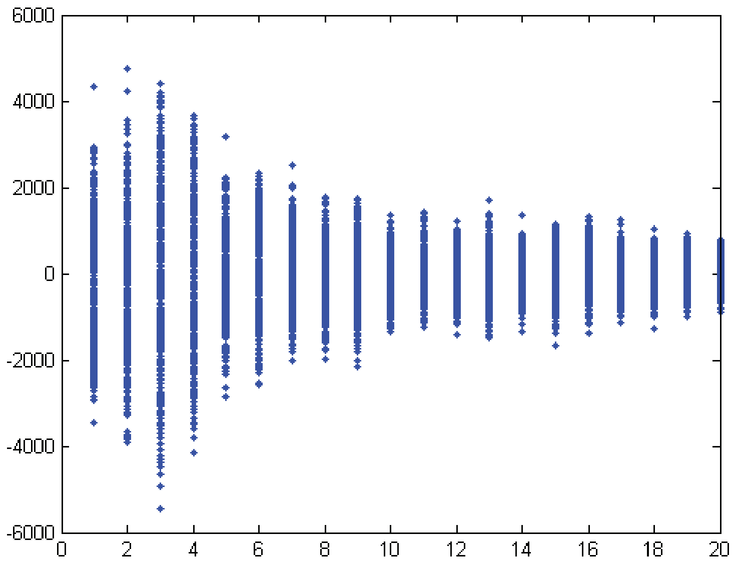
3.5 Security and Irreversibility evaluations

In our BTPS design, AD_i is assumed to be public and length N of the PCA feature vector (i.e., how many eigenfaces are used) decides the security level since m is constructed by N selected feature vectors from $c\mathbf{1}_{aux}$ and $c\mathbf{2}_{aux}$. In our quick proof-of-concept experiment, N is set to be 20 and therefore the complexity of identifying the correct 20 feature vectors selected is 2^{20}. In practice, we can extend the security depth by increasing N, or keep the DB_{aux} as a secret parameter.

Whether an adversary can distinguish the generated sugar protected templates from those honey protected templates is the key criterion for the proposed honey template generation method. We further check if a machine learning algorithm can be used to classify the protected templates and identify the ST from other HTs. SVM was trained by half (200 test PIs) generated protected templates with ground-truth labels (sugar or honey) and evaluated by the other half (the other 200 test PIs) generated protected templates in our experiments and the classification results are given in Table 1 with various settings. It indicates that the proposed honey template generation method can well hide the sugar template among those honey ones though the FMR and the FNMR are less than 50%, they should be already large enough to discourage the adversary to launch a masquerade attack if N is large enough. In Table 1, the values $(1, 1, 1)$ of (w_1, w_2, w_3) imply that we have not taken into consideration these parameters.

The need for a deep understanding of the security properties of our BTP scheme is crucial and has to be well characterized. In (71) is shown that all template protection schemes including fuzzy encryption, biometric salting and cancelable biometrics offer limited protection against different attacks. For instance, fuzzy encryption is vulnerable against linkage attacks, biometric encryption to Hill-Climbing

(a)



(b)

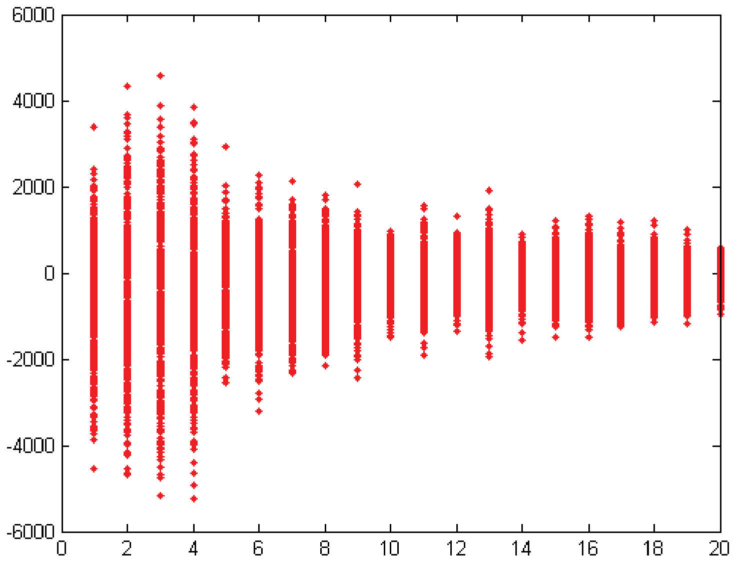


Figure 3.4: Dynamic range of protected Sugar (a), and protected Honey Templates (b), with the mask vector mechanism.

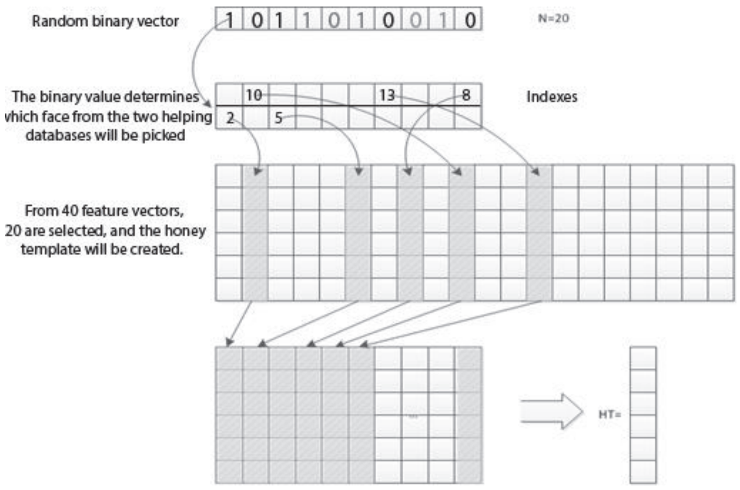

Random binary vector
1 0 1 1 0 1 0 0 1 0
N=20
The binary value determines which face from the two helping databases will be picked
10
13
8
2
5
Indexes
From 40 feature vectors, 20 are selected, and the honey template will be created.
HT=

Figure 3.5: Honey templates generation diagram in the mask vector BTP.

attack and helper data scheme is limited especially by the correlation of features.

3.5.1 Irreversibility evaluation tests and results

To show that the honey-based BTP on faces shows lack of irreversibility, we tested the correlation values: (1) between the feature elements and the protected sugar templates; (2) between the excerpt of feature vectors used to create the honey templates and the honey templates themselves; and (3) between sugar templates and the correspondent honey templates in their protected and unprotected form.

Correlation test between plain feature vectors and protected templates
Correlation tests are performed to measure how strong the relationship between two vectors or variables is. Our first test to this regard is between the plain feature vectors c^t and the protected templates ST of user i. In our terms, the first test will be applied between the PCA coefficients $c_t = (c_{t1}, c_{t2}, ..., c_{t20})$ (weights of eigenfaces that are trained from DB^{aux} to decompose each of test sample in DB^t) and the sugar protected templates $ST = (st_1, st_2, ..., st_{20})$. We have run this test for different values of the tuning parameters triplet (w_1, w_2, w_3). The results are presented in Table 3.3 where we can see that there is zero correlation between the two vectors. This means that there can be no information leakage from the protected sugar template.

Correlation test between selected or unselected feature vectors and protected

Table 3.3: Classification of protected templates by SVM for different values of w_1, w_2, w_3.

w1	w2	w3	FMR	FNMR
1	1	1	0.3433	0.4050
1	1	36	0.2287	0.4000
1	1	72	0.2380	0.4200
1	2	6	0.1470	0.4350
1	2	36	0.1553	0.3800
1	2	72	0.2417	0.4100
2	2	6	0.1150	0.4750
2	4	6	0.1537	0.3800
36	6	6	0.1573	0.3500
72	36	6	0.1767	0.3950

templates

As we mentioned, the sugar template is constructed by selecting randomly a part of the set of plain feature vectors. In order to argument the no correlation between the final protected sugar template and these two separate and non-overlapping subsets we performed two correlation tests between the selected vectors used to construct the sugar templates and the protected sugar templates. The unselected portion of the set of feature vectors can be expressed as:

$$A_j^* = \sum_{q=1}^{20}((v_q c1_{auxqj} + (1 - v_q)c2_{auxqj}) \tag{3.6}$$

The results of the two abovementioned tests are presented in Table 3.4.

The problem of irreversibility between templates must be considered also for the honey protected template case. We can see that between the selected vectors of the sugar feature vectors and the corresponding honey-templates the correlation mean value is very low. Nearly the same correlation situation can be seen between the protected honey templates and the counterpart of the selected vectors in the set of the plain feature vectors. The results are presented in Table 3.5.

The correlation between the selected feature vectors that we have chosen to build the honey-templates and the honey-templates themselves is close to zero. The same holds for the correlation between the unselected feature vectors and the constructed honey-templates. The Equal Error Rates for the sugar templates and the honey templates are measured for different values of the three tuning coefficients

Table 3.4: Correlation values between plain feature vectors; selected and unselected vectors; and the protected sugar template.

No.	A	B	Corr (A, B)
1	Sugar feature vectors (SB)	Protected Sugar Template (ST)	0
2	Selected vectors for Sugar Template (A_j)	Protected Sugar Template (ST)	0,06423
3	Unselected vectors for Sugar Template (A_j^*)	Protected Sugar Template (ST)	-0.06416

Table 3.5: Correlation values between selected and unselected vectors and protected honey template

No.	A	B	Corr (A, B)
1	Selected vectors for Sugar Template (A_j)	Protected Honey Template (HT)	0,0506
2	Unselected vectors for Sugar Template (A_j^*)	Protected Honey Template (HT)	0.05004

$w_1, w_2, and w_3$. In Table 3.6 are listed some combination cases of the EER of sugar templates and the EER of honey templates. The optimal values couple of (EERSugar, EERHoney) is where the triplet (w_1, w_2, w_3) is equal to (72, 36, 6).

3.5.2 Performance comparison between protected and unprotected case

Unprotected case

The first comparison is made between samples of unprotected sugar templates. As a first score we measure the similarity between sugar feature vectors which belong to the same subject. We have called this $GENS(SB, SB)$ that stands for *Genuine Score* of the sugar feature vector. A second score we have calculated is the similarity between the first samples of the sugar feature vectors. We have called this $IMPS(SB, SB)$ which stands for *Imposter Score* of the sugar feature vector. As a similarity measure we have used the inverse of SSD (Sum of Squared Differences). For these scores we have the following equations:

Table 3.6: EER for sugar and honey templates for different values of w_1, w_2, w_3.

w1	w2	w3	FMR	FNMR
1	1	1	0.4078	0.4197
1	1	36	0,3993	0,4080
1	1	72	0,3993	0,4084
1	2	6	0.4249	0.4368
1	2	36	0.4214	0.4353
1	2	72	0.4214	0.4357
2	2	6	0.4365	0.4509
2	4	6	0.4668	0.4771
36	6	6	0.4749	0.4854
72	**36**	**6**	**0.4809**	**0.4994**

$$GENS_{(SB,SB)} = \frac{1}{\sum_{i=1,j=1}^{i=40,j=10}(SB_i(j) - SB_i(j+1))^2} \tag{3.7}$$

$$IMPS_{(SB,SB)} = \frac{1}{\sum_{i=1}^{i=40}(SB_i(j=1) - SB_{i+1}(j=1))^2} \tag{3.8}$$

The accuracy of the unprotected case is the Equal Error Rate (EER-unprotected case) and the results are presented in Table 3.7.

Protected case

The accuracy test of the unprotected case is repeated for the protected templates also. We have two other scores: $GENS(ST, HT)$ and $IMPS(ST, HT)$ for the EER evaluation. The GENS(ST,HT) comparison score is between genuine protected templates and the correspondent 15 protected honey templates, calculated as:

$$GENS_{(ST,HT)} = \frac{1}{\sum_{i=1}^{i=40}(ST_i(j) - HT_i(j+1))^2} \tag{3.9}$$

The last comparison score is between first samples of protected templates of the same subject.

$$IMPS_{(ST,HT)} = \frac{1}{\sum_{i=1}^{i=40}(ST_i(j=1) - HT_{i+1}(j=1))^2} \tag{3.10}$$

Results for both error rates are measured for different values of the tuning parameters triplet (w_1, w_2, w_3), where the best result is highlighted (Table 3.7).

Table 3.7: EER for unprotected and protected case for different values of w_1, w_2, w_3.

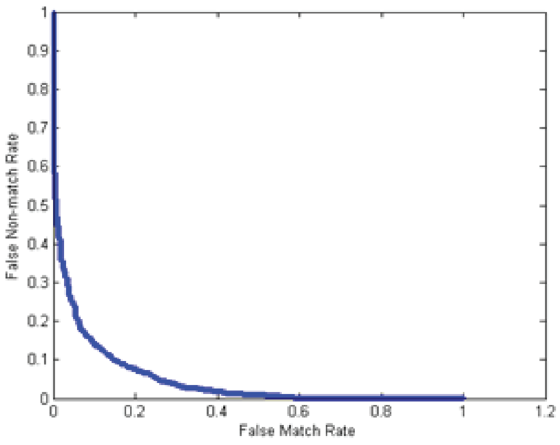
w1	w2	w3	EER (protected)	EER (unprotected)
1	1	1	0.1223	0.3351
1	1	36	0.1223	0.2791
1	1	72	0.1223	0.2801
1	2	6	0.1223	0.2000
1	2	36	0.1223	0.1735
1	2	72	0.1223	0.1580
1	6	36	**0.1223**	**0.1285**
2	2	6	0.1223	0.1696
2	4	6	0.1223	0.1975
36	6	6	0.1223	0.2577
72	36	6	0.1223	0.2704

3.5.3 Recognition performance evaluation

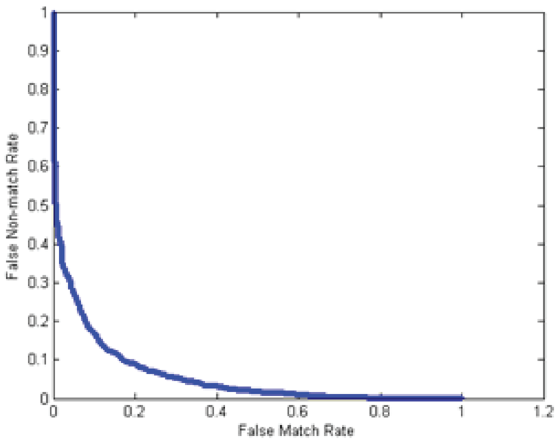
To evaluate the recognition property of the proposed BTPS and honey template based biometric system, we compare the plain template case (without BTPS and without honey templates) and the proposed BTPS and honey template based case. In the plain template case, we use the following testing protocol: for all 40 faces in the testing database DB^t, we cross compare the 10 PCA feature vectors from each same face and this resulted in 1800 genuine comparison scores; and for all 40 faces we use only the PCA feature vector from the first sample of each face to perform cross comparison and this resulted in 780 imposter comparison scores.

In the proposed BTPS and honey template based case, the highest comparison score out of the 16 comparison scores between the probe's PI^* and all 16 PIs in the database is recorded as the final comparison score to verify the probe.

From Fig. 3.3 we can see an example of recognition performance comparison between the plain PCA feature vector case and the proposed BTPS and honey template case with the setting . We also see performance degradation in other parameter settings. Recognition degradation, as in many other BTPS, can be limited



(a) Unprotected templates: EER = 0.1223



(b) BTPS with honey templates: EER = 0.1285

Figure 3.6: Recognition performance comparison: (a) Unprotected templates: EER = 0.1223 (b) BTPS with honey templates: EER = 0.1285.

in an acceptable range if the parameters are fine tuned.

3.6 Conclusions and further work

We borrowed the honeywords concept used for detecting leaked passwords in a biometric system and proposed a BTPS based honey template construction method in this research. The honey biometric protected templates can be used as chaff data to hide the storage address of the sugar (genuine) biometric protected template. Once a honey template is matched by a probe, the system can reasonably conclude with a high probability that the corresponding biometric data entries in the database had been already leaked and a pre-image masquerade attack is launched. This could be very helpful in detecting such data leakage accidents which cannot be achieved by existing biometric template protection schemes.

3.7 Brief chapter summary

In this chapter we provided an explanation of the honey-based idea applied in a biometric context. To give an answer to **RQ1**, we can assert that Biometric Template Protection Schemes can be augmented with Honey Templates, indistinguishable from real ones. We applied the experiments on a new BTPS, on a verification system based on faces, which comprises another contribution to this thesis. A large set of tests was performed for both criteria of irreversability and recognition performance (including feature vectors, protected templates, and honey templates.

The results from the test show that the system performance is deteriorated with the honey templates augmentation, but this level is acceptable compared to the overall effect that BTP schemes tend to give in this aspect. Regarding **RQ1.1**, we answer that, yes, honey templates influence the system performance in this case (EER increases from 12.25% to 12.85%, which is of a minor influence).

We can also conclude that the level of 12% of EER rates is very high for the mask-vector algorithm we presented in this chapter. By fine tuning the algorithm parameters, or by improving the algoritm the EER values we believe this rate would improve, but, as we first mentioned, this is a proof-of-concept that honey templates are applicable in a new BTPS.

In the coming three chapters we propose the application of honey templates to protection schemes that fall within the categories of Feature Transformation (Random Projection, and Filters), Biometric Cryptosystems, and Hibrid Schemes. We will see how for different existing protection schemes, provide different performances when augmented with honey templates.

Chapter 4

Biometric Honey Templates adoption in Feature Transformation Mechanisms

As we have mentioned, the aim of augmenting a biometric system with honey templates is to minimize privacy loss in the case of a possible database breach. In actual biometric verification systems, if an attacker steals the content of the biometric database, he could easily link the template to the subject, since they reside in the same space, and then he might reconstruct the biometric features from the protected templates by breaking their protection algorithm. In the system we are proposing in this thesis we aim to break this link between the subject (and their respective $PersonalInformation$) and the biometric content the system holds. And in this chapter, we further explore this scenario by designing a $game$ between the system $Defender$ and the $Attacker$ to better evaluate the later's efforts in being successful in his/her attempts.

4.1 Problem statement

We now suppose the attacker possesses a leaked biometric database and has knowledge on the defender system, algorithms and parameters. While thinking as an attacker, we also suppose that he owns Machine Learning led classification tools to discriminate real protected templates from synthetic ones. Classification is the process of predicting the class of some other given data. Generically speaking, they train data, construct a classification model, and upon receiving new data they predict (with a given accuracy) the class where this data would reside.

To follow this process on honey and sugar templates (namely $binaryclassification$), an attacker -except from the leaked database- should also own (or design, simulate, create) two separate sets of templates (honey and real) similar in terms of data parameters with those found in the leaked database, in order to build a reliable and accurate classifier for the division of the new set in two classes. With this tool the attacker increases the probability of guessing the real template, e.g. if there are $k = 15$ honey templates for each subjects, the probability p of guessing the real one, without the help of any tool, is equal to $1/(k+1)$, which is $p = 0.06$. With the classifier, the prediction might be such that there are $r = 3$ possible real templates in this set, i.e. the probability to have the real template among them is increased to $p_r = 0.33$, over 5 times higher.

4.2 Definitions for a honey-based Defender-Attacker game

For the creation of similar sets of templates, we further imagine and suppose that he/she is a knowledgeable attacker whose aim is to $mimic$ the target system. Our aim, as defenders, is to make this impossible, or hard enough for the attacker. The analysis we provide is supported by implementing two honey-based face verification systems: a Defender system *D*, and an Attacker system *A*. The aims of D are: a) to design a secure biometric system augmented with honey templates; and b) to accomplish the indistinguishability requirement on templates. The chosen BTP for this *game* is *Random Projection*, one of the *Feature Transformation* protection schemes.

The aims of A are: a) to build a biometric system *A* by mimicking *D*; b) to classify the leaked templates in order to be able to discriminate them and pick the genuine ones only, for further attacks. Before evaluating an attacker's efforts in mimicking the system design of the target biometric system, let's define what a generic biometric system augmented with honey templates is in this context. Here the asterix in the acronyms can be substituted by *D*-Defender or *A*-Attacker.

Definition 13 (Honey-based Biometric System) A biometric system augmented with honey templates, is the tuple:

$$BS^*_{HT} = (DB^*, DB^*_{aux}, \mathbf{BTP}^*, \mathbf{HT}^*) \tag{4.1}$$

where: DB^*,is the biometric database containing both sugar and honey templates in protected and randomized form, DB^*_{aux} is an auxiliary database, **BTP*** is the protection algorithm applied on templates, and **HT*** is the honey templates generation algorithm. . □

To mimic a target system, the attacker would operate on two levels: 1) *data level*, i.e. generate sugar and honey templates with the same structural properties as *D*;

and 2) *performance level*, i.e. having the same recognition performance as *D*. To achieve this it is supposed that:

a. The attacker *A* has knowledge on the target system *D* regarding: honey templates generation algorithm $\mathbf{BTP}^{D}$, template protection mechanism $\mathbf{HT}^{D}$, and recognition accuracies. As a metric for the recognition accuracy is used the *Equal Error Rate* (EER). The attacker goal is to create such a system that the difference between the two EERs should be infinitesimally small:

$$EER^{A} - EER^{D} = \varepsilon_r \tag{4.2}$$

b. He owns DB^{A}_{Aux} auxiliary databases similar to DB^{D}_{Aux}. The *similarity* relates to content feature properties, parameters and database structure.

c. The attacker owns a classifier C^{A} having an accuracy rate c to classify the two categories of templates.

From the defender's perspective c should be close to the ideal case, that it is impossible for the classifier to discriminate templates:

$$c^{D} \longrightarrow 0.5 \tag{4.3}$$

From the attacker's perspective the classification rate should be infinitesimally small for a clearer discrimination of the two kinds of templates:

$$c^{A} \longrightarrow \varepsilon_c \tag{4.4}$$

d. The attacker owns a leaked database from the defender system, DB^{D}, where sugar and honey templates are randomly stored for each subject. The aim of the attacker is to classify the database content with the tools he/she owns (C^{A}). After classifier's training with its own databases, the attacker will test the templates in DB^{A} to discriminate them and proceed with further attacks.

4.3 Experimental setup for mimicking *D*: Algorithm II

4.3.1 Subjects database creation (DB^{*})

In order to illustrate the described scenario of system A mimicking system D, experiments are carried out on *faces94* public database (examples presented in Fig. 4.1). This database comprises of three sets (*female*, *male*, *malestaff*) having a number of 153 subjects with 20 samples each. Its images background is plain

Figure 4.1: Samples of a subject in *faces94* database.

green, there are minor variations in head turn, and there are considerable face expression changes. Their image resolution is 180×200 pixels. Before applying feature extraction, images are processed further. They are converted in greyscale images, and the resolution is 92×112 pixels.

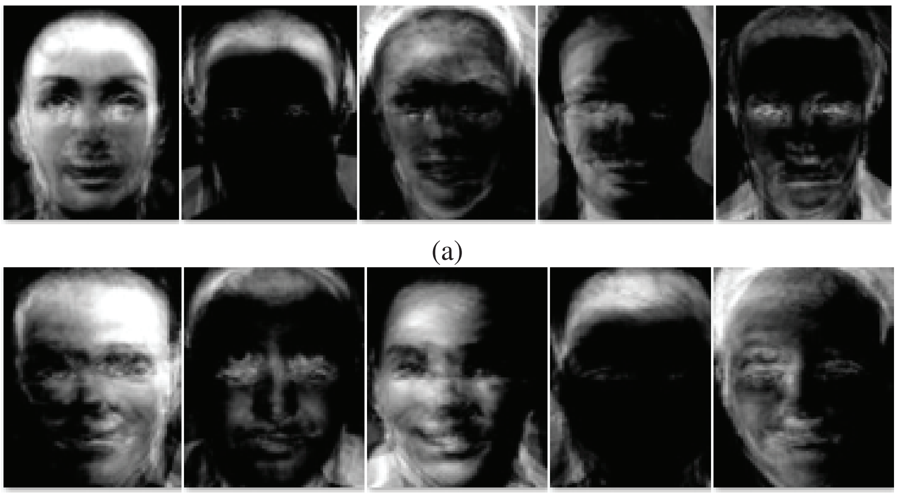
To equally create the two subjects databases, 140 subjects are chosen and equally divided for the two systems. The division is such that, *when templates are generated, they provide the same recognition performance at feature level and protected level*. In real-life scenario, an attacker can have large scale databases of images with the same parameters as the target system, and pick those which provide same recognition performance (we consider that A knows the value of EER^D).

After feature extraction from genuine users, a set of "honey features" should be created. To preserve the indistinguishability property at feature level too, honey features are generated by averaging a random number of *t*-images by respective databases DB^D and DB^A. In these experiments $k = 9$ honey templates are generated, i.e. there are in total 10 templates for each sample. Thus, the biometric database of the defender and attacker systems contain 70 subjects $\times 20$ samples/subject $\times 10$ sweet templates $= 1,400$ templates each.

4.3.2 Feature extraction and template protection

Sugar templates are generated after a feature extraction process based on *Principal Component Analysis*. From experimental observation, in both systems, the number of PCA components bringing most of the information from image features is equal to 20, generated with the help of an auxiliary database DB^*_{aux}.

The Defender system generates the *eigenfaces* by an auxiliary database DB^D_{aux}.


(a)

(b)

Figure 4.2: *Eigenfaces* from Defender (a) and Attacker (b).

It comprises of 40 subjects from publicly available images. The same does the Attacker. He owns DB^{A}_{aux} to generate his eigenfaces. In Fig. 4.2 are shown some samples of eigenfaces from Defender and Attacker feature extraction process.

Template protection

The protection mechanism applied on feature vectors is *Random Projection*. RP is a technique which provides unlinkable templates by projecting a biometric feature vector in a set of orthonormal random vectors. To achieve this, the feature vector is multiplied by a random real-value matrix (112). The dimensionality of the matrix is 20×40 meaning that the final representation of the templates will be real-number vectors of dimensionality equal to 40.

4.4 Recognition accuracy results

For the honey templates, these steps are followed. First, the defender and attacker generate for each sample $k = 9$ *honey feature vectors*. They use a pseudo-random number generator to *(1)* shuffle their templates, *(2)* pick $t = 30$ random images, *(3)* build an average feature vector, and *(4)* randomize the signs of each component. On both sugar and honey features the same protection algorithm is applied. The attacker's resulting templates will have the same representation as ST^D and HT^D of the defender (real-numbered vectors of dimensionality equal to 40).

As a recognition performance metrics, *Equal Error Rate* is used. The main components differing between D and A are: **1)** the auxiliary database DB^*_{aux}; **2)** the subjects' database DB^*; **3**) different matrices in the template protection mechanism. Each difference is tested and EER is measured.

Algorithm 4.1: Generating RP - protected honey templates HT^A and HT^D.

```
Input  : A set of honey features $hf^A$ or $hf^D$, and two integers $m$ and $n$.
Output: A set of RP-protected honey templates $ht^A$ or $ht^D$.
1 for $i \leftarrow 1$ to $n$ do
2     for $i \leftarrow 1$ to $m$ do
3         $ht_{ij} \leftarrow$ PRNG($rand_{value}$);
4 Return($ht_{ij} * hf_{ij}$);
```

1) In Table 4.1 are shown different EERs for the *Defender* and *Attacker* systems considering different contents of auxiliary databases. Experimental results show that the EERs do not change significantly if different auxiliary databases are used. For this test, 3 different auxiliary databases are observed and their images are comprised of 40 frontal face pictures of individuals. These images are collected on-line when *labelled for non-commercial reuse* (this is the same database used in Chapter 3). The most approximate combination is between round #3 of Defender and round #1 of Attacker. It is highlighted in Table 4.1 showing a difference of $0,012$.

Rounds	EER_{PCA} (Defender)	EER_{PCA} (Attacker)	Best EER difference
$\#1$	0,046	**0,0389**	
$\#2$	0,051	0,0366	**0,012**
$\#3$	**0,049**	0,0348	

Table 4.1: EER after feature extraction of Defender and Attacker systems for three **different auxiliary databases**.

2) The second test will proceed on the subjects database. The attacker can have a set of database records such that its recognition performance approximates that of the defender. In our experimental setup, we have generated two databases from the *faces94* database, such that their division will provide us two similar recognition rates. To achieve this we iteratively generate these divisions, showing the results in Table 4.2. Recognition performance has a stronger dependency from the database content, compared to the dependency from the auxiliary database. In this case more than 10 different divisions of databases are created. To illustrate the EER variability, only a part of them is presented, showing in the last row the most similar EER value.

Rounds	EER_{PCA} (Defender)	EER_{PCA} (Attacker)	Difference
#1	0,048	0,0397	0,0083
#2	0,054	0,041	0,013
...			
#10	0,038	0,047	-0,009
Best case	**0,045**	**0,0418**	**0,0032**

Table 4.2: EER after feature extraction of Defender and Attacker systems for **different biometric records**.

Protected domain

The last step in mimicking a biometric system considering the performance criteria is in the *protected domain*. In this step, templates are generated by multiplying feature vectors with random real-value *auxiliary matrices* AM, in order to increase their dimensionality to spread the template information. The attacker does not possess the random matrix generator, but he can try the following: does the system performance value depend significantly on AM? To answer this, 10 more rounds of protected templates are performed with different auxiliary matrices AM. Results are shown in Table 4.2.

In fact there is a slight difference, but the trend remains the same, yielding an important confidence result for the attacker. Regardless the different auxiliary matrices, the trends of EERs remain the same. I.e., if the defender system heritages a slightly higher performance from the feature level compared to the attacker, this trend will be transferred at the protected domain performance. Considering this result, the attacker system performance similarity criteria will be more easily achieved.

The final performance test is on augmented system with honey templates. The attacker will try different sets to check if the final recognition performance is affected. In fact, to evaluate this 10 rounds of *honey features* were generated and systems performances compared. The differences between the two system according to the performance criteria are slight. In fact from Table 4.3, and from the complete set of experiments, an average value of $\varepsilon_r = 1,4 \times 10^{-3}$ is achieved.

Rounds	EER_{PCA} (Defender)	EER_{PCA} (Attacker)	Difference
#1	0,048	0,0422	0,0058
#2	0,0462	0,0431	0,0031
...			
#10	0,0465	0,0443	0,0022

Table 4.3: EER after feature extraction of Defender and Attacker systems in the protected domain for **different auxiliary matrices** AM, without honey templates.

In Table 4.4 are shown the EER results of the two system protected templates for different rounds of honey features generation.

Rounds	EER_{RP} (Defender)	EER_{RP} (Attacker)	Difference
#1	0,0597	0,0611	-0,0014
#2	0,0584	0,0597	-0,0013
...			
#10	0,0605	0,0637	-0,0032

Table 4.4: EER after feature extraction of Defender and Attacker systems in the protected domain for **different honey features**.

We can conclude that an attacker successfully builds the biometric system BS^A to mimic BS^D. They have same template representation, same records cardinality, and very similar recognition performance.

4.5 A classification challenge for honey templates generation

Creation of honey biometric features can be done either by finding real biometric samples other than from the user's body characteristics or by biometric sample (or feature) synthesis. Taking real biometric samples as honey templates could be not only ethic sensitive but also costly in operation. Biometric sample or feature synthesis can be varied in technical difficulty depending on modalities and feature extraction methods. Here the difficulty should be understood as the difficulty in distinguishing the honey templates generated from synthesized samples (or features) from the sugar template generated from the real sample (or feature).

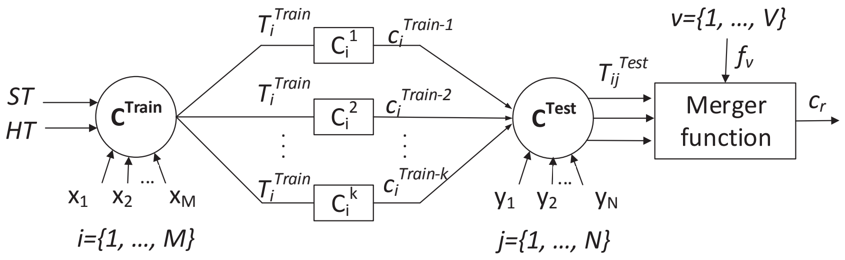


Figure 4.3: An inclusive multi-classification tool for biometric templates distinguishability.

4.6 Testing the Defender/Attacker game

Considering that an attacker successfully is able to mimic a target biometric system augmented with honey templates BS^D_{HT}, the next step is building the classifier C^A and train it with the templates ST^A and HT^A he now owns.

In the implementation of biometric honey templates schemes in Chapter 3 (Mask Vector) and as we will see in Chapter 5 (Bloom Filters), different classifiers are trained and used for testing of face biometric templates. In both works protected templates are represented as real value vectors (after a feature transformation protection scheme) and binary matrices (protection mechanism are Bloom Filters). In these implementations we have fixed training and testing set. In the context of this chapter, considering a knowleadgeable attacker, we propose that he/she will follow a long list of test runs to find the one with the highest confidence. Since in our experimental setup it is not fully possible to proceede with a similar test list, we simmulated this scenario by building different training test sizes and positions (position here means the starting index of the training subset) a classification framework.

To answer this question, our experimental setup does not offer the very large training set that the attacker might have, but we simmulate this scenario by creating a variety of traning sets, by creating a considerable number of subsets from our actual training set. These subsets vary in size, position (i.e. starting table index from which they are extracted), and the different training sets will be run in different classification algorithms. This is included in the proposed framework below. The results will tell us (as Defenders), after an Attacker successfully mimics D, under which classification circumstances he/she will better distinguish sugar and honey templates.

(A) a training set C^{Train}. The input will be a set of equal number of sugar and honey templates.

(B) a set of *M* input conditions. There can be different input conditions $X = x_1, x_2, ..., x_M$ applied on the training set, e.g. x_1 determines the training set size, x_2 the training set position, etc.

(C) a set of *K* standard classifiers. All classifiers (C_i^1, ..., C_i^K) will train the same set T_i^{Train}, dependable on the input condition *i*.

4.6.1 Classification algorithms

In this work a total of 12 classifiers from 5 different categories are included to test the indistinguishability property between sugar and honey templates: *SVM*, *k-Nearest Neighbor*, *Decision Trees*, *Discriminant analysis*, *Logistic Regression*.

1. SVM Are fast and accurate for binary classification and for high dimensional data. When classification is computed, the *kernel* needs to be specified which is simply a *similarity function* used to quantify the degree of similarity between objects. A *binary SVM* classifies templates by finding the best hyperplane (65) that separates data points of one class from those of the other class. The largest the margin between the classes, the best the hyperplane is. Depending on the function, the kernel can be: *linear*; *quadratic*; or *cubic*.

2. k-Nearest Neighbor Is a way of categorizing data based on their distance to other data (neighbors). There are different distance metrics, which will define the type of kNN classifier (110). Depending on the distinction level between classes, the number of neighbors defined in the kNN is: *fine*, *medium*, or *coarse*. In general, kNN classifiers have good accuracy in low dimensions (83).

3. Decision Trees. The number of splits defines the tree as: *fine*, *medium*, or *complex*. Even though the training set might show a high accuracy classifier, in complex trees this accuracy drops during testing. This is noticed in almost all classifiers.

4. Discriminant Analysis. Is a popular classification algorithm accurate and easy to interpret. It assumes that different classes generate data based on different Gaussian distributions. To train this classifier, the fitting function estimates the parameters of a Gaussian distribution for each class.

5. Logistic Regression. Another classifier for two classes. It models the class probabilities as a function of the linear combination of predictors.

(D) a testing set C^{Train}. It comprises of a random number of sugar and honey templates.

(E) a set of *N* output conditions. On the testing set will be applied the conditions $Y = y_1, y_2, ..., y_N$. E.g. y_2 which might determine the testing set size, should be such that $y_2 \neq x_2$.

(F) a merger function f_v. It will fuse the output of the testing results T_{ij}^{Test}. The merging function could be: *maximum score*, *minimum score*, *weighted score*, etc. We suppose there are V different merging functions. From an attacker's perspective the merger function block of the classification mechanism is of no interest. In the actual implementation, there is not needed a final classification score.

In Fig. 4.3 is shown the inclusive multi-classifier diagram containing the elements described above. The resulting set of tests is comprised of $\{T_{ij}^{Test}\}$. To find the cardinality of this set, the complexity of the classification mechanism C is calculated. For the generic case:

$$Complexity(C) = M \times K \times N \times V \tag{4.5}$$

Example 1

Let's suppose we have two input conditions: x_1 determines that the training size is equal to 200 sugar and honey templates; x_2 considers for training first templates in every subject only; there are 4 different classifiers (*fine SVM*, *complex Trees*, *Linear Discriminant*, *Logic Regression*), there are two testing conditions again determining size and position, and three different merging functions (*max*, *min*, *average*). As a result, there will be: $2 \times 4 \times 2 \times 3 = 48$ different classification results c_{r_i} with $i = \{1, ..., 48\}$.

We now simulate the scenario where Defender and Attacker build a classification tool but use it for different purposes. They train it with the data they respectively own. Starting from the training sets conditions, on the 1400 sugar templates and 12,600 honey templates in each system 7 different slot sizes *q* are trained (*100*, *200*, ..., *700* sugar templates and the same amount of honey templates). The training set is:

$$\mathrm{C}_{\text{Train}} = \mathbf{ST}_q \cup \mathbf{HT}_q \tag{4.6}$$

for $q = \{100, 200, \ldots, 700\}$

4.7 Results and algorithm analysis

After the training set, accuracy of the classifiers algorithm was measured and results presented in Table 4.5. Here we add the processing time of each algorithm, calculated via the Classification Learner application, Matlab [1]. For each classifier in our context we need to measure two different rates, the number of honey templates classified as sugar templates, and vice versa. These rates will provide us the classifiers accuracy in distinguishing templates, but these rates will be differently analysed from the two points of view, those of the Defender and Attacker. Below we provide the definitions of these metrics:

Definition 14 (False Sugar Templates match Rate) *FSTR* represents the False Sugar Templates match Rate, and it measures the number of *honey templates* classified as *sugar templates* in a testing set C^{Test}.

$$FSTR = \frac{|c_{HT \rightsquigarrow ST}|}{|HT|} \tag{4.7}$$ □

Definition 15 (False Honey Templates match Rate) *FHTR* represents the False Honey Templates match rate, and it measures the number of *sugar templates* classified as *honey templates* in a testing set C^{Test}. □

$$FHTR = \frac{|c_{ST \rightsquigarrow HT}|}{|ST|} \tag{4.8}$$

Since different input conditions are applied to the training phase, it becomes harder to synthetize all classification results. Having applied 7 different training sizes, 8 different training slots, 12 classifiers, and all these applied on two systems it results in $7 \times 8 \times 12 \times 2 = 1344$ different classification tests. Table 4.5 shows the classification accuracy and training time of the 12 classifiers, for a specific training set. What is observed from the results, is the huge variability among different training sizes, and slot positions. To illustrate this, in Table 4.6 are presented FSTR and FHTR for a classifier (complex Decision Trees), for both systems.

No.	Classifier	Category	Accuracy (%)	Time (sec)
1	Decision	Complex	95,5	2,4951
2	Trees	Medium	95,5	0,52415
3		Simple	95,5	0,60938
4	Discriminant	Linear	69,0	0,8311
5	Analysis	Quadratic	88,0	0,5894
6	Logistic Regression	NA	90,5	2,397
7	SVM	Linear	85,0	1,0442
8		Quadratic	88,0	0,478
9		Cubic	99,0	0,40798
10	kNN	Fine	99,5	0,77012
11		Medium	93,5	0,43037
12		Coarse	60,5	0,46749

Table 4.5: Classification accuracy, and training time for $K = 12$ different categories of classifiers, for $q = 100$ in the training set.

Here are some observations. The accuracy rate of a classifier in the training set, is not always reflected in the testing set. And this can be seen from the results in Table 4.6, if we compare the rates with the accuracy value of $95,5\%$ of a complex Decision Tree. In all other classifiers the same result is observed, which means that the input conditions x_i highly determine the resulting classification rates for templates distinguishability.

As a result, four different relations between the two rates are considered, after classifier's training process. The first two mentioned in the Definitions 4.3 and 4.4 present the ideal case in each of the two perspectives.

Definition 16 (Perfect classification - Defender) *From a Defender's perspective the perfect classifier C^** is when:

$\{FSTR \rightarrow 0,5\}$ AND $\{FHTR \rightarrow 0,5\}$

This definition tells us that if half of sugar templates are classified as honey templates, and half of honey templates are classified as sugar templates, i.e. we achieve *perfect indistinguishability*.

[1]Matlab, Statistics and Machine Learning Toolbox: Classification Learner. Accessed, April 2016.

q	$FSTR^D$	$FHTR^D$	$FSTR^A$	$FHTR^A$
100	0,50	0,53	0,97	0,10
200	0,785	0,20	0,45	0,365
300	0,693	0,46	0,77	0,24
400	0,73	0,318	0,71	0,285
500	0,554	0,312	0,626	0,158
600	0,498	0,18	0,63	0,113
700	0,564	0,084	0,626	0,087

Table 4.6: Classification rates (FSTR and FHTR) of a complex Tree Decision classifier for different training set sizes q. Two sides of the table relate to Defender (D) and Attacker (A).

Definition 17 (Perfect classification - Attacker) *From an Attacker's perspective the best classifier* C^* is when:

$\{FSTR \to 0\}$ AND $\{FHTR \to 0\}$ □

In this case it means all sugar templates are classified as sugar templates, and all honey templates are classified as honey template, yielding *perfect distinguishability*.

There are two more cases, when a classifier fails to classify. If $FSTR \to 0$ and $FHTR \to 1$, the testing templates are all considered by the classifier as *sugar templates*. If $FSTR \to 1$ and $FHTR \to 0$, the testing templates are all considered by the classifier as *honey templates*. There is another case, when $FSTR \to 1$ and $FHTR \to 1$. This means that templates are classified oppositely to what they truly are (sugar as honey, and vice-versa). This is an odd situation which we did not encounter in our testing datasets. Now that the Defender and Attacker have their respective classification rates, the last effort of the Attacker is to test the database content DB^D he has previously stolen. Classification results are presented in Table 4.6.

As results show from Table 4.5, the Linear SVM classifier presents the lowest values of $FSTR^A$ and $FHTR^A$. Its rates are still high enough not to allow him to efficiently predict template types. We can conclude that even though an attacker can mimic a given biometric system, he can not discriminate real and honey templates.

No.	$Classifier$	$FSTR^A$	$FHTR^A$
1	Complex DT	0,8636	0,1271
2	Medium DT	0,8438	0,1777
3	Simple DT	0,8665	0,1392
4	Linear Discr.	0,7000	0,4108
5	Quadratic Discr.	0,7517	0,2488
6	Log. Regression	0,7198	0,3804
7	**Linear SVM**	**0,6297**	**0,4534**
8	Quad. SVM	0,9490	0,0349
9	Cubic SVM	0,9655	0,0268
10	Cosine kNN	0,8507	0,1335
11	Cubic kNN	0,8040	0,1832
12	Weighted kNN	0,8142	0,1409

Table 4.7: Average classification rates (FSTR and FHTR) of an Attacker System for discrimination of sugar and honey templates in a Defender database (considering x_1).

4.8 Brief chapter summary

The work of this chapter arouse form the problem that in actual biometric verification systems, if an attacker steals the content of the biometric database, he could easily link the template to the subject, since they reside in the same space. In the aim of breaking this link between the subject (and their respective $PersonalInformation$) and the biometric content the system holds, we further explored this scenario by designing a classification $game$ between the system $Defender$ and the $Attacker$. This way we can better evaluate the later's efforts in being successful in his/her attempts.

We estimate that an attacker -except from the leaked database- should also own (or design, simulate, create) two separate sets of templates (honey and real) similar in terms of data parameters with those found in the leaked database, in order to build a reliable and accurate classifier - i.e. their next aim is to mimic D in terms of parameters. In this chapter we designed the Defender system D, considering that the attacker is highly knowleadgeable about D and owns different classification tools, we concluded that an attacker can mimic D.

Results help us answer positively to our research question **RQ1** (can a BTP scheme be augmented with Honey Templates), and to **RQ1.5**, as Yes, it is possible for an attacker to mimic a honey-based system in terms of its parameters, and use it to classify leaked templates. The level of confidence of the attacker in the classification framework depends on the training sets, and algorithms. Another contribution in this Chapter are the definitions that add to the metrics apparatus regarding honey templates, specifically the rates FSTR and FHTR, making the answer to the research question **RQ1.3** more complete. The values of these metrics show the attacker's and defender's confidence in their systems, allowing us to better define in the future our protection mechanisms.

As future work the Defender-Attacker game in this chapter can be extended in other setups. We have included in later research of this thesis an algorithm for password generation based in Deep Learning algorithm (Appendix A) and evaluated as according to the ideas implemented in this chapter. Another contribution of our work is the fact that mimicking scenario performed by the possible Attacker might be subject to future implementation including intelligent agents in simulation, that would improve an attacker's evaluation of synthetic data.

Chapter 5

Biometric Honey Templates adoption with Bloom Filters

Biometric verification can be considered one of the most reliable approaches to person authentication. However, as we mentioned throughout this thesis, biometrics are highly sensitive personal data and any information leakage poses severe security and privacy risks. Biometric templates should hence be protected and impersonation with stolen templates must be prevented, while preserving system's performance. In the present chapter, a general biometric template protection scheme based on Honey Templates and Bloom filters is proposed, in order to grant privacy protection to the enrolled subject and detect the use of stolen templates.

This chapter covers one other implementation of Honey Templates among the Feature Transformation mechanisms. The performance and security evaluations show the soundness of the proposed scheme for facial verification. The benchmark is conducted with the publicly available BioSecure Multimodal DB and the free Bob image processing toolbox, so that research is fully reproducible. We include the idea of classification framework presented in Chapter 4 and provide the last metrics for the indistinguishability property.

5.1 Introduction

In the present chapter, we propose a general (trait-independent) BTP scheme applied on face templates which fulfils the requirements of the ISO/IEC IS 24745 (49) for BTP schemes, and at the same time is able to detect and prevent masquerade attacks. To that end, a combination of the Honey Templates framework proposed in (113) and Bloom filter based protected templates (88, 38) is used.

The former allows us to detect the use of stolen templates, and further protect the fully unlinkable and irreversible Bloom filter based templates. The main advantages of the latter with respect to other BTP approaches are the generality - it has already been successfully applied to iris (88), face (38), fingerprint (69, 6) and a multimodal system based on iris and face fusion (91), and the fact that it shows no performance degradation (as previous works have shown) fulfilling at the same time the irreversibility and unlinkability requirements for cancelable biometrics.

In order to carry out reproducible research, experiments are run over the publicly available multimodal BioSecure database (82) and a free implementation of the selected state-of-the-art face verification system (117) within the Bob Image Processing Toolbox (9). To the best of our knowledge, this is the first biometric experimental evaluation of a Bloom filter protection scheme which can also handle template leakage. The rest of the this research section is structured as follows. Related works to BTP, honeywords and Bloom filters are presented in Sect. 5.1.1. The new scheme is described in Sect. 5.2. Then the experimental framework for the system evaluation and results are presented in Sect. 5.3 and 5.4 respectively. Final conclusions are drawn in Sect. 7.5.

5.1.1 Masquerade Attacks and Honey Templates

In the biometric context, in a honey-based system the use of stolen templates can be detected when the *pseudonymous identifier* (PI^*) is presented again to the system: if it is a sugar template, the system will detect the leakage and impersonation attempt. A particular case study for facial verification was presented in (73) using irreversible Eigenface-based templates. The performance, security and irreversibility evaluations show the feasibility of such an approach.

5.1.2 Biometric Template Protection and Bloom Filters

Very recently the Bloom filter concept was applied to iris-based biometric systems in (88). A Bloom filter is a simple space-efficient data structure for representing sets, which supports membership queries. They were initially proposed to enable fast and efficient database queries (11), and, in the last decades, they have become popular in the networking literature (19), proving their efficiency for numerous and diverse applications.

In the context of biometric verification, it is proposed in (88) to divide the binary iriscode, common to most iris recognition systems (26), into $nBlocks$ blocks comprising $nBits \times nWords$ bits. From each such block, one Bloom filter, $\mathbf{b}$, of length 2^{nBits} is computed - the final cancelable template is thus composed of $nBlocks$ Bloom filters.

In order to extract a Bloom filter from a given binary block, columns (denoted as words) are mapped to their decimal value, and the corresponding index in $\mathbf{b}$ - initially set to zero - is set to one. Each bit can be thus set to one multiple times, but only the first change has an effect. Irreversibility in this scheme is granted since, given a single Bloom filter $\mathbf{b}$, the reconstruction of the corresponding binary block involves an arrangement of $|\mathbf{b}| \leq nWords$ different words to a binary block of length $nWords$, where $|\mathbf{b}|$ represents the number of activated indexes in $\mathbf{b}$. It has been shown that even small values of $|\mathbf{b}|$ yield relatively large number of different possible blocks, thus granting irreversibility (44).

In subsequent works, the concept was extended to face verification (38) and to multibiometric systems based on mixing iriscodes (90) or on the feature level fusion of face and iris (91). Regarding fingerprint verification, two different approaches have been recently proposed utilizing variable length templates based on minutiae vicinities (69) and fixed-length templates based on Minutiae Relation Codes (6). It should be noted that the privacy concerns were raised in (44, 18) were recently addressed in (37). For more details on Bloom filter based template protection, the reader is referred to (38, 89).

5.2 Proposed System: Honey Bloom Filter Template Generation

As depicted in Fig. 5.1, the proposed system builds upon the Honey Templates approach to counterfeit masquerade attacks, which, in turn, makes use of the Bloom filter based scheme for the generation of irreversible and unlinkable biometric templates (Fig. 5.2).

5.2.1 Honey Templates

5.2.2 Bloom Filter Based Protected Templates

In order to generate protected biometric templates in accordance with the ISO/IEC IS 24745 standard (49), the method proposed in (37) will be used. In the present scheme, this sub-system can receive two different inputs: i) a biometrics sample at authentication time (Fig. 5.1 right), or ii) a binary feature vector at the database generation step (Fig. 5.2), which can comprise real biometric information of the i-th subject ($\mathbf{T}_i^1$), or a random binary vector ($\mathbf{T}_i^j$, with $j = 2, \ldots, K+1$), instead of a biometric image. In both cases, the scheme comprises three key steps (see Fig. 5.3):

1. *Feature extraction and encoding*: in the first step, if the input is a biometric sample, a two-dimensional binary feature vector is extracted from it. Then,

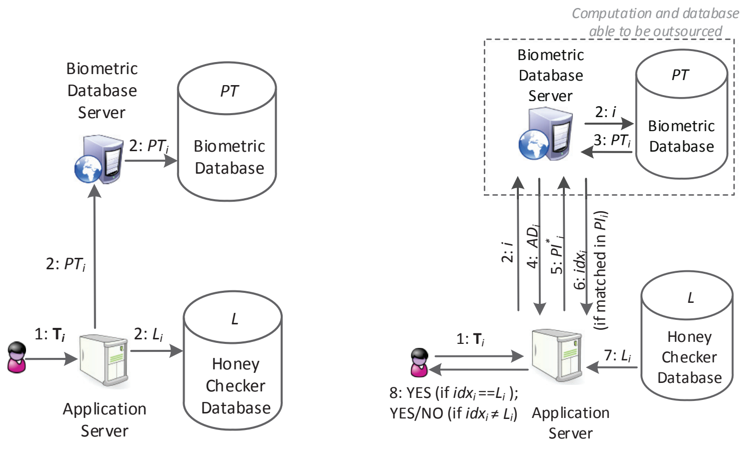


Figure 5.1: Architecture for a honey templates based biometric system (113): enrolment (left) and verification (right)

the resulting vector (or the input random binary vector for the database generation step), $\mathbf{T}_i^j$, is divided into $nBlocks$ blocks of size $nBits \times nWords$ bits.

2. *Structure-preserving feature re-arrangement*: in order to achieve unlinkable templates, we need to dissipate the information of the feature vectors among different blocks, while preserving verification performance. To that end, we first re-group the $nBlocks$ blocks into G groups consisting of B blocks $(nBlocks = G \times B)$, and then re-arrange the features in a structure-preserving manner in two consecutive sub-steps:

   (a) *Row-wise permutation* $(perm)$: for each of the G sets, the rows of the vertical concatenation of all B blocks are permuted. This way, information is diffused between blocks and block-based attacks prevented. While this sub-step prevents a potential loss of discriminative power of resulting feature blocks (neighbourhoods within rows are preserved), the dissipation of rows among groups of blocks significantly improves the information diffusion and prevents block-based attacks.

   (b) *Word-wise shift* $(shift)$: in this second sub-step, circular vertical shifts are applied to words independently for each block. Since vertical shifts are not performed equally for each word in a block, different words may result from identical ones and vice versa, further concealing the number of different words of each original block.

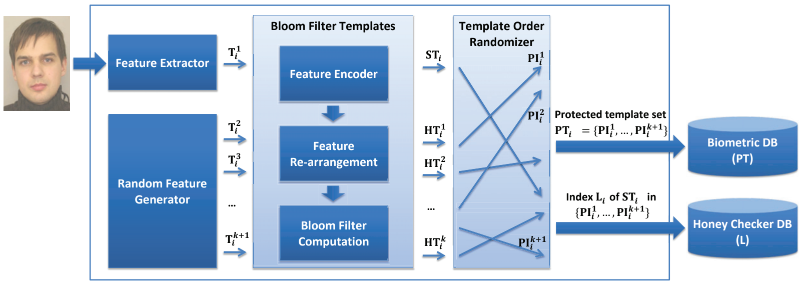


Figure 5.2: Database generation: in order to ensure a higher security, templates and indices will be stored in databases which should be handled by independent systems: the set PT_i in the Biometric Database and the index L_i in the Honey Checker Database

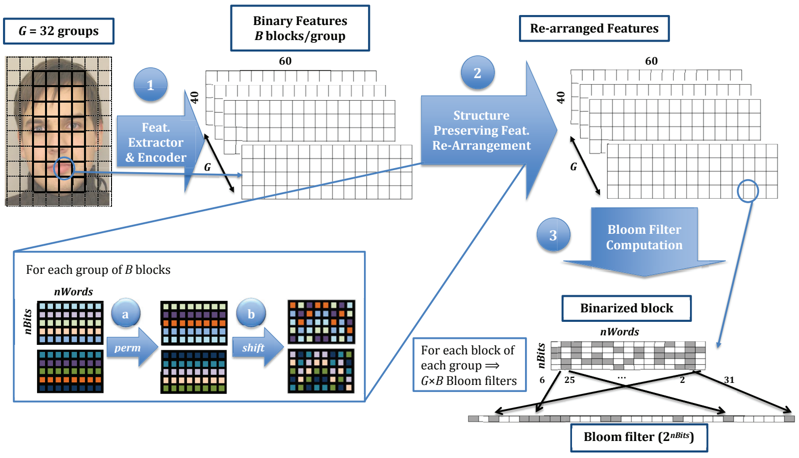

G = 32 groups
Binary Features B blocks/group
Re-arranged Features
Feat. Extractor & Encoder
Structure Preserving Feat. Re-Arrangement
Bloom Filter Computation
For each group of B blocks
perm
shift
Binarized block
For each block of each group ⟹ G×B Bloom filters
Bloom filter (2^nBits)

Figure 5.3: Bloom filter templates generation.

3. *Bloom filter computation*: in the final step, one Bloom filter is computed from each of the $nBlocks$ blocks, such that the final protected template PI consists of $nBlocks$ Bloom filters of size 2^{nBits}.

The distance between two Bloom filter-based templates, PI and PI^*, is defined as the average distance of all pairwise Bloom filters, $\mathbf{b}_i$ and $\mathbf{b}_i^*$, with $i = 1, \ldots, nBlocks$. In order to compute such pairwise distances, since Bloom filters comprise a variable number of ones, their dissimilarity can be efficiently computed as the Hamming Distance between them, normalised by the Hamming Weight of both filters $|\mathbf{b}_i|, |\mathbf{b}_i'|$. Therefore, given two protected templates, their dissimilarity $DS\,(PI, PI^*)$

is estimated as

$$DS\left(PI, PI^*\right) = \frac{1}{nBlocks} \sum_{i=1}^{nBlocks} \frac{|\mathbf{b}_i \oplus \mathbf{b}_i^*|}{|\mathbf{b}_i| + |\mathbf{b}_i^*|} \tag{5.1}$$

where the XOR operator counts the number of disagreeing bits.

For the experiments, we set $nBits = 5$ and $nWords = 15$, the optimal configuration found in (38, 91). For more details on the Bloom filter template computation the reader is referred to (38).

5.3 Experimental Framework: Algorithm III

5.3.1 Database and Verification System

In order to make the present study reproducible and comparable to future research, experiments are carried out on the widely used public BioSecure Multimodal Database[1] (82). The database comprises three datasets captured under different acquisition scenarios, namely: i) Internet Dataset (DS1, captured through the Internet in an unsupervised setup), ii) Desktop Dataset (DS2, captured in an office-like environment with human supervision), and iii) the Mobile Dataset (DS3, acquired on mobile devices with uncontrolled conditions). The DS2 face subset used in this work includes four frontal images of 210 subjects, captured in two time-spaced acquisition sessions (two images per session), with an homogeneous grey background and using a reflex digital camera without flash ($210 \times 4 = 840$ face samples). Eyes were automatically annotated with Neurotechnology VeriLook SDK 4.0[2].

The face verification system that served as baseline for the Bloom filter based BTP scheme is a free implementation within the Bob image processing toolbox[3] (9) of the LGBPHS algorithm (117), a state-of-the-art feature extraction method robust to illumination changes (41). In order to extract the features, face images are convolved with a set of 40 Gabor filters where phase information is discarded, thus leading to 40 Gabor Magnitud Pictures (GMP). Afterwards the LGBP map of each GMP is computed using a Local Binary Pattern (LBP) operator. These maps are further divided into 80 non-overlapping sub-images (only the central $G = 32$ sub-images will be considered), from which histograms are computed and concatenated to form the final representation. For more details, the reader is referred to (117).

[1]Publicly available at http://biosecure.it-sudparis.eu/AB
[2]http://www.neurotechnology.com/verilook.html
[3]http://idiap.github.io/bob/

5.3.2 Experimental protocol

In order to ensure compliance to the requirements established in the ISO/IEC IS 24745 (49) on biometric template protection, assuming that unlinkability is achieved by the Bloom filter based template protection scheme, two different aspects should be analysed: $i)$ whether verification performance degrades with respect to the unprotected biometric system, and $ii)$ the irreversibility of the protected sugar and honey templates. Additionally, the indistinguishability of the honey and sugar templates is studied in order to ensure the fulfilment of requirements of the honey templates scheme. The experimental protocol thus comprises three different steps:

Performance evaluation. Verification performance is analysed for three systems: $i)$ the unprotected face verification system, $ii)$ the Bloom filter based BTP scheme, and $iii)$ the proposed Honey Templates based system. The same protocol will be followed in all cases in order to grant a fair comparison: genuine and imposter scores were computed as the cross comparison between samples of the same or different subjects, respectively. For the Honey Templates system, the probe sample is compared to the full list of sweet templates, taking the minimum Hamming distance as the final score.

Indistinguishability in the Protected Domain. In order to check the indistinguishability of the sugar and honey templates in the protected domain, three different classifiers are applied: Quadratic Discriminant Analysis, Logistic Regression, and Support Vector Machine. We chose these algorithms as most recommended in Machine Learning Classification literature taking into consideration two facts: their accuracy, and the representation of sugar and honey templates (high-dimensional discrete data, separable in two classes).

Indistinguishability and Irreversibility Evaluation. Since an attacker could gain access to the secret permutation and shift keys used in the Bloom filter computation, and try to recover unprotected templates which match the stored references (18), we need to further analyse the indistinguishability of those reconstructed unprotected templates. To that end, in the last set of experiments, we study the indistinguishability and irreversibility of both honey and sugar templates in the unprotected features domain, assuming the eventual attacker has access not only to the stored templates but also to the secret permutation and shift keys. The attacker will first reconstruct the templates and then use the Hamming Distances (HDs) between real ($\mathbf{T}$) and reconstructed ($\mathbf{T}^*$) feature vectors in order to either determine whether the template is sugar or honey (indistinguishability) or in order to check whether he can access the system (i.e., $HD\left(\mathbf{T}_i, \mathbf{T}_i^*\right) < \delta$, where δ is the verification threshold).

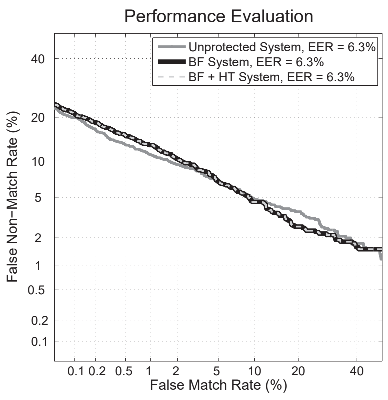


Figure 5.4: **Performance Evaluation**. Detection Error Trade-Off (DET) curves for the baseline unprotected system, the Bloom filter based BTP (BF system) and the proposed scheme (BF + HT system).

In order to reconstruct the templates, given a protected template PI (either honey or sugar), we will reconstruct, in a block-wise manner, a feature vector $\mathbf{T}^*$ as similar as possible to the original input $\mathbf{T}$. To that end, for each Bloom filter $\mathbf{b}$, for each activated index $i = 1, \ldots, |\mathbf{b}|$ the corresponding word $\mathbf{s}_i$ that activated it, is computed. According to the reconstruction approach proposed in (18), the entire feature block is reconstructed as one single word repeated $nWords$ times, where that word represents the bit-wise average of the $|\mathbf{b}|$ reconstructed words $\left(\mathbf{s}_1, \ldots, \mathbf{s}_{|\mathbf{b}|}\right)$. Then, the permutation and circular shift are inverted with the corresponding secret keys.

For further details on the unlinkability and robustness to cross-matching attacks of the protected templates, the reader is referred to (37).

5.4 Experimental Evaluation

5.4.1 Performance Evaluation

In Fig. 5.4, the Detection Error Trade-Off curves for the unprotected system, the Bloom Filter based scheme and the proposed Honey Template system are depicted. Furthermore, we include in Table 5.1 the Equal Error Rates (EER) for the three systems and the accuracy in terms of the False Non-Match Rate (FNMR) for a False Match Rate (FMR) of 0.1%. As it may be observed, there is no performance degradation due to the template protection scheme applied to the original verification system.

In order to confirm that verification performance for the proposed BTP approach

Table 5.1: **Performance Evaluation**: EER and FNMRs at FMR = 0.1% for the baseline unprotected system, the Bloom filter based BTP (BF system) and the proposed scheme (BF + HT system).

	Unprotected	BF System	BF + HT System
EER	6.3	6.3	6.3
FNMR	19.8	21.1	21.1

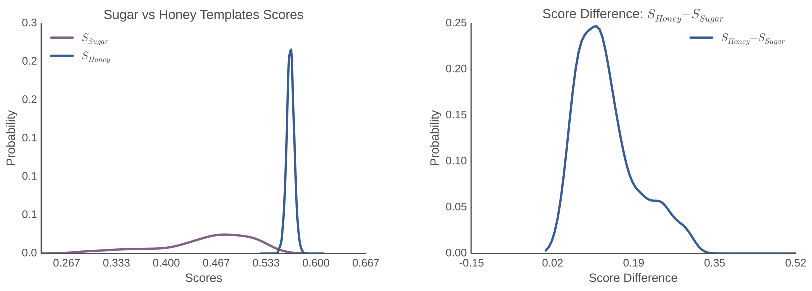


Figure 5.5: **Scores Analysis**. Score distributions for sugar (S_{Sugar}, purple) and honey (blue, S_{Honey}) templates, when compared to the probe template (left). The difference between the $K = 10$ honey scores and their corresponding sugar scores ($S^i_{Honey} - S_{Sugar}$ for $i = 1, \ldots, K$) is depicted on the right.

is exactly the same as the performance obtained using only the Bloom filter based protection scheme, two score distributions are depicted in Fig. 5.5 (left), namely: i) honey scores, obtained comparing the probe template, PI^p, to each stored honey template, PI^{Honey_i}, with $i = 1, \ldots, K$ ($S_{Honey_i} = DS\left(PI^p, PI^{Honey_i}\right)$), and ii) sugar scores, obtained comparing PI^p to the stored sugar reference PI^{Sugar} $\left(S_{Sugar} = DS\left(PI^p, PI^{Sugar}\right)\left(\right)\right.$. As it may be observed, the dissimilarity scores are lower in almost all cases for the sugar templates, as desired: since the final score is computed as the minimum of the sugar and honey scores, in all cases the index retrieved from the Honey Checker Database will be that of the sugar template, causing no false leakage alarms. There is however a small overlap between those distributions. To analyse that overlap, Fig. 5.5 (right) shows the distribution of the differences between honey scores and their corresponding sugar score, $S_{Honey_i} - S_{Sugar}$, for $i = 1, \ldots, K$. All values are positive, proving that honey scores are higher than their corresponding sugar score for all the references stored in the database, and that way confirming that no false leakage alarms were raised.

5.4.2 Indistinguishability in the Protected Domain

Feature Selection. After confirming there is no performance degradation due to the Honey Templates and Bloom filter based protection, we need to check by template classification their indistinguishability in the protected domain. Before classifying, considering the high dimensionality of the templates (32,768 bits), data preprocessing is needed. To achieve this, the most discriminant features will be selected, then follow two different approaches, namely: *filter methods*, or *wrapper methods* (35). The former methods analyse the content of the feature data, then select a subset of features based on their statistical properties. The latter methods divide the feature space into subsets, requiring features to be ordered.

In our particular case, we chose the first approach by applying a filter to the set of bits, which calculates for each feature the sparsity ($\hat{S_f}$) of activated bits. Since we have 840 sugar templates, the features' bit sparsity ranges between 840 (i.e., the bit is activated in all templates) and 0 (i.e., the bit is not activated in any template). With this calculation we try to select the most discriminative ones (those with $\hat{S_f}$ between 450 and 600), creating new patterned sugar and honey templates. We define *patterned protected template*, a template from which a number of features is removed, which satisfy a predefined number of activated bits. The same pattern (output of the filter) will be used on honey templates too (i.e. same features will be removed from all templates). In Fig. 5.6, each bar represents the number of features in each interval of activated bits.

Template Classification. After feature selection, a set of classifiers is trained and tested on sugar and honey templates. Specifically, the chosen classifiers are: *Logistic Regression* (LR); *Support Vector Machine* (SVM); *Quadratic Discriminant Analysis* (QDA)[4]. We divided the patterned 840 sugar templates ($\mathbf{ST}^P$) and 8,400 honey templates ($\mathbf{HT}^P$), in two sets: training, and testing. The former contains the first n-sugar templates, and the n-honey templates (e.g. $n = 100$):

$$\mathrm{C}_{\mathrm{Train}} = \mathbf{ST}_i^P \cup \mathbf{HT}_j^P \tag{5.2}$$

$i = 1, 2, \ldots, n$

$j = 1 * K, 2 * K, \ldots, n * K$

The testing set comprises of the remaining sugar templates and their corresponding

[4]For more on classification algorithms: Bishop, Christopher M. "Pattern Recognition." Machine Learning (2006).

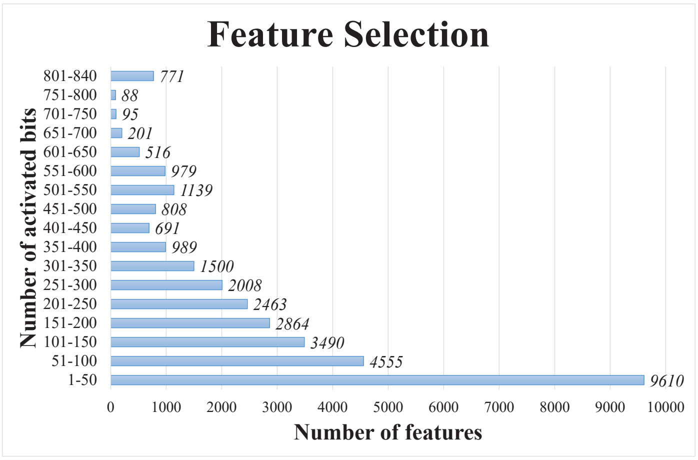


Figure 5.6: **Feature Selection**: graph showing number of features in each interval of activated bits. For example, while 9,610 features contain 1-50 activated bits, 516 features contain 601-650 activated bits. Therefore, the length of the bars also represents the number of features which would be selected should the interval be chosen. Larger number of features yield larger patterned templates, and vice-versa.

honey templates:

$$C_{\text{Test}} = \mathbf{ST}_i^P \cup \mathbf{HT}_j^P \tag{5.3}$$

$i = n + 1, n + 2, \ldots, 840$

$j = (n + 1) * K, (n + 2) * K, \ldots, 840 * K$

After the training phase, accuracy of the classifiers' algorithm was measured and results presented in Table 5.3. Here we add the processing time of each algorithm, calculated via the Classification Learner application, Matlab[5]. Not only algorithms' accuracies were provided, but from the training phase we could conclude that the feature selection process was time efficient. On average the training process lasted $2,5$ seconds, whereas when trying to train original templates (before reducing their features, i.e. with $dim = 32,768$) the process could be executed on QDA only and it took 336 seconds, i.e. feature selection is an important prerequi-

[5]Matlab, Statistics and Machine Learning Toolbox: Classification Learner. http://se.mathworks.com/products/statistics/classification-learner/

Table 5.2: **Template Classification.** Accuracy of classifiers' algorithms, and their processing time (corresponding to patterned templates of 808 features in the interval 401-450).

No.	Algorithm	Accuracy	Training time (sec)
1	Logistic Regression	77,5%	6,10
2	SVM	100%	1,77
3	Quad. Discriminant	99,5%	1,85

Table 5.3: **Template classification.** FMR and FNMR rates of two classification algorithms for different intervals and training sets.

Algorithm	Bin	$\mathrm{FMR^C}$	$\mathrm{FNMR^C}$	$\mathrm{mean_{Diff}}$
LR	$451 - 500$	$0{,}79$	$0{,}58$	$0{,}19$
	$501 - 550$	$0{,}72$	$0{,}57$	$0{,}15$
	$551 - 600$	$0{,}65$	$0{,}59$	$0{,}12$
SVM	$451 - 500$	$0{,}26$	$0{,}28$	$0{,}23$
	$501 - 550$	$0{,}39$	$0{,}45$	$0{,}08$
	$551 - 600$	$0{,}39$	$0{,}33$	$0{,}14$
QDA	$451 - 500$	$0{,}17$	$0{,}16$	$0{,}34$
	$501 - 550$	$0{,}41$	$0{,}40$	$0{,}10$
	$551 - 600$	$0{,}39$	$0{,}38$	$0{,}12$

site in the classification process.

We further proceeded with the testing phase. To measure template classification we used two rates: False Match Rate ($\mathrm{FMR^C}$ - the number of honey templates classified as such), and False Non Match Rate ($\mathrm{FNMR^C}$ - the number of sugar templates not classified as such). Table 5.3, summarizes the test results for every algorithm.

The ideal scenario for a system designer would be a random distribution (a probability for each rate equal to 0.5), under which the classifier would have no way to discriminate sugar from honey templates. In order to measure such difference, we propose the *mean of difference* metric, which merges the two rates in one metric:

Definition 18 (Mean of differences for template indistinguishability) *The mean of differences between* $\mathrm{FMR}^{\mathrm{C}}$ *and* $\mathrm{FNMR}^{\mathrm{C}}$*, and the ideal case* $p = 0.5$*, as:* measures the property of indistinguishability between honey templates as:

$$\mathrm{mean}_{\mathrm{Diff}}\left(\mathrm{FMR}^{\mathrm{C}}, \mathrm{FNMR}^{\mathrm{C}}\right) = \frac{|p - \mathrm{FMR}^{\mathrm{C}}| + |p - \mathrm{FNMR}^{\mathrm{C}}|}{2} \tag{5.4}$$ □

The lower the difference from the ideal case, the better from the system's perspective. The best case in Table 5.3 is $\mathrm{mean}_{\mathrm{Diff}} = 0.08$, a value close enough to zero to conclude that templates are indistinguishable. From an attacker's perspective, these values should be read on the contrary. In fact, he/she can classify the templates to a high probability in the protected domain if a QDA classifier is used (specifically by choosing the interval of 451-500 activated bits). As the table shows, templates can be classified to a certain portion, which leads the way to further improvements on the proposed technique.

5.4.3 Indistinguishability and Irreversibility Evaluation in the Unprotected Domain

In order to further ensure the indistinguishability of the sugar and honey templates, let us assume that the attacker has access to the permutation and shift keys used to compute the Bloom filter based templates and tries to reconstruct the unprotected templates associated with the stored protected references. To that end, taking into account the reconstruction approach proposed in (18), HD-based distributions between the original templates and the ones obtained with the suggested reconstruction methodology are depicted in Fig. 5.7.

As can be observed, impostor HDs between real templates (dashed red) are even lower than the HDs obtained with the reconstructed sugar and honey templates (solid lines). This implies that the Bloom filter based system does not allow an efficient reconstruction of templates close to the original ones, thus granting irreversibility even in the challenging scenario where the impostor has access to the secret permutation and shift keys. Furthermore, the security of the system is enhanced, since access will not be granted to such reconstructed templates.

On the other hand, the sugar and honey templates distributions are very similar (Kullback-Leibler diverge of 0.009), which further confirms the indistinguishability of both kind of templates, even if we try to reconstruct the original feature vectors.

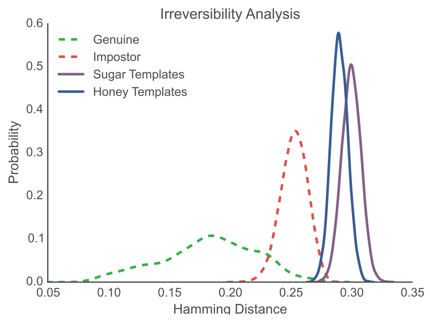


Figure 5.7: **Irreversibility and Indistinguishability Evaluation**. Genuine (dashed green) and impostor (dashed red) HDs between real unprotected templates, compared to the HDs between reconstructed sugar templates (solid purple) and their corresponding real template, and between reconstructed honey templates (solid blue) and their corresponding real template.

5.5 Conclusions

Experiments were carried out on the face corpus of the publicly available BioSecure DS2 multimodal database, using the free Bob Image Processing Toolbox. The performance evaluation showed that verification accuracy was preserved with respect to the original unprotected system. Additionally, a thorough classification analysis was conducted to show to what extent honey and sugar templates are indistinguishable, even in the case an eventual attacker has access also to the secret key used to compute the templates. To this point, in the future a more thorough experimental work needs to be done to improve templates indistinguishability, better honey template construction, feature selection schemes, or other classification algorithms employed. Furthermore, the irreversibility of all sweet templates was confirmed and the security of the system increased: reconstructed templates show HDs to their corresponding original templates larger than real templates belonging to other subjects.

It should be also noted that the complete PT_i associated to each subject comprises $K+1 = 11$ templates (K honey and one sugar) of 4 KB, each of them 98% smaller than the unprotected templates (184 KB). This results in a compact template size of only 44 KB, still 76% smaller than the unprotected templates, with the additional advantages of being unlinkable, irreversible and able to detect stolen templates.

5.6 Brief chapter summary

In the present chapter, we have proposed a new biometric template protection scheme which not only fulfils the requirements established in the ISO/IEC IS

24745 on biometric information protection, but can also detect leakage attacks and initiate appropriate system countermeasures. To that end, the Honey Templates scheme is used for the detection of stolen templates, while irreversible and unlinkable templates based on Bloom filters are utilized for protected biometric verification.

This chapter covers one other implementation of Honey Templates among the Feature Transformation mechanisms. The performance and security evaluations show the soundness of the proposed scheme for facial verification. The benchmark is conducted with the publicly available BioSecure Multimodal DB and the free Bob image processing toolbox, so that research is fully reproducible.

We include the idea of classification framework presented in Chapter 4 and provide the last metrics for the indistinguishability property, mean of differences. In this scheme the recognition performances are improved compared to previous ones, answering again, as Yes, to our research question **RQ1** regarding the implementation to honey templates. In **RQ1.1**, Honey Templates do not deteriorate the system performance when the template protection mechanism are Bloom Filters. For **RQ1.3**, regarding honey templates metrics, with the definitions provided in this chapter we complete and verify research question.

Chapter 6

Biometric Honey Templates adoption with Fuzzy Commitment in a Hybrid Scheme

Technically, to achieve information protection, data integration, or secure communication, there exists a plethora of systems and tools, at the heart of which stand cryptographic mechanisms. Nevertheless, strong cryptographic tools do not always imply strong security and efficient data protection. From its basic principles, the security of a system depends on the security of the weakest link. Similarly, the security of a protection mechanism depends on the security of the most vulnerable part of its implementation. Deception techniques used in Information Security are again applied in a biometric context, offering not only protection mechanisms on the templates, but also are capable of notifying if database leakage and user impersonation occurred.

In this chapter is presented an application of Biometric Honey Templates in a hybrid protection setup, comprised of three components which generates synthetic (honey) templates by preserving the indistinguishability property. While in previous chapters we applied Random Projection and Bloom Filters categorized as *Feature Transformation* schemes where Honey Templates can be adopted, in this chapter we add *Fuzzy Commitment*, one of the protection mechanisms part of the *Biometric Cryptosystem* category.

Considering an attacker having full knowledge of the system, a face verification system and a *testing framework* are implemented. Results from the tests showed that they have a high level of dissimilarity, by preserving the system recognition performance. Another contribution to this work is the *Theorem of Augmentation*. It evaluates the maximum number of honey templates with which a system can be augmented.

6.1 Introduction

As we mentioned in the beginning of this thesis, biometric templates protection mechanisms can be categorized in two main groups: *biometric cryptosystems*, and *feature transformation*. Biometric cryptosystems are also referred to as *helper-data systems* since some public information is stored in the storage module (92). *Feature transformation* schemes apply on the template a transformation function and the comparison process is done on the transformed domain.

Among the different protection mechanisms there are also *hybrid schemes* which combine two or more different techniques. In high-level security scenarios the combination of protection schemes and other deception techniques ensures a better security. This is the protection mechanism implemented in this chapter: a hybrid protection scheme combined with the addition of synthetically generated templates, *honey templates*, making it a two-level security approach.

Apart from the new hybrid scheme approach which improves the *indistinguishability* property of templates, another contribution to this work includes the *Theorem of Augmentation*. It evaluates the maximum number of honey templates with which this scheme can be augmented, which was not evaluated in previous chapters. A security analysis of the mechanism is provided and a framework is built to test the recognition performance of an implemented verification system based on faces.

6.2 Fuzzy Commitment and binarization conditions

Fuzzy Commitment Scheme (FCS) is a cryptographic primitive which is both concealing and binding, being applied in a variety of biometric authentication scenarios. FCS basically works with the help of an Error Correction Code (ECC) which is able to correct a defined number of errors of an encoded binary message, the *codewords*. Messages are hashed and stored in the biometric database, whereas codewords are processed further with the plain feature vectors, forming the templates. For sugar and honey templates different random binary messages are generated, such that they have the same error correction capability.

The Fuzzy Commitment Scheme (FCS), conceptually is what results from the combination between a commitment scheme and fuzzy logic (FL). The latter is a type of logics where variables do not have only one of the two values (True/False), but they can have a range of values as result of approximate reasoning (60). This is the case with biometric samples, which in fact for the same subject and under the same conditions, they present statistically the same feature information. Formally, the commitment scheme transforms a message *m* into the couple *(c, d)* where *c* can be considered as a box or a safe and *d* is the key which opens it. FCS is defined as a tuple of three actions (*Setup*, *Commit*, *Open*) and it follows these steps:

1. The commitment key is generated (*Setup*)

2. The couple *(c, d)* is created for every message *m* from the message space *M* (*Commit*)

3. The message *m'* from *M*, can commit (*Open*).

If we refer to Figure 6.1, the ECC module will take as input a binary vector of length *m* generated by a Pseudo Random Bit Generator. It will encode it and pass the resulting codeword cc_m (for sugar or honey cases) to the function *h*. This is an irreversible function (an XOR operation) which takes as input, in addition to the codewords, the binary templates from the Feature Transformation Block. The result will be a corrupted codeword, which will serve as the final templates ST_i^{FC} and HT_{ij}^{FC}.

In our implementation, as an ECC are used BCH codes, which are classified as *cyclic linear block codes*. They can correct a number t (where $t > 0$) of bit errors (14). Their input is considered as the message or secret value s_{ij}. The values s_{ij} are encoded and the codewords c_{ij} are generated. Formally, for a set $\mathbb{C}$ of codewords *c*, and a given number of errors t_ϑ , having as threshold *t*, the error correcting code can correct a noisy codeword c_ϑ and associate it with *c*, if the error t_ϑ is lower than the defined error *t*.

$$\begin{gathered}\forall\, c, c_\vartheta \in \mathbb{C}, \text{and}\, \forall\, t, t_\vartheta \in \mathbb{N} : \\ \text{if } t_\vartheta \leq t, \text{ then } f(c_\vartheta) = f(c)\end{gathered} \tag{6.1}$$

In Eq. (6.1) *f* is an associative encoding function. In our implementation is used a BCH encoder with the narrow-sense generator polynomial (75). The *s*-value vectors are Galois elements in Galois Field *GF(2)*. The involved parameters are:

q: defines the vector space of the codewords. This space is the Galois field.

n: codeword length (dimensionality of the codewords *c*)

m: the *s*-values length. It is used as input of the ECC component of the system and every *s* will be associated with a codeword *c*.

t: number of correcting errors.

There is a relation between the parameter *q* and the codeword length *n*:

Table 6.1: Excerpt from the accepted combinations between m (length of s-values) and t (number of error corrections) for a codeword length $n = 1023$.

No.	Length (m)	Errors (t)
...	...	...
97	86	183
98	76	187
99	66	189
100	**56**	**191**
101	46	219
102	36	223
103	26	239
...	...	...

$$n = 2^q - 1 \tag{6.2}$$

For the vector space parameter is chosen the value $q = 10$. This implies that the vectors' length will be $n = 1023$. In Table 6.1 are shown some of the combinations between the codeword length *n*, the *s*-values length *m*, and the number of errors *t*. For example, if the length of the message to be encoded is $m = 56$, then the relative number of correcting errors is $t = 191$.

As defined in Eq. (6.1), the maximum number of corrupted bits should be lower than the error correction capability *t*. To achieve this, the *h*-function should be controlled to generate a generic fuzzy commitment template T^{FC} such that the difference (Hamming Distance-HD) with the codewords cc_m is less than *t*:

$$HD(cc_m, T^{FC}) \leq t \tag{6.3}$$

To satisfy this, the generic templates T^{FT} elements should be influential enough on the fuzzy commitment process. There are two possibilities: T^{FT} element values have to comply the codeword pattern distribution of bits '*1*'; or, they can be sparse enough so that the resulting bit values from the XOR operation will not be influenced. The pattern distribution of binary representation of the feature transformation templates is not a valid operation in the actual context, whereas the second option is achievable by dynamically lowering the threshold value ϑ of buffer *b* til it reaches the desired number Ŝ of bits '*1*' in the binary vector $V^b = (v_1^b, v_2^b, ?, v_n^b)$.

$$\hat{S} = \sum_{i \in \{1,2,...,n\}} (v_i^b) \tag{6.4}$$

To quantify the desired threshold value, we give the following claim which can be considered as a general binarization technique of real value vectors, resulting in a predefined noise *t*.

The threshold ϑ for the binarization of a real-value vector $V(v_1, v_2, ..., v_n)$, can be dynamically increased by a value ϵ til the binary vector $V^b = (v_1^b, v_2^b, ..., v_n^b)$ reaches a sparsity level t*, such that*:

$$\delta = \min_{i \in 1,..,n} \{v_i\} + \epsilon \tag{6.5}$$

Threshold ϑ will be used by buffer *b*, to generate the new feature transformation templates and serve as "noise" for the fuzzy commitment templates.

6.3 Theorem of Augmentation

The last component of the hybrid scheme is the Fuzzy Commitment protection block, needed to evaluate the number of honey templates a biometric system can be augmented with. We provide the following:

Theorem of Augmentation. *In a BCH code, increasing the cardinality of the codeword space* C *by* k*-times, will not influence the chosen message length* m*, which has a difference of values Δ_{mi+1} with the successive m_{i+1}, as long as*:

$$k \leq 2^{\frac{\Delta m_{i+1}}{2}} \tag{6.6}$$

PROOF The cardinality of the new set *C'* which is *k*-times incremented for each codeword, is:

$$|C'| = 2^{m_i} * k \tag{6.7}$$

$$|C'| = 2^{m_i} * 2^{\lceil log_2 k \rceil} \tag{6.8}$$

$$|C'| = 2^{m_i + \lceil log_2 k \rceil} \tag{6.9}$$

For *C'* we see that the new parameter *m'*, the new length of *s*-values, is:

$$m' = m_i + \lceil log_2 k \rceil \tag{6.10}$$

The values of *m'* can be greater than the predecessor value m_i. Since this value should not approximate the successive value, it should be less than half the distance from the value m_{i+1}:

$$m_i \leq m' \leq m_i + \frac{m_{i+1} - m_i}{2} \tag{6.11}$$

$$m_i \leq m' \leq \frac{m_i + m_{i+1}}{2} \tag{6.12}$$

Substitute *m'*:

$$m_i \leq m_i + \lceil log_2 k \rceil \leq (m_i + m_{i+1})/2 \tag{6.13}$$

Simplify:

$$0 \leq \lceil log_2 k \rceil \leq \frac{m_{i+1} - m_i}{2} \tag{6.14}$$

From ceiling function property, if $\lceil x \rceil \leq n$, then $x \leq n$:

$$log_2 k \leq \frac{m_{i+1} - m_i}{2} \tag{6.15}$$

Yields:

$$k \leq 2^{\frac{m_{i+1} - m_i}{2}} \tag{6.16}$$

$$k \leq 2^{\frac{\Delta m_{i+1}}{2}} \tag{6.17}$$

□ ■

Example 1.
Let's suppose *m=56*, *m'=66* and *k=15*. We'll have $\lceil log_2 k \rceil = 4$, resulting in *m'=60*. The value of *m'* is slightly larger than the suggested value of *m*. From

Table 6.1 we can also see that the nearest acceptable value to it is still *m=56*. If we consider in this case $\Delta m_{i+1} = 10$, then the maximum number of honey templates that can be applied to this system is equal to *32*. As a result, from Theorem of Augmentation we not only prove the relation between parameters, but also quantify the maximum number of honey templates the system is able to handle.

6.4 Hybrid scheme description for template protection

After feature extraction, the hybrid protection scheme is built by cascading: a feature transformation mechanism; and then a fuzzy commitment scheme (FCS) (63), as in Fig. 6.1.

Feature extraction. During enrolment or verification, plain feature vectors SB_i are extracted from face images using Principal Component Analysis (PCA) technique. By definition, PCA is a mathematical procedure that converts a set of correlated variables to a set of uncorrelated variables (53). It finds a reduced set of vectors which could serve as a basis from which all other vectors are generated as their linear combination. Face images are seen from the PCA technique under the same point of view, with the aim to find the most informative base vectors (28), called *eigenfaces*.

Random Projection. To distribute the plain features SB_i from the PCA module in a higher dimensionality, the *random projection* mechanism is used (112). It multiplies the plain feature vectors with random matrices AM_i to increase the length of the feature vector representation. This will make the scheme more adaptive in terms of feature length and improves the security of the scheme.

Binarization. The outputs SB_i^{RP} of the *feature transformation scheme* (*A*) will serve as input for the Fuzzy commitment scheme (*B*), but they first need to be binarized. For this reason the buffer *b* will convert the random projection templates SB_i^{RP} to their respective binary representation SB_i^{FT} according to a threshold parameter. The parameter ϑ can be dynamically incremented till the binary vector reaches a sparsity level *t* (amount of bits *1*).

Honey templates generation. Honey templates are created similarly. For every template SB_i are generated k-random real-value vectors following the idea implemented in (73). They are multiplied with the random matrices AM_{ij} and the result HB_{ij}^{RP} is binarized according to the threshold ϑ.

Fuzzy Commitment. The last component of the hybrid protection scheme is the Fuzzy Commitment (FCS),

The three components are formally represented by the functions *f*, *g*, and *h*. Each

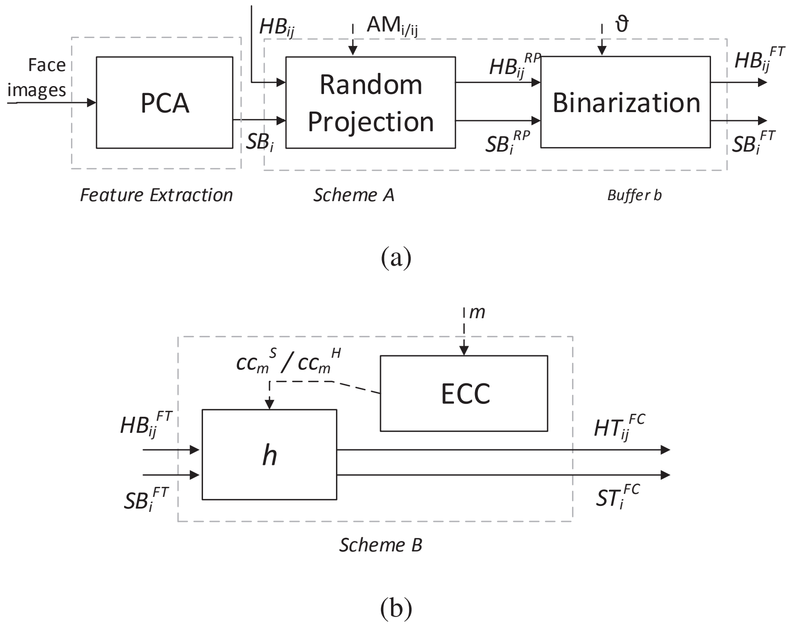


Figure 6.1: Hybrid scheme for sugar and honey template protection: (a) Generation of sugar templates (SB^{FT}) and honey templates (HB^{FT}) after Feature Transformation (Scheme A and buffer b); (b) Generation of sugar templates (ST^{FC}) and honey templates (HT^{FC}) after Fuzzy Commitment (Scheme B).

component takes as input two types of vectorial data (honey-H and sugar-S) and a key. Each component, T^f, T^g, and T^h is defined in the following equations accordingly:

$$T^f = f(B_{ij}, k^f) = B_{ij} \times k^f \tag{6.18}$$

$$T^g = g(T^f, k^g) = \begin{cases} 1 & \text{if } T^f \leq k^g \\ 0 & \text{if } T^f > k^g \end{cases} \tag{6.19}$$

$$T^h = h(T^g, k^h) = T^g \oplus k^h \tag{6.20}$$

In Equations (6.18) - (6.20), B_{ij} is a biometric feature vector and the key k^f is the random matrix AM_{ij}; the key k^g is a threshold variable; the final template T^h is the result of the compound function, whereas the key of the scheme is the set $K = \{k^f, k^g, k^h\}$ with k^h the output of ECC. In our work we have adopted such a scheme by *chaining* two mechanisms: a feature transformation mechanism;

and then a fuzzy commitment scheme (FCS). Between the two is added a buffer module to adopt the input
output between the two mechanisms.

Since we have an augmented system with honey templates, this needs a new evaluation of the key space *C*. The dimensionality of the s-values, is not algebraically defined but depending on the actual implementation, can be decided how much error correction capabilities the scheme allows (75). First let's define some symbols:

C': the augmented set of codewords.

m_i: the length of the s-values in *C*.

m_{i+1}: the length of the subsequent s-value.

Δm_{i+1}: the difference between m_{i+1} and m_i.

m': the length of the s-values in *C'*.

6.5 Framework description of template testing in a face verification honey-based system: Algorithm IV

6.5.1 Database properties

For the experiments it is used an excerpt of the *faces94* database (31) comprised of *70* subjects with *20* samples. The variation of images is minor, but they have considerable face expression changes. The whole dataset is processed as follows: changed the size from 180×200 to 92×112; lowered the bit-depth from *24* to *8* converting images to grayscale; and changed the resolution to 96 dpi. Every sample was associated with a number of $k = 15$ honey templates, having in total 16 sweet templates each.

Both types of feature vectors SB_i and HB_i have a dimensionality of $dim(PCA) = 20$ real-value feature vectors. Random Projection module increases the dimensionality of the vectors to $dim(RP) = 1023$ fed by the auxiliary matrix AM_i which is a random real-value matrix of dimensions 20×1023. The output from the Random Projection will be binarized as much as to be considered corrupted templates to be XORed with the ECC codewords resulting from the coding of the secret values *s*.

In the biometric database are going to be stored: *i*) the template after fuzzy commitment; and *ii*) the hashed secret values *s* (or Auxiliary Data AD). During verification the new feature will be transformed to a new template using the same *s*-value. Both templates are then decoded and if the hash values are equal, the user is authorized.

6.5.2 Testing framework

To prove the soundness of the proposed protection mechanism we have applied *classification*, and *performance* tests. Classification tests are applied on honey and sugar templates, whereas performance tests in cases are applied on sugar templates only, and in others in honey templates too for comparison reasons. These tests are applied on different points of the system, specifically after creation of different templates stages: (1) PCA vectors (*Feature Extraction*); (2) Random Projection templates (*Scheme A*); (3) Binarized random projection templates (*Buffer b*); (4) Fuzzy commitment templates (*Scheme B*).

In every level different supplementary tests are added taking into consideration that a possible attacker may have a plethora of tools to distinguish sugar templates from honey templates, at every level of its creation. A classifier (in our case Support Vector Machines, SVM) is trained by labelling sugar templates as *'1'* and those honey ones as *'0'*. We chose SVM because of its accuracy in high-dimensional data. While its training is ready, the SVM classifier *C* can be used to test a sugar template to see if it can be classified as a honey one (labelled as '0').

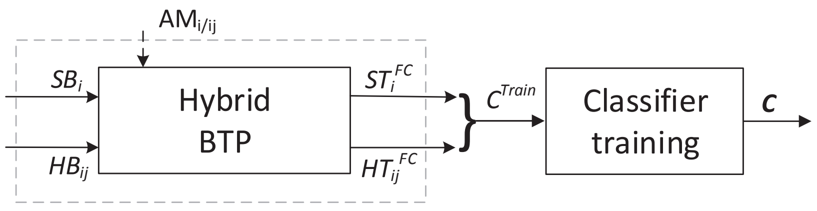


Figure 6.2: Classifier training using the set CTrain from labeled ground-truth sugar protected templates (SB_i^{FT} labeled as *1*) and honey protected templates (HB_{ij}^{FT} labeled as *0*).

In Figure 6.2 is shown the training phase. The C^{Train} set, is comprised of two subsets from sugar and honey templates: ST_i^{FC} and HT_{ij}^{FC}. For these two subsets there are two considerations:

(a) **size** q, meaning the cardinality of C^{Train} compared to the whole set of sugar and honey templates;

(b) **position** r, meaning which samples will be chosen from sugar and honey templates to create C^{Train}.

For the different sizes, the sets are designed as:

$$SVM_q^{Train} = ST_{1:q} \cup HT_{1:q,j} \tag{6.21}$$

$$SVM_q^{Test} = ST_{(q+1):m} \cup HT_{q+1:m,j} \tag{6.22}$$

where: $q = \{100, 200, ..., 1200\}$ and $j = 1$.

Here the index i represents the number of samples. In our database we have $m = 70 subjects \times 20 samples = 1400 samples$, and from honey templates we pick the first template. For the classification test in different slots *r*, the sets are designed as follows:

$$SVM_r^{Train} = ST_{r*i+1:r*(i+1)} \cup HT_{r*i+1:r*(i+1),j} \tag{6.23}$$

$$SVM_r^{Test} = ST_{r*(i+1)+1:m} \cup HT_{r*(i+1)+1:m,j} \tag{6.24}$$

where: $r = \{100, 200, ..., 600\}$, $i = \{1, 2, ..., 6\}$, and $j = 1$.

Taking into consideration that in every step we have randomized processes, the set of tests is repeated in 5 rounds. The four levels are described below and for each level is listed the type of undergoing test. The whole framework is presented schematically in Fig. 6.3.

Level I. Plain feature vectors

Performance Test no. 1: applied on simple feature vectors *SB*, generated from the PCA technique.

Level II. Random Projection

Performance Test no. 2: after random projection of plain feature vectors. This test is applied on sugar templates only (SB^{RP}).

Performance Test no. 3: applied on templates created after the random projection process (SB^{RP} and HB^{RP}).

Level III. Feature Transformation

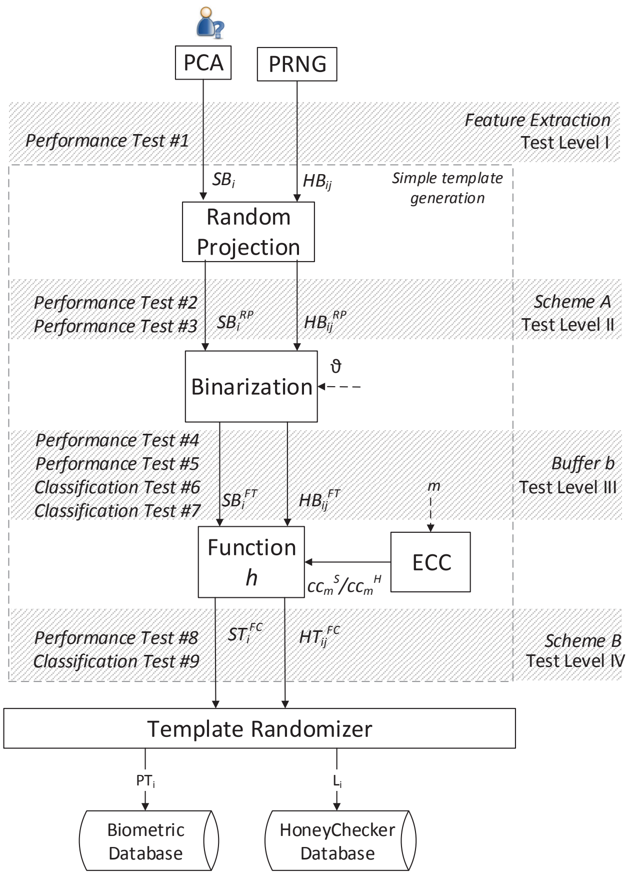


Figure 6.3: Classification and performance test framework for a hybrid honey-based protection mechanism.

Performance Test no. 4: this test is applied on sugar templates after binarization (SB^{FT}).

Performance Test no. 5: applied on both sets of templates to test the performance of the Feature Transformation Scheme augmented with honey templates (SB^{FT} and HB^{FT}).

Classification Test no. 6: this test is applied on different sizes of training and testing sets of the classifier. In this case, 11 tests on different classifier training and testing sets were conducted.

Classification Test no. 7: templates are classified for different slots of training and testing sets of the classifier. For our case, 5 different slots are considered.

Level IV. Fuzzy Commitment

Performance Test no. 8: applied on sugar and honey templates generated after the Fuzzy Commitment templates.

Classification Test no. 9: applied for different values of the length *m* of the secret values cc_m^S and cc_m^H. Depending on this value, 34 subtests are conducted.

6.6 Tests results for classification and system performance

Classification tests metrics. From all the classification tests we have measured the SVM classifier False Match Rate (FMR^{SVM}) and False Non Match Rate ($FNMR^{SVM}$). To measure the undistinguishability property we have used a metrics: the mean of the absolute value of the difference between the FMR^{SVM} and $FNMR^{SVM}$, and the ideal case $p = 0.5$. From the system perspective, the lower the value of mean difference, the better.

$$\begin{aligned} meanDiff(FMR^{SVM}, FNMR^{SVM}) = \\ = \frac{\mid p - FMR^{SVM} \mid + \mid p - FNMR^{SVM}}{2} \end{aligned} \tag{6.25}$$

Performance tests protocol. In the plain template case, we use the following testing protocol: for all 70 subjects in the testing database we cross compare the 20 samples from each same subject resulting in $13,300$ genuine comparison scores; and for impostor comparison score the first sample of the 70 subjects was compared, resulting in $2,415$ scores. For the honey-based systems the probe sample is compared to the whole set of sweet templates and the highest score is the selected one. As a metrics for the system performance we have used the Equal Error Rate (EER).

Tests results and analysis. The EER on plain PCA feature vectors is equal to: $EER^{PCA} = 0.0578$. In Table 6.2, are presented the performance test results after Random Projection and the average EER values are calculated. It can be observed that RP will bring no performance degradation in the system, whereas adding honey templates will increase the EER. Comparing performances after binarization, it can be seen that they do not differ, meaning also a higher indistinguishability between the two (Table 6.2).

Finally, the overall performance of the hybrid scheme is $EER = 0.0776$, which proofs the applicability of the system.

Table 6.2: Performance results after Random Projection (Level II, left side), and after Binarization (Level III, right side) for 5 rounds.

No.	**Without HT** (*Test no. 2*)	**With HT** (*Test no. 3*)	**Without HT** (*Test no. 4*)	**With HT** (*Test no. 5*)
1	0.0588	0,0979	0.0652	0,0652
2	0.0584	0,1028	0.0654	0,0654
3	0.0572	0,0987	0.0630	0,0632
4	0.0574	0,0949	0.0599	0,0618
5	0.0577	0,0969	0.0623	0,0636
Avg. EER	0.0579	0,0982	0.0632	0,0636

6.7 Security analysis and attack scenarios of the hybrid honey-based protection scheme

6.7.1 Scheme components security

Designing a secure biometric template protection scheme requires withstanding by a security model. Here the goals of security must be stated clearly and possible attacks examined. In our approach, after describing how the honey-based scheme is designed we evaluate it according to the model. There are *unconditional*, *provable*, and *computational* security. **Unconditional security** is also called *perfect secrecy* and the assumption is that the attacker has unlimited computational power. A system is considered perfectly secure if data cannot be revealed after they are protected. In **provable security** the protection mechanism can be as difficult to break as well-known difficult problems. It is also called *practical security*. And in **computational security** is assumed that the needed computational amount can be too large to break a protection mechanism. It is usually measured as 2^n.

We split the security analysis of our scheme in two main parts, one related to the biometric recognition requirements, and one related to the security of each component separately (*f*, *g*, and *h*) and the compounded function $(f \circ g \circ h)$. They are both important, even though in the literature can be frequently seen as two separate problems. The security requirement are as below:

1. Irreversibility. The main requirement that a biometric template protection scheme has, is to be *irreversible*. According to (49) the template resulting from applying a protection scheme on biometric feature vectors should not reveal any information about them. Indeed in our case, this property is satisfied since all three functions are standard biometric irreversible schemes.

2. Function *f* security. Function f (random projection) increases the template length from 20 real-value vectors, to 1023 real-value vectors. The key for this transformation is a 20 x 1023 real-value matrix. Since this is not a square matrix, it does not exist its inverse.

3. Function *g* security. In g the template is binarized. Even if an attacker knows the key k^g it is impossible to reveal the template real-value representation.

4. Function *h* security. The length of the final template is equal to 1023 bits, which is high enough for the computational security model. The length of the key k^h prior to hashing is 203 bits which is again a high security standard. The key k^h is kept secure by hashing it.

5. Template indistinguishability. This is the main requirement for a honey-based protection mechanism. To assure it, the honey templates design should preserve this property in every component, i.e. $mean_{Diff}$ should be ideally 0.

Classification tests of Level II and III are shown in Table 9.2. The best result is seen for test no. 9, which has a lower mean difference. There are some comments to be added here. The best slot size (test no. 6) was achieved for $q = 400$, the best slot position (test no. 7) was achieved for $r = 301 \div 600$, and the best value for message length (test no.9) was achieved for $m = 203$.

Table 6.3: Classification rates for tests 6, 7, and 9. (Level II and III)

Test no.	FMR^{SVM}	$FNMR^{SVM}$	$mean_{Diff}$
6	0.1027	0.6850	0.2912
7	0.1036	0.6217	0.2591
9	0.5142	0.4967	0.0123

Tests 6 and 7 results show a need for improvement.

6.7.2 Attack scenarios

After Distinguishability test requirements, the second step is analysis of attack scenarios. There are different types of attackers defined on their level of knowledge on the system design, and the sophistication of tools they possess. In a generic biometric systems, attackers can create fake biometrics, resubmit previous biometric signals, inject features via a Trojan, inject a decision score in the communication channel, or attack the template storage. Considering honey templates, some of these threats diminish. In Table 6.4 are listed actions of attackers on a honey-based biometric system. As it can be observed, the honey templates approach serves as a solution not only to the specific block we have considered (database leakage) but its benefits are extended to lower other threats too. But, adding another component, the HoneyChecker Database, will also add other threats, which are included in the same Table.

Table 6.4: Attacker options and actions in a biometric system when honey templates are implemented

No.	Threat	Attacker actions
T1	Fake biometrics	The attacker can decide to reconstruct the whole set of *sweet-faces*. But the feature indistinguishability would not let him pick up the genuine artifact.
T2	Resubmission of a previous biometric signal	If the attacker is trying to resubmit a template from the set of sweet-templates, and if it is a honey one, it will be detected. We can say that the threat from Replay attack and the Hill-Climbing attack diminishes.
T3	Feature extractor threat	The attacker who possesses the whole set of sweet-templates has to command the malicious program (a Trojan) to generate features, but coming from which template?
T4	Communication channel attack	The attacker accesses the communication channel and can inject a score. In this case time processing becomes very important.
T5	Storage attack	The attacker can steal all the sweet-templates but he still does not know the correct index of the genuine template.
T6	*Honey-checker database attack*	*If the attacker possesses the indices of sugar templates stored in the Honey-checker database, he can perform other attacks such as masquerade attack. This is a 'win' situation for the attacker.*
T7	*Communication channel attack between Honey-checker and Storage*	*During verification, the injection of an index in the communication channel is of no value if the attacker has no information on the indices.*

The worst case scenario is an attacker having full knowledge on the system design, who has previously collected sets of sweet templates and now wants to distinguish newly presented sets of templates. He only has no knowledge on the keys, which brings us at the condition of Shannon's maxim, according to which "*a cryp-*

tosystem should be secure even if everything about the system, except the key, is public knowledge". Among the variety of these attacks the ones which might effect our approach are the categories of: **Exhaustive search**, **Cryptanalysis**, and **Key-based** attacks.

In **Exhaustive search attack** an attacker will try to decipher a template with every possible resource, (e.g. key space, or couples of features vectors and protected templates). This is a very straight-forward attack but if the space cardinality is big enough (e.g. more than 100 key bits) this category of attacks is not feasible. The most known in this category is *Brute Force attack*. Depending on the length n of the key, the maximum number of tries will be 2^n. *Dictionary attack* is similar to *Brute Force Attacks*, but now the attacker has a list (dictionary) of associated feature vectors and protected templates.

In **Cryptanalysis attacks** depending on what the attacker owns or knows, there is a variety of ways for him to break the system. Different kinds of attack fall under this category in which we have made an analogy between *protected templates* and *ciphertext*. E.g. as in *Ciphertext Only attack* the attacker possesses the protected template and try to discover fixed properties. Or in *Known-Plaintext attack* the attacker knows the feature vector and the protected template and analyses them to decipher future templates. And in **Chosen-Plaintext attack** the attacker has a feature vector of his choice and is able to obtain the corresponding set of protected templates. In this case it is an attack during Enrolment.

Key-based attacks are a category where the adversary tries to find relations between different keys by observation, or look for similar representations. In the later case the problem of *Weak keys* is presented. If the protection mechanism is poor, the key can be recovered. In general weak keys are a very small portion of the whole key space, making this kind of attack does not probably rise security problems. In *Related-key attack* the attacker might observe that keys might have mathematical relation. Or, in some other case, the keys might have the same bit values in certain intervals.

6.8 Conclusions

This approach can be applicable in high security scenarios in verification mode. Considering nowadays memory costs, the augmentation process will *not affect* expenses. Each user having 16 templates of 128 Bytes each, needs approximately 3.5 kB of space allocation. The processing speed is *fast* and mostly determined by the FCS encoder. For each user enrolment the generation of 16 templates takes approximately 3 seconds in a laboratory setup.

Another contribution to this work include *Theorem of Augmentation*. It evaluates the maximum number of honey templates with which a system can be augmented. A security analysis of the mechanism was provided, considering each component separately and the scheme as a whole.

As future work, there are still many parts of the whole system to be implemented, (the Randomization block), and more on databases design, e.g. the HoneyChecker database. Regarding the recognition accuracy, it can be further improved, as well as the indistinguishability requirement in the first two components of the hybrid scheme. Security evaluation can be extended to more detailed scenarios and crypt-analysis be more profound in some specific security requirements.

6.9 Brief chapter summary

In this chapter is presented an application of Biometric Honey Templates in a hybrid protection setup, comprised of three components: Random Projection, Binarization, and Fuzzy Commitment. The solution includes the generation of honey templates that preserves the indistinguishability property. System performance was evaluated after each block. *Fuzzy Commitment* is one of the protection mechanisms part of the *Biometric Cryptosystem* category.

Considering an attacker having full knowledge of the system, a face verification system and a *testing framework* are implemented. Results from the tests show that they have a high level of dissimilarity, by preserving the system recognition performance. Another contribution to this work is the *Theorem of Augmentation*. It evaluates the maximum number of honey templates with which a system can be augmented. As a result, from Theorem of Augmentation we not only prove the relation between parameters, but also quantify the maximum number of honey templates the system is able to handle. This chapter completes research question **RQ1**, as we have now applied honey templates in different template protection schemes. Regarding **RQ1.4**, we answer as Yes, there is an upper limit in the number of Honey Templates that can be augmented in this scheme.

Chapter 7

Automatic and human classifiers comparison

7.1 Introduction

In a biometric system, during registration, verification, or identification, the subject has to present his/her biometric characteristic (face, fingerprint, signature, iris, voice, etc.) and the system must extract a set of features from this content. These features, used for comparison to recognize or identify a subject, are protected and stored in a data storage system (such as a database) in the form of a template. By definition, *a template is a set of stored biometric features comparable directly to probe biometric features* (5). It represents sensitive information and should be protected without losing the capability to identify or verify a person (97). This is important since templates, if not properly protected, can reveal much of the subject's information. An imposter could use this information in another application and pretend to be who he/she is not. Templates can also be substituted if the storage system is not secure, or they can be modified when they pass through communication channels. In this sense, treating template protection means that other potential attacks on the system must be considered.

To protect the template, we suggest that multi-level security should be applied on biometric databases. As proposed in (114) and implemented in (72), a trap-based mechanism can be added to the biometric template protection scheme to lead attackers astray. In both works, genuine biometric templates were hidden among a set of synthetic templates. The main property of these decoy elements to be satisfied is indistinguishability, i.e. it should be impossible for an attacker to differentiate a real template from a synthetic (honey) template, even if he uses trained classifiers, or trained human subjects, after reconstructing pre-images of cracked templates. Ideally, 'impossible' means that the probability to distinguish a sugar from a honey template is p=0.5.

In this research chapter, we present a new template protection scheme where template blocks are generated based on feature vector random projection, and then concatenated to create the full template. To prove their indistinguishability property, we assumed that the attacker having full knowledge on the system. Human judges were asked to classify them based on their perception and the results are compared to an SVM classifier. Finally, we evaluate the indistinguishability property of the proposed template protection scheme based on the classification result.

7.2 Honey-based biometric system

Using biometrics as a means to verify or identify a subject has also raised a lot of concerns mainly because an adversary attack can impose real threats to the overall system. To overcome the attacks, there have been many efforts from the scientific community in designing, implementing and testing different biometric template protection schemes. One of the promising approach is using a deception mechanism.

In information security, deception mechanisms are used to lead attackers astray from sensitive data. They use decoy elements to deceive internal threats or external unauthorized intruders. One good representative example are honeypots, which are network machines used to distract adversaries from more important machines; or honeyfiles which disguise real content among synthetic files. In fact, the implementation of honey objects in the biometric context was first introduced in (114), where an adoption of *honeywords* (57) to biometric templates was presented and further implemented in (72) on faces. According to this idea, if an adversary steals the password file of a user, he can easily access his system account, but if in the password file there are k-passwords he can't detect the real password from the 'honeywords' (synthetic passwords). One of the main challenges of this method was the fact that it was difficult to randomly generate honeywords which would look like real user passwords.

7.2.1 Architecture design

In the adoption of the *honeywords* concept, we have used and defined the following terms:

Honey templates: synthetic templates placed in the templates file of an authentication server to deceive attackers.

Sugar templates: real biometric template placed among the honey templates in the subject's file in the authentication server.

Sweet templates: the set of biometric templates containing the real biometric template of a subject and the generated honey-templates.

Template honey-checker: a system that checks if a template submitted by a subject is a sugar template or a honey template.

If an adversary possesses the template's file and cracks the protection mechanism, it should be hard for him to ascertain the user's real template when he attempts to impersonate this user. So, in some probability, he will submit a honey template. In this case an alarm may be set off, or specific policy rules will follow, taking into consideration that the system detected an information leakage.

During *enrolment*: a subject presents his/her biometric characteristic and after feature extraction and protection scheme application, the sugar template ST_i is stored together with a set of honey templates HT_{ij}. The honey templates are generated following specific criteria, depending on the actual biometric trait feature representation. As a result we have:

$$PT_i = \{ST_i, HT_{i1}, HT_{i2}, ..., HT_{ik}\} \tag{7.1}$$

The sugar template is randomly positioned in the memory space allocated to user i and its index L_i is stored in the Honey Checker Database. The whole set of sweet templates PT_i is stored in the Biometric Database.

During *verification*: for user i will be generated a new template(in biometrics the newly presented template is not exactly the same as the stored template during enrolment). It will be compared with the whole set of templates and a best score will determine the index of the template it matched. This index will be compared with the L_i index it retrieves from the Honey Checker Database. If they are the same, the user is verified, and if not, then the system will consider that a honey template is being used and a probable information leakage has happened.

The two databases are proposed to be independent. This independence improves the whole system architecture security, since a possible intruder must compromise both to obtain the identity information of a subject. Therefore, honey-templates inherit the benefits of distributed or threshold cryptography. Its idea is to protect information by distributing it among a cluster of cooperating computers (1). As in the case of honeywords, they can be placed in separate domains and can run different operating systems.

In fact, if honey templates are properly implemented and used in biometric systems, they can lower the threats. Augmenting a biometric system with honey templates ensures a two-level security.

7.3 Attack scenarios on Honey-based system

Regardless the implemented security mechanisms of a system, attackers always try to crack them to get access to the system data. There may be different types of attackers defined on their level of knowledge on the system design, and the sophistication of tools they possess. In this work we consider that the Attacker is aware of: (1) the biometric system design; (2) database structure and of the fact that it is using honey templates; (3) algorithm used to generate protected honey and sugar templates. He has previously gathered sets of sweet templates but he can not distinguish them (since they sugar templates are randomly stored among honey templates). After getting the file of subject u_i, the attacker will try to run a classification test and then make a decision. This is considered a *high level attacker*, and his perspective will be further elaborated.

7.3.1 Attacker's perspective on existing honey-based biometric system

The main challenge for a honey based system is to generate honey objects, in our case synthetic face templates, which cannot be distinguished from real templates. The protection mechanism we have adopted on the faces feature vectors is the plain-feature-defined sub-set selection borrowed from the idea in (115). Recognition performance degradation can be anticipated from the adoption of BTPS as discussed in (97). This degradation caused by adding honey templates can be easily seen as well in our case.

To make a quick proof-of-concept evaluation of the proposed honey template based biometric system, we created two small scale face databases denoted as DBaux and DBtst representing an auxiliary database (for purposes of PCA training and the construction of ADi used by BTPS) and a testing database, respectively. DBtst is formed by 40 faces with 10 samples each as a sub-set of the ORL face database (81). To be fair in performance evaluation, DBaux is formed by 40 faces from public websites with 1 sample for each face. All these 40 samples in DBaux were cropped and normalized in their size to the same specification of those ORL samples (i.e., 92112 pixels, 256 gray scales). If their construction satisfies the indistinguishability property, honey templates can be implemented for any biometric characteristic. In this section, we present a detailed construction of face templates based on the Principal Component Analysis (PCA) algorithm.

7.3.2 PCA for feature vectors generation

PCA by definition is a mathematical procedure that uses an orthogonal transformation to convert a set of correlated variables to a set of uncorrelated variables (54). It is used extensively in multivariate datasets to find a reduced set of vectors which could serve as a basis from which all other vectors are generated as their linear combination. Converted to vectors, images are seen from the PCA technique under the same point of view, with the aim to find the most informative base vectors (29). PCA helps us compute a set of *eigenfaces* (i.e. the set of vectors which will comprise the basis). It also reduces the dimensionality of images which is in favor of further processing.

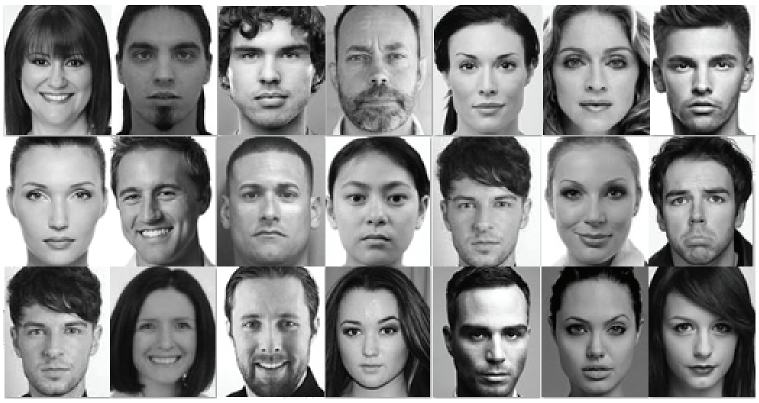
PCA needs two sets of data: training and testing. In previous work (72), two databases of images were used: the first one is the training set used to generate the eigenfaces; and the second one is the testing set (its images will be expressed as a linear combination of the eigenfaces of the first set). The content of the first database is from publicly available images. They are 40 in total and are shown in Figure 7.1(a), whereas in 7.1(b) are shown samples of the testing database which is an excerpt of the *ORL* database(81). All the images were subjects in an upright, frontal position. The size of each image is 92×112 pixels, with 256 grey levels per pixel (0-255). For proper implementation, images of both sets were converted and processed so they have the same parameters.

For training, first, the PCA coefficients are generated as follows. Each image I_i is converted from matrix to vector and then matrix *G*, contains all the images *I* in their vector form. The dimensionality of *G* is $dim(G) = dim(I_i)$. This results in $dim(I_i) = 92 * 112 = 10304$ and $dim(G) = 10304 * 40$. Of course that this dimensionality is too large to process further. To find the most singular features among images, the common ones will be removed. Initially the mean value of the training set is calculated, having $n = 40$. Figure 7.2 shows the reconstructed mean face of this database.

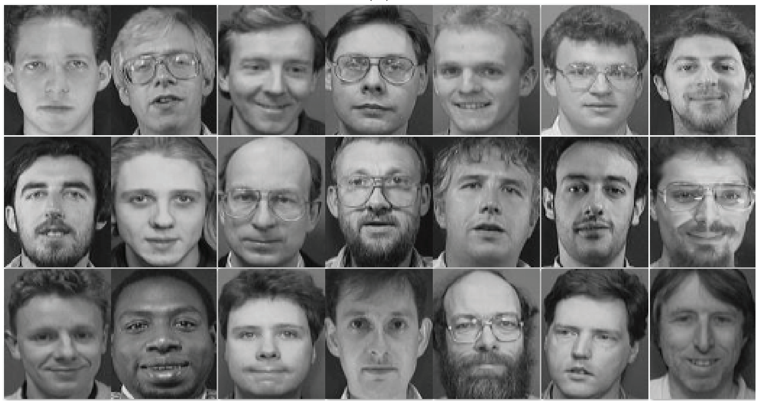
The mean face will then be subtracted by each face vector. The reason for this is that every image already contains the mean face and it represents redundant information. The final step is eigenvectors calculation by means of the covariance matrix (20). The covariance matrix will help describe the distribution of the *feature vectors*.

7.3.3 Template protection on honey-based system: Algorithm V

Sugar feature vectors in our experiments are generated according to the PCA technique. On these feature vectors is applied a protection scheme, which is presented in Figure 7.3. In this scheme, a mask vector m_j is generated according to the



(a)



(b)

Figure 7.1: (a) Samples in training database which is used for eigenfaces generation. (b) Samples in testing database which is used for recognition performance (72)

sign values of the plain feature vectors SB_i. Then, a double round of modulus operation is applied. First, is calculated the remainder of the mask vector and the standard deviation (σ_x and σ_y are two parametrized standard deviations to better tune the performance of the system). Its result is added to the plain feature vector and then a second round of the modulus operation is applied. The resulting vector is the protected sugar template.

On the other hand, for the honey templates random real-value vectors were used, RB_{ij} with $j = 1, 2, ..., k$. For each of the 400 samples of the testing database, 15 honey templates were generated. Their generation follows the same technique of Figure 7.3. In Figure 7.4 we can see an example of user 1 in the testing

Figure 7.2: Mean face reconstruction of the set of images in the training database.

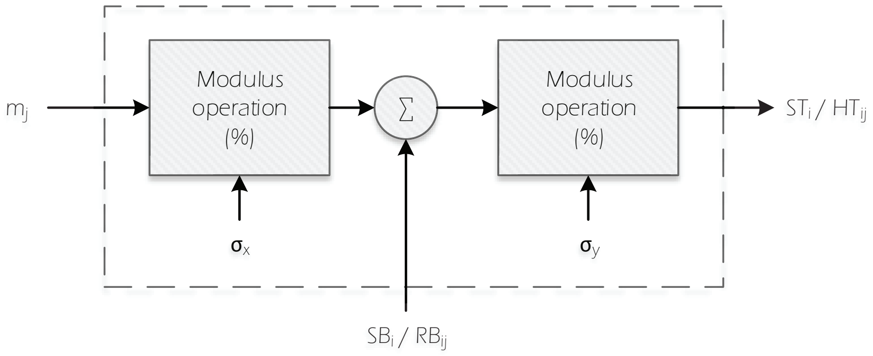


Figure 7.3: Protection mechanism on sugar and honey templates

database with the raw image, sugar template reconstruction, and a subset of its corresponding honey templates.

The technique we mentioned can be found in its extended form in (72), but being a quick proof-of-concept it presents two main difficulties related to: extra auxiliary data which needs to be stored; and the representation of the template format is simple. For these reasons, as a protection algorithm to be applied on the PCA feature vectors of our database we suggest an improvement of the dynamic random projection method (115), where the auxiliary database is eliminated and the template representation is more complex.

7.3.4 Secret-based Random Projection as a protection mechanism on feature vectors

Random projection is a technique which provides unlinkable templates by projecting a biometric feature vector in a set of orthonormal random vectors. To achieve

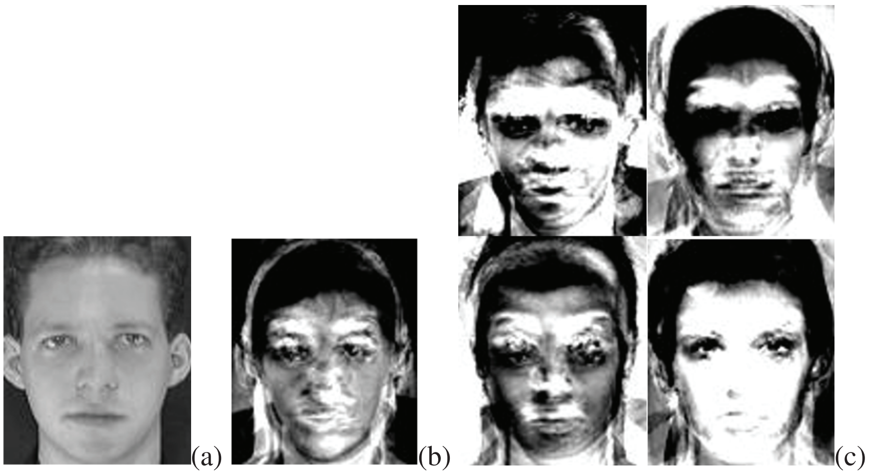


Figure 7.4: Example of first sample of user u1: (a) raw image at enrollment; (b) image of constructed sugar template; and (c) images of a subset of the corresponding honey templates

this, the feature vector is multiplied by a random real-value matrix.

To add security to the scheme the projection matrix is different for every template and we propose it is generated by a cascade of Pseudo Random Number Generators (PRNGs) where only the initial seed is determined and the others are randomly generated. The templates are going to be created as a concatenation of template-blocks where each block is generated as in the RP method, i.e. the plain feature vector will be projected by a number of N-random generated matrices. So, the input of the random projection are the PCA-feature vectors of dimensionality $dim = 20$ we provided in Sect. 7.3.1. In matrix multiplication from linear Algebra, in an orthogonal transform, if the transform matrix is orthogonal (such that its vectors form a basis), then there exists a *1-to-1* mapping between input and output vectors (a *bijective function*), i.e. the process is reversible. To preserve the security of the output vectors, and having an irreversible process, the projection matrix should have rank lower than the feature vector dimensionality, i.e. < 20. The number of rows can be e.g. equal to 10, but having as output feature vectors (protected templates) of dimensionality 10 is of course not secure, for this reason we construct a scheme, shown in Figure 7.5.

We first suppose we have from user i a plain feature vector fv_i represented by 20 real-value numbers. The first number generator ($PRNG_0$), having an initial determined seed, will generate one random secret value s_i. This value will serve as a seed for the second generator ($PRNG_i$) which will generate N-random seeds

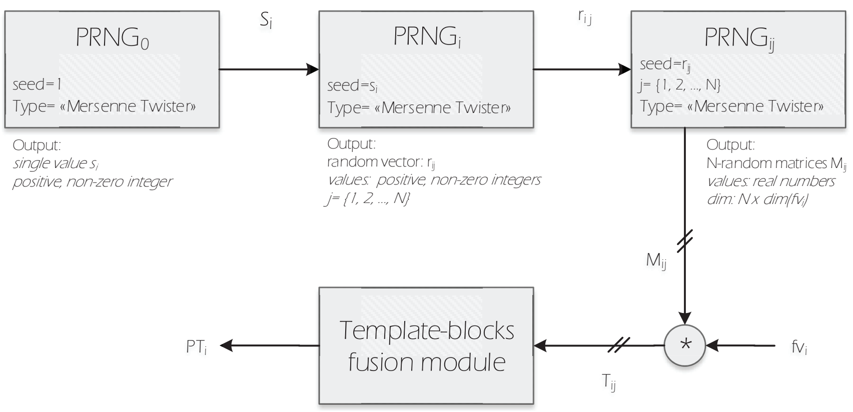


Figure 7.5: Secret-based random projection scheme for template protection

r_{ij} for the final $PRNG_{ij}$. The output of this generator will be a set of N-random matrices of dimensionality $N \times dim(fv)$, where in our case $dim(fv) = 20$. Each of these matrices will be multiplied with the feature vector of user i, having as an output N-template blocks T_{ij}. These blocks will be concatenated at the template-blocks fusion module and we finally have the i-th user protected template, which will be a real-value vector of length $n = 20 * N$. The performance of the secret-based random projection technique shows a slight improvement compared to the previous result in (72). The EER in this case is equal to $11,18\%$ and the performance can be shown in the graph in Figure 7.6. Honey templates are generated in the same way as sugar templates, but instead of plain feature vectors we used synthetic real-value feature vectors. These are created as such: for every subject, a random number *t* is generated. The database of images order is temporarily randomized and the first *t* image samples are chosen. We then perform the average of pixel values and create a synthetic image. This image will be treated in the same way as real images: PCA coeffiecients will be extracted, and the protection mechanism will be applied. For every subject, and for every new honey template, the value of *t* varies in the interval $10 - 100$.

After we generated and protected both sets of sugar and honey templates, we need to be sure that they do not differ. This should stand for both cases: if we classify them with an automatic classifier, and even visually in terms of features or other image peculiarities. For this reason, we have further argued the indistinguishability property of the templates, based on a human classification of pre-images and an automatic classifier. From the system perspective, a template leaked in its protected form can threat the system's security. An adversary can find a pre-image of the protected template, having as a result raw data of the biometric characteris-

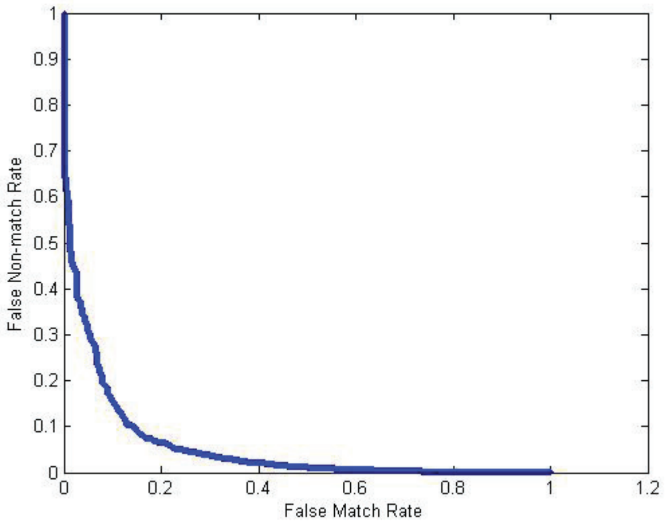


Figure 7.6: EER of sugar templates with secret-based random projection as a protection mechanism.

tic. In fact, the pre-image may not totally match the plain template. If we assume that even other auxiliary data stored in the database are leaked, this may lead the adversary to the creation of potentially 'look-alike' characteristics with genuine ones. We apply this technique for three advantages: first we eliminate the auxiliary database used in our previous scheme; secondly the augmented secret values will make the security of the mechanism more robust; and thirdly the vector representation of the templates is more complex.

7.4 Template distinguishability tests

In our honey-based system, we can say that the difficulty of the attacker to crack the algorithm resorts to probability (i.e. information-theoretic security) instead of computational complexity based security. Nevertheless, an accurate evaluation of pre-images in terms of distinguishability helps us define the security performance of the scheme.

7.4.1 Pre-images classification

Let's suppose that the attacker has cracked our template protection scheme and possesses a set of 6400 mixed templates: 400 sugar and 6000 honey. To discriminate the two sets he can apply two classification techniques: an automatic classifier (Support Vector Machine -*SVM*) and a human classification of pre-images. In this successful attack scenario the attacker has reconstructed the cracked templates having as a result images similar to Figure 7.4(b) and (c). For SVM classification we first transform the pre-images to PCA feature vectors, i.e. define training and testing set for PCA feature vectors, and secondly define training and testing set for SVM classification. We have chosen the Support Vector Machine classifier since it is more effective in high dimensional data. Its aim is in constructing a hyper-plan to divide the vectors in two classes. We have in total 6400 pre-images (400 sugar pre-images, SP_I; and 6000 honey pre-images, HP_I). For the training set of PCA feature vectors we followed the same technique described earlier in Section 7.3. The training set in this case (substituting the database of public faces shown in Figure 7.1(a)) is a union of two sets, as in (19). Having $m = 15$ honey templates for each sample, we take from HP_I the first honey feature vector of each of the 10 samples for the first 10 subjects.

$$PCA_{Train} = S_{PI_i} \cup H_{PI_j}, \forall i, j \in \aleph \tag{7.2}$$

$i = \{1, 2, ..., 100\}$

$j = (j_1, j_2, ..., j_q, ..., j_{100})$

$j_q = (q-1) * m + 1$

$$|PCA_{Train}| = 200 \tag{7.3}$$

For the generation of eigenfaces, pre-images are first converted in vectorial form; the mean face is calculated (shown in Figure 7.7); it is subtracted from every image; eigenvectors and eigenvalues are calculated and sorted by finally having 20 eigenfaces, the new basis (Figure 7.8).

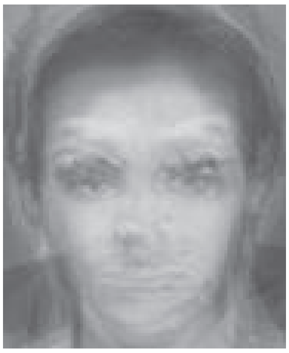


Figure 7.7: Mean face of PCA_{Train} set.

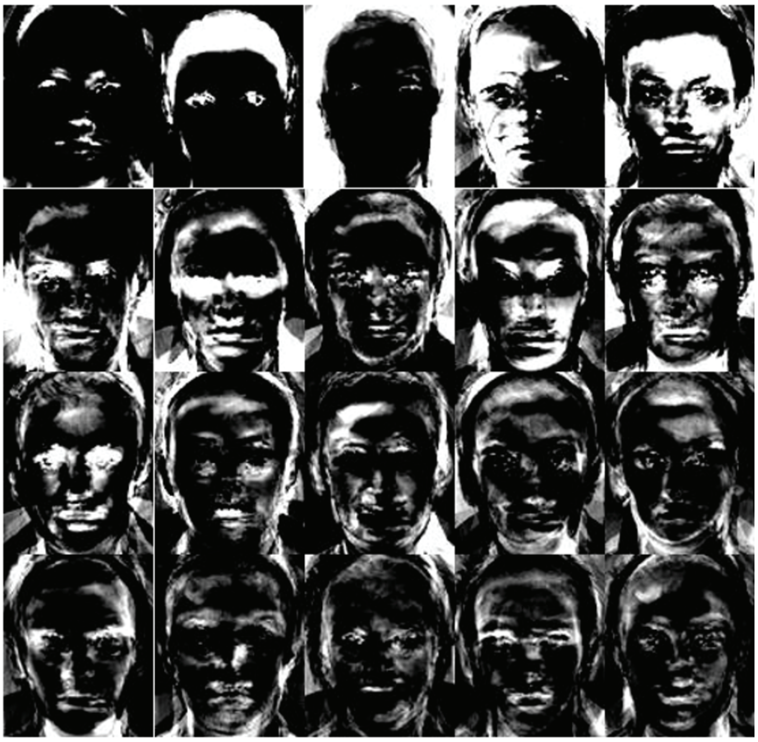


Figure 7.8: Eigenfaces: generated from PCA_{Train} set.

After eigenvectors' normalization, we have the PCA coefficients for the training set, represented as real-value feature vectors of dimensionality equal to 20. The PCA testing set consists of what remains from the whole set of sugar and honey pre-images after we have taken the training set.

$$PCA_{Test} = S_{PI_i} \cup H_{PI_j} - PCA_{Train}, \tag{7.4}$$

$i = \{101, 102, ..., 400\}$

$j = \{1501, 1502, ..., 6000\}$

$$|PCA_{Test}| = \left|S_{PI_i} \cup H_{PI_j}\right| - |PCA_{Train}| = 4800 \tag{7.5}$$

The testing set is going to be used for the SVM classifier training and testing phases. After reconstruction of images of PCA_{Train} and PCA_{Test} we store the sugar and honey feature vectors as S_{FV} and H_{FV}, defined as:

$$SVM_{Train} = S_{FV_i} \cup H_{FV_j} \tag{7.6}$$

$i = \{1, 2, ..., 100\}$

$j = (j_1, j_2, ..., j_q, ..., j_{100})$

$j_q = (q - 1) * m + 1$

$$SVMTestingSet = S_{FV_i} \cup H_{FV_j}, \forall i, j \in \aleph \tag{7.7}$$

$i = \{101, 102, ..., 300\}$

$j = (j_{101}, j_{102}, ..., j_q, ..., j_{300})$

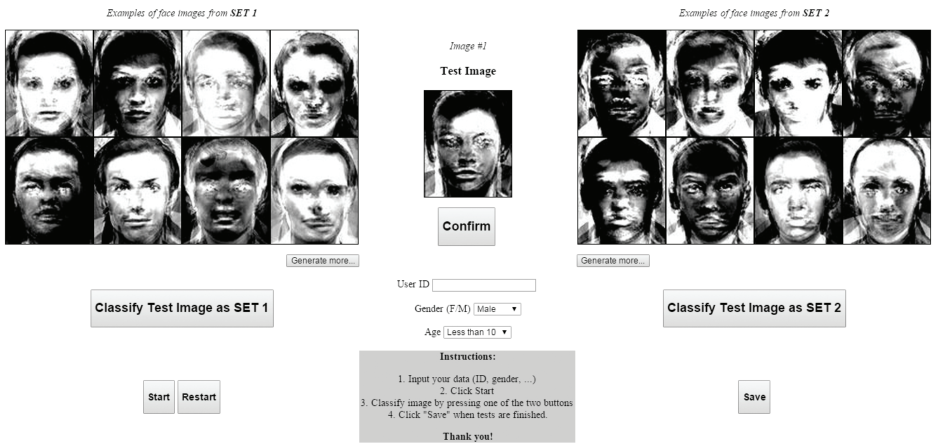
$j_q = (q - 1) * m + 1$

The cardinality of the training set is 200 and of the testing set is 400. We have labelled the two sets as 0 and 1 (meaning *sugar* and *honey* respectively), and have as a result an error rate of 0.44, which is a high enough value to tell the undistinguishability between sugar and honey templates.

7.4.2 Pre-images classification with human testers

Another approach we took for the evaluation of indistinguishability between sugar and honey templates is to ask a group of subjects/testers. Attackers, can establish similar tests too and they can get trained in this visual classification of pre-images. For this reasons we conducted this study and obtained 30 volunteers. The whole pool of subjects was unsupervised. They were each asked to classify a set of 30 pre-images chosen randomly between sugar and honey templates, through a web-based system we provided (Fig. 7.9).

There was no time limitation in the classification process, anyhow time was measured to understand if human classifiers were really looking at features or other elements they could percept to make the difference between pre-images. Before starting they were offered random two sets of faces (one from sugar pre-images and one from honey pre-images) so they could explore them and gain a perception on the images. Even during the test, they had these sets at their disposal. The testing platform was built in two versions: desktop-based for on-campus volunteers; and web-based for remote volunteers. Among the human classifiers: 57% were female; 50% in the age category of 26-35 years; and 37% did the test remotely using the web-based platform. The average error rate of the subjects is 0.419 and in Figure 7.7, we can see the distribution of error rates divided in 8 intervals. The


Examples of face images from SET 1
Examples of face images from SET 2
Image #1
Test Image
Confirm
Generate more...
Generate more...
Classify Test Image as SET 1
Classify Test Image as SET 2
Start
Restart
Save

User ID

Gender (F/M) Male

Age Less than 10

Instructions:

1. Input your data (ID, gender, ...)
2. Click Start
3. Classify image by pressing one of the two buttons
4. Click "Save" when tests are finished.

Thank you!

Figure 7.9: System GUI for the volunteer classification testing.

maximum error rate is 0.6 and the minimum value, meaning the most accurate tester had an error rate equal to 0.23. From Table 7.1, it can also be seen the error rates of different categories, based on gender, platform and age.

Table 7.1: Statistics on the error rates and average time of pre-image classifiers.

Category		Average error rate	Average time (sec)
Gender	Female	0,413	11,5
	Male	0,427	17,7
Platform	On-campus	0,436	14,8
	Web-based	0,407	13,9
Age	≤ 17	0,427	13,0
	18 - 25	0,407	16,1
	26 - 35	0,420	12,9
	36 - 40	0,450	17,9

In the different sub-categories we can see that women were slightly more accurate than men and also faster in making their decision (in Table 7.1, we can see even the average time that a subject needed to make a decision for one image). Volunteers that we gathered on-campuses were more accurate than online testers, even though their estimated time was nearly the same. Regarding the ages, the most accurate category was 18-25 years old, who also took more time than the other categories in their classification.

We have divided the subjects in 8 intervals, based on their estimated error rate. Here, 27% fall in the error interval of 0.31-0.35. Regarding the time average of pre-image classification, being a cognitive process, we can notice that the average time of the first images is longer that of the last images.

After the test we interviewed the on-campus subjects only. They were asked on the perception they had on the images, level of difficulty and types of features they were looking on to make a decision. The features they were looking on the images were: the periocular area, face shape and background. They estimated that the distinguishability as difficult. From an attacker point of view, we can summarize that a human classifier is behaving better in terms of error rate with its average rate of 0.419, compared to the SVM classifier, whose rate is 0.44.

7.5 Conclusions

For the experiment result, we can also evaluate the indistinguishability property of the proposed template protection scheme. The results suggest that the proposed method has a quite good indistinguishability property because the error rate of both human classifier and SVM is relatively high.

The classifier is provided by only two options (sugar or honey). A random based selection would have a 0.5 error rate on average to guess which category the pre-image belong to. The error rate obtained by both classifiers is very close to the value of 0.5. Therefore, the classifier cannot distinguish the sugar from the honey well. It means that the proposed method can build quite indistinguishable honey templates.

7.6 Brief chapter summary

In this chapter we provided the benefits that could bring the adoption of honey templates as a security mechanism in biometric systems. We analyzed the different threats and system vulnerabilities and provided how these could be relieved by using honey templates. We argued that PCA can be used in template protection design as a feature transformation scheme, but, by using this technique, a possible attacker could reconstruct the templates and try to define if a given pre-image could be a sugar template, or one of the honey templates. For this reason, template undistinguishability has to be measured and strong metrics must be set off for practical implementation.

To provide sound result we used two classification technique: one based on human classification and one on automatic classification. For the first type we constructed a platform where human testers could classify a set of random pre-images coming from sugar or honey templates. We compared the obtained result with an SVM classifier. We noticed that, compared to the SVM classifier, human testers showed better results in terms of classification distinguishability and from an attacker point of view.

This result means that with a trained subject an attacker could improve its chances of guessing a sugar template among a set of honey templates, after the pre-image classification. The accurate tester we had with the minimum error rate equal to 0.23 and other testers who achieved a similar result could be subject to further studies regarding image perception and human memory classification processes.

In this chapter we provide the answer to **RQ3**, regarding who would be more accurate between automatic classifier or human testers, if face biometric templates -real or honey- were reconstructed.

Chapter 8

Measuring the attacker's confidence on a honey-based system

8.1 Introduction

One of the challenges of this approach is the design of honey templates, precisely their distinguishability with sugar ones. If an attacker somehow possesses the content of a biometric database augmented with honey templates, he should not be able to differentiate the two categories of templates. Even if he has full knowledge on the system (i.e. the fact that honey templates are deployed, what algorithm parameters are used to generate honey templates, classifier rates, etc.), and possesses sophisticated classification tools, a thorough analysis is needed to evaluate his chances in distinguishing templates.

Supposing this type of attacker, we propose in this thesis a methodology for the theoretical evaluation of template predictability based on Bayesian inference. We simulate a honey-based system generating and storing K synthetic templates for each subject in the same way the authors propose in (113).

Furthermore, a trained classifier C will be used to test a sugar transformed template ST, to see if it can be classified as a honey one. If yes, this ST will be saved in the biometric database, otherwise a new ST is generated to be tested by C again. Templates testing and storing will be even controlled by the classifier by this condition: sugar templates are classified and stored as honey templates to a certain percentage q (i.e., the iteratively generated sugar templates can be all considered as honey, or only a portion q of them). We suppose the attacker knows even q.

So, to verify that this approach really benefits the defender, (or the contrary, the attacker will have a higher confidence in discarding honey templates and keeping sugar templates) we raise two main issues: 1) will a defender be better protected if he artificially tunes the parameter q to tweak the distribution of observed sugar and honey templates, supposing both defender and attacker own an ideal classifier? And, 2) in real life applications where the classification rate is not ideal, will a defender be better protected when percentage is tuned?

From a security perspective, the proposed methodology will provide a theoretical proof on the predictability of honey and sugar templates. This will offer a clearer reliability on the scheme proposed in (113) (and partially implemented in (73)). It relates to the system practical implementation too (the system design will be affected by the fact if it is safer to tune the template distribution).

Finally, we prove that a defender **is not** better protected when tuning the percentage of templates, and he **is** better protected when classification rates are not ideal. This means that the methodology is applicable to real-life scenarios.

Experimental results will be provided by simulating a face verification biometric system. The biometric template protection (BTP) mechanism applied both on biometric and honey templates will be Random Projection, a feature transformation scheme. In section 2 an overview of the honey-based system architecture will be given, in sections 3 and 4 the two security issues regarding template predictability will be theoretically proven, and section 5 will analyse the experimental results of the face verification simulated system.

8.2 Bayes inference for templates predictability

The main challenge of honey templates generation, is their distinguishability. Given a set of sugar and honey templates, would an attacker be able to classify biometric templates and discriminate sugar templates from honey ones? In fact, regardless the implemented security mechanisms of a system, attackers take always risk of cracking them to get access to the system data. There may be different types of attackers based on the level of knowledge on the system design, and the sophistication of tools they possess. We suppose the following types of attackers.

Low level attacker: This attacker is not aware of the system design, database structure and as consequence of the honey-templates existence. After getting the template file, the attacker considers them as genuine templates. He decides to use template PT_{ij}, and with probability $P = (K-1)/K$ he will try a honey template.

Middle level attacker: This attacker is aware of the biometric system design, database structure and of the fact that it is using honey templates. He possesses no other tools and in this case he can be considered as a *medium level attacker*. After getting the template file, the attacker will try to guess the template. He decides to randomly use template PT_{ij} or he can preserve it, e.g. for further classification, which makes him a . If his guess is correct, the attacker will be able to enter the system. But the probability of guessing the genuine template depends on the number of sweet templates (as a result, the security complexity of the scheme is not dependent only on the protection algorithm complexity, but it is even probability-based). Depending on the number of honey templates, this probability should be equal to:

$$P\left(j = m\right) = \frac{1}{K} \tag{8.1}$$

High level attacker: Attacker is aware of the biometric system design; database structure and of the fact that it is using honey templates; algorithm used to generate protected honey and sugar templates. He has previously gathered sets of sweet templates, but he can not distinguish them (since sugar templates are randomly stored among honey templates). After getting the template file, the attacker will try to run a classification test and then make a decision.

According to the third scenario an attacker will observe a set of sweet templates, from which he has no knowledge which is a sugar template. Regardless of the supposed security increase of the honey templates method, we need to give proof of attacker's predictability. To analyse it, we provide these considerations:

a. The attacker A has designed a system similar to the defender system D. He owns auxiliary databases and a testing biometric database.

b. The attacker owns a classifier C having an accuracy rate c to classify the two categories of templates. His aim is to have c^A as close as possible to c^A

c. After classification the attacker has: *observed sugar templates* (ST_s) and *observed honey templates* (HT_s) according to classifier's decision. Prior to classification they could have been *original sugar templates* (ST_o), or *original honey templates* (HT_o).

To measure the degree of confidence of the attacker and evaluate the inference, we propose Bayes theorem (105). This theorem will relate "the prior" degree of confidence in the event A with "the posterior" degree after considering the event B, as in Equation 2:

Algorithm 8.1: Iterative generation of sugar templates depending on the parameter *q* for *n*-users.

```
input  : A set of biometric features $SB_i$.
output: A set of protected sugar templates $ST_i$.

for $i \leftarrow 1$ to $n$ do
    Read($SB_i$);
    $AD_i \leftarrow$ PRNG($rand_{value}$);
    $ST_i \leftarrow$ genTemplate($BTP, SB_i, AD_i$);
    category $\leftarrow$ classifyTemplate($ST_i$);
    if category $is$ Sugar then
        countSugar $\leftarrow$ countSugar ++;
    if category $is$ Honey then
        countHoney $\leftarrow$ countHoney ++;
    if countSugar / countHoney $==$ $q$ then
        // keep $q$ sugar templates
        store($ST_i$, Temp);
        store($AD_i$, Temp);
    else
        GO TO 3;
        // a new auxiliary data will be generated
```

Algorithm 8.2: Generation of *k* honey templates with defined auxiliary data, for a specific user *i*.

```
input  : Auxiliary data $AD_i$
output: A set of protected honey templates $HT_{ij}$

Read($AD_i$);
for $j \leftarrow 1$ to $k$ do
    $HB_{ij} \leftarrow rand_{value}$;
    $HT_{ij} \leftarrow$ genTemplate($BTP, HB_{ij}, AD_i$);
    category $\leftarrow$ classifyTemplate($HT_{ij}$);
    if category $is$ Honey then
        store($HT_{ij}$, Temp);
    else
        GO TO 3; // a new honey feature will be
         generated
```

Algorithm 8.3: Storage of randomized templates, and sugar templates indexes in respective databases.

input : A set of ordered protected templates: one sugar and k-honey stored in Temp

output: 1) A record of templates in the Biometric Database; 2) The index of the sugar template in the Honey Checker Database.

```
1 Read(Temp);
2 $PT_i \leftarrow$ randomize(Temp);
3 $L_i \leftarrow$ index($ST_i, PT_i$);
4 store($PT_i$, $HBiomSys^{DB}$);
5 store($L_i$, $HCheckSys^{DB}$);
```

$$P\left(A|B\right) = \frac{P\left(B|A\right) \cdot P\left(A\right)}{P\left(B|A\right) \cdot P\left(A\right) + P\left(B|\sim A\right) \cdot P\left(\sim A\right)} \tag{8.2}$$

In Eq. (2), for the attacker to be able to predict and distinguish templates, he needs to know from the system: the prior probabilities $P\left(ST_o\right)$ and $P\left(HT_o\right)$, i.e. the total number of sugar and honey templates in the database; and secondly the likelihood, or the conditional probability $P\left(B|A\right)$ which is related to what is observed from templates classification.

In the honey templates context, there are four probabilities to be considered and evaluated by the Bayesian inference:

- $P\left(ST_o|HT_s\right)$: probability that an observed honey template is in fact an original sugar template.

$$P\left(ST_o|HT_s\right) = \frac{r^{HT} \cdot (1-k)}{r^{HT} \cdot (1-k) + c^{ST} \cdot k} \tag{8.3}$$

- $P\left(ST_o|ST_s\right)$: probability that an observed sugar template is indeed an original sugar template.

$$P\left(ST_o|ST_s\right) = \frac{r^{ST} \cdot (1-k)}{r^{ST} \cdot (1-k) + c^{HT} \cdot k} \tag{8.4}$$

- $P\left(HT_o|HT_s\right)$: probability that an observed honey template is indeed an original honey template

$$P\left(HT_o|ST_s\right) = \frac{c^{HT} \cdot k}{c^{HT} \cdot (k) + r^{ST} \cdot (1 - k)} \tag{8.5}$$

- $P\left(HT_o|ST_s\right)$: probability that an observed sugar template is in fact an original honey template.

$$P\left(HT_o|HT_s\right) = \frac{c^{ST} \cdot k}{c^{ST} \cdot (k) + r^{HT} \cdot (1 - k)} \tag{8.6}$$

In Equations (3)-(6), the parameters: r^{ST} and r^{HT} represent the probability that a sugar or honey template is generated from the defender's system; c^{ST} and c^{HT} represent the classification rates of observed sugar and honey templates; and the parameter k shows the actual honey templates ratio related the whole population of templates (i.e. $1-k$ is the respective sugar templates percentage). The template can be considered as a function of the percentage r^T and the classification rate c^T.

$$T = f(r^T, c^T) \tag{8.7}$$

Two cases are evaluated in the following sections, considering different percentages of templates generation, and different classification rates.

8.3 Designing attacker's confidence

8.3.1 CASE A: Attacker's confidence for tuned template generation

To analyse if a defender will be better protected in case he artificially tunes the biometric template protection parameter to tweak the distribution of observed sugar and honey templates, two scenarios are compared: *i*) without any parameter tuning for template generation; *ii*) with parameter tuning *q* for template generation. The understanding of the scenarios is given via examples, and a generalized evaluation will follow. In Case A, it is supposed that the classification rates are equal to the ideal case, i.e. $c^{ST} = c^{HT} = 0.5$.

Example 1.

Let's consider as a first baseline case that there is no percentage of sugar templates generation (this implies that half of sugar templates are generated as such, and half of honey templates are generated as honey templates). We suppose there are $K = 9$ honey templates for 1 sugar template. This yields that the prior probability *k* of observing a honey template is equal to 0.9, and the prior probability of observing a sugar template is $1 - k = 0.1$.

Since the classifier's rate r is equal to 0.5 it is expected and the results show that the level of confidence of an attacker to predict that an observed sugar template is indeed an original sugar template, equals to that of honey templates. Results are summarized in Table 1. ∎

Table 8.1: Theoretical evaluation of probabilities for two scenarios: straightforward generation of sugar and honey templates (Example 1), and tuned generation of templates having $q = 30\%$ (Example 2).

Probability	*Example 1*	*Example 2*
$P\left(ST_o \vert HT_s\right)$	0.1	0.1346
$P\left(ST_o \vert ST_s\right)$	0.1	0.0625
$P\left(HT_o \vert ST_s\right)$	0.9	0.9375
$P\left(HT_o \vert HT_s\right)$	0.9	0.8654

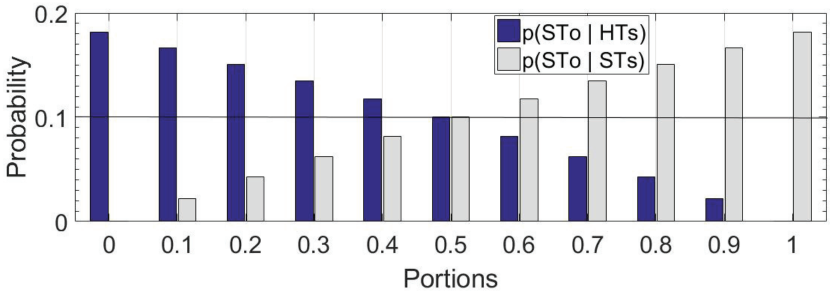


Figure 8.1: Comparison of probabilities $P\left(ST_o|HT_s\right)$ and $P\left(ST_o|ST_s\right)$ for a constant classification rate $c = 0.5$ and different sugar templates portions r^{ST}.

If the defender does not tune the percentage of sugar templates, i.e. 50% of ST_o are HT_s and 50% HT_o are ST_s an attacker has equal chance to try a ST_s or HT_s because $P(STo|HTs) = P(STo|STs) = 0.1$. To the attacker it makes no difference which templates to try.

In the second type of system, templates are forced to be generated with a percentage r^{ST} which is not equal to 50%. This is an information the attacker has. He wants to re-calculate the probabilities of trying ST_s or HT_s and would see if trying ST_s or trying HT_s can give better chance than the baseline case. If this is the case, which of ST_s and HT_s should he try? Let's give an example.

Example 2.

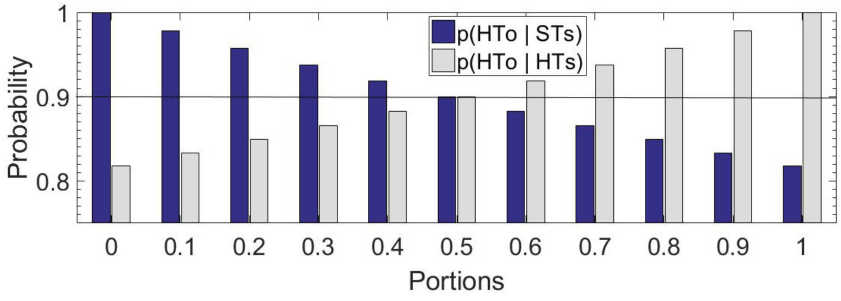


Figure 8.2: Comparison of probabilities $P\left(HT_o|HT_s\right)$ and $P\left(HT_o|ST_s\right)$ for a constant classification rate $c = 0.5$ and different sugar templates portions r^{ST}.

Let's now consider there is a percentage of sugar templates generation (having again $K = 9$ honey templates for 1 sugar template, i.e. $k = 0.9$). In this case the ratio between sugar and honey templates is equal to 0.3 meaning that the parameters are: $r^{ST} = 0.3$, $r^{HT} = 0.7$, $c^{ST} = c^{HT} = 0.5$. Probabilities of Eq. (3)-(6) are evaluated and presented in Table 1. Results show that $P\left(ST_o|ST_s\right) = 0.0625 < P\left(p = 0.1\right)$, and $P\left(ST_o|HT_s\right) = 0.1346 > P\left(p = 0.1\right)$. Similarly, $P\left(HT_o|ST_s\right) = 0.9375 > P\left(p = 0.9\right)$, and $P\left(HT_o|HT_s\right) = 0.8654 > P\left(p = 0.9\right)$. ▪

From results of Example 2, an attacker knows he should try HT_s instead of ST_s, because he has a higher confidence that this honey template was originally a sugar template. Therefore, he is going to try a HT_s to spoof the authentication system.

In the graph in Fig. 8.1 are shown the probabilities for different portions of ST_o. They range from 0%, 10%, to 100% of sugar templates generated as sugar templates. From the graph it yields that the degree of confidence of an attacker to try an observed honey template is higher if $r^{ST} < 0.5$. And, in the case the defender generates sugar templates with $r_{ST} > c = 0.5$, the attacker knows he should try ST_s, since it gives him a higher chance than the baseline case to spoof the system.

Whereas, in Fig. 8.2 are shown the probabilities for different portions of HT_o. Again they range from 0%, 10%, to 100%, and in this case the degree of confidence of an attacker to try an observed honey template is higher if $r^{ST} > 0.5$. Following these results, there are two strategies that can be taken: (1) *increase degree of confidence in* ST_o; (2) *decrease degree of confidence in* HT_o.

(1) ***Increase degree of confidence in*** ST_o

Generalization of strategy (1), will help a defender *D* determine the percentage of template tuning, supposing an ideal classifier with $c_{ST} = c_{HT} = c$ is implied. By Example 2, it was concluded that if $r^{ST} < c = 0.5$ the attacker should try HT_s, and if $r^{ST} > c = 0.5$ the attacker should try observed sugar templates. In both cases chances are higher, so the best strategy for the defender system is not to force the template generation, but just let $r^{ST} = r^{HT} = c$. This indicates that the defender should not tune the BTP parameters and manipulate sugar templates. To formalize this claim, let's give formal proof through Lemma 1.

Let D *be a defender biometric system, and* A *an attacker biometric system trying to mimic the process of templates generation in* A*. Suppose that* D *uses a biometric template protection mechanism* BTP^{HT} *implying honey templates, according to which sugar templates* ST_o *are tweaked according to a parameter* r^{ST}*. Suppose a classifier* C *rates of observed templates are equal:* $c^{ST} = c^{HT} = c$*.*

Lemma 1 (Attacker's Confidence when observing ST) *To better defend the biometric system* D*, and equilibrate an attacker's confidence in trying* ST_s *or* HT_s *when the observed template is* ST_o*,* D *should have* $r^{ST} = r^{HT} = c$*.* □

PROOF Let $conf^{A}_{ST}(ST)$ be the confidence level of an attacker in trying ST_s and $conf^{A}_{HT}(ST)$ in trying HT_s, given that the template was an original sugar one. They are measured according to Bayes inference in Eq. 2, as:

$$conf^{A}_{ST}(ST) = P(ST_o|ST_s)$$

$$conf^{A}_{HT}(ST) = P(ST_o|HT_s)$$

First, let's suppose that $r^{ST} < c$. The following transformations show that:

$$\begin{gathered}
r^{ST} < c \\
r^{ST} \cdot k < c \cdot k \\
r^{ST} \cdot k + r^{ST} < r^{ST} + c \cdot k \\
r^{ST} < r^{ST} \cdot (1 - k) + c \cdot k \\
\frac{r^{ST}}{r^{ST} \cdot (1 - k) + c \cdot k} < 1 \\
\frac{r^{ST} \cdot (1 - k)}{r^{ST} \cdot (1 - k) + c \cdot k} < 1 - k \\
P\left(ST_o|ST_s\right) < P\left(p = 0.1\right)
\end{gathered}$$

$$\Rightarrow conf^{A}_{ST}(ST) < P\left(p = 0.1\right)$$

Similarly, since $r_{ST} = 1 - r_{HT}$, now $r_{HT} > c = 0.5$.

$$\begin{aligned}
r^{HT} &> c \\
r^{HT} \cdot k &> c \cdot k \\
r^{HT} \cdot k + r^{HT} &> r^{HT} + c \cdot k \\
r^{HT} &> r^{HT} \cdot (1 - k) + c \cdot k \\
\frac{r^{HT}}{r^{HT} \cdot (1 - k) + c \cdot k} &> 1 \\
\frac{r^{HT} \cdot (1 - k)}{r^{HT} \cdot (1 - k) + c \cdot k} &> 1 - k \\
P\left(ST_o|HT_s\right) &> P\left(p = 0.1\right) \\
\Rightarrow conf^{A}_{HT}(ST) &> P\left(p = 0.1\right)
\end{aligned}$$

Since, $conf^{A}_{HT}(ST) > conf^{A}_{ST}(ST)$, an attacker would try HT_s instead of ST_s, if $r^{ST} < c = 0.5$. In case $r^{ST} > c = 0.5$, by following the same transformations as above, the attacker confidences will change: $conf^{A}_{ST}(ST) > conf^{A}_{HT}(ST)$, meaning that he has a higher confidence in trying ST_s.

If $r^{ST} = r^{HT} = 0.5$, both confidences are equilibrated and the probabilities of an attacker to try an observed sugar or honey template would be equal, meaning that the defender system *D* is better protected. ■

(2) *Decrease degree of confidence in* HT_o

The second strategy is to decrease the degree of confidence in honey templates. Following the same proof logic of Lemma 1 in strategy (2) we first suppose $r^{ST} < c = 0.5$. In this situation, $P(HT_o|ST_s) > P(p = 0.9)$ and similarly, $P(HT_o|HT_s) < P(p = 0.9)$ (where $P(p = 0.9)$ is the baseline case). Here, the attacker should try HT_s because it gives a lower chance that this was originally a honey template, than the baseline case. It is even less than the case of trying an ST_s.

Now the attacker checks the other situation, where $r^{ST} > c = 0.5$. This will yield that $P(HT_o|ST_s) < P(p = 0.9)$ and similarly, $P(HT_o|HT_s) > P(p = 0.9)$. Now the attacker should try a ST_s since it has a lower chance to be an original HT_o than trying a ST_o. To formalize this claim, let's give proof through Lemma 2.

Lemma 2 (Attacker's Confidence when observing HT) *To better defend the biometric system* D*, and equilibrate an attacker's confidence in trying ST_s or HT_s when the observed template is HT_o,* D *should have $r^{ST} = r^{HT} = c$.* □

Proof Let $conf^A_{ST}(HT)$ be the confidence level of an attacker in trying ST_s and $conf^A_{HT}(HT)$ in trying HT_s, given that the template was an original honey one. They are measured according to Bayes inference in Eq. 2, as:

$conf^A_{ST}(HT) = P(HT_o|ST_s)$

$conf^A_{HT}(HT) = P(HT_o|HT_s)$

First, let's suppose that $r^{ST} < c$. The following transformations show that:

$$\begin{gathered}
r^{ST} < c \\
r^{ST} \cdot (1-k) < c \cdot (1-k) \\
c - c \cdot k > r^{ST} - r^{ST} \cdot k \\
c > c \cdot k + r^{ST} - r^{ST} \cdot k \\
\frac{c}{c \cdot k + r^{ST} \cdot (1-k)} > 1 \\
\frac{c \cdot k}{c \cdot k + r^{ST} \cdot (1-k)} > k \\
P\left(HT_o|ST_s\right) > P\left(p = 0.9\right) \\
\Rightarrow conf^A_{ST}(HT) > P\left(p = 0.9\right)
\end{gathered}$$

Similarly, since $r_{ST} = 1 - r_{HT}$, now $r_{HT} > c = 0.5$.

$$\begin{gathered}
r^{HT} > c \\
r^{HT} \cdot (1-k) > c \cdot (1-k) \\
c - c \cdot k < r^{HT} - r^{ST} \cdot k \\
c < c \cdot k + r^{ST} - r^{HT} \cdot k \\
\frac{c \cdot k}{c \cdot k + r^{HT} \cdot (1-k)} < k \\
P\left(HT_o|HT_s\right) < P\left(p = 0.9\right) \\
\Rightarrow conf^A_{HT}(HT) < P\left(p = 0.9\right)
\end{gathered}$$

As a conclusion, if $r^{ST} < c = 0.5$, the attacker should try HT_s, and if $r^{ST} > c = 0.5$, the attacker should try ST_s. In both cases, the chances to pick up an original HT_o are lower than the baseline case. So the best strategy for the defender is not to force the portion of sugar and honey template, but just let $ST_o = HT_o = c$. This indicates that, for the defender to be better protected, D should not tune the *BTP* parameters to manipulate ST_o. ■

Though using two different strategies, the attacker has the same conclusion from above analysis that if the defender changes *r*, the attacker will have higher chance to spoof the system. From the defender perspective, the defender should not change *r*, to a percentage other than 0.5.

8.3.2 CASE B: Attacker's confidence considering real-life classifiers

In Case A it is supposed that the classification rate is ideal, which in fact is not possible in real-life classifiers. Is a defender D better protected in case the classification rate $c \neq 0.5$ in a biometric system where templates are generated according to an iterative classification process? To answer this question, the same methodology as in Case A will be followed. In **Example 3** the new baseline scenario is presented, whereas in **Example 4** and **Example 5** are illustrated two situations of a real-life classifier in a system with tuned sugar templates generation.

Example 3. As a baseline scenario for Case B, let's suppose that the system does not employ a tuned template generation. We have now a classifier with $r = 0.2$, i.e., classifies 80% of all ST_o as ST_s and 80% of HT_o as HT_s. Theoretical results of the four predictions of Equations (3)-(6) are presented in Table 2. ▪

Example 4. Here a system with classification rate of observed sugar templates of $c_{ST} = 0.8$, and classification rate of honey templates $c_{HT} = 0.2$ is considered. The percentage of sugar templates is $r^{ST} = 0.3$. Results are shown in Table 2.

From the respective probabilities in Table 2, it can be seen that the attacker has a higher confidence in trying a sugar template, compared to an observed honey template. In fact, $P(ST_o|HT_s)$ is higher than the baseline case and $P(ST_o|ST_s)$ is lower. In this scenario, the attacker would choose an observed sugar template. ▪

Example 5.

Now, we suppose a system with classification rate of observed sugar templates of $c_{ST} = 0.8$, and classification rate of honey templates $c_{HT} = 0.2$. The percentage of sugar templates is $r^{ST} = 0.7$. Results are shown in Table 2.

Even in this case, the attacker has a higher confidence in trying a sugar template, compared to an observed honey template. Again he would choose an observed sugar template..

In both **Example 4** and **5**, if the percentage parameter q, becomes equal to the non-ideal classification rate c, the defender system would be better protected.

Table 8.2: Theoretical evaluation of probabilities for two scenarios: straightforward generation of sugar and honey templates (Example 3), and tuned generation of templates having $q = 30\%$ (Example 4), and $q = 70\%$ (Example 5). The classification rate of the classifier is $c^{HT} = 0.2$.

Probability	*Example 3*	*Example 4*	*Example 5*
$P(ST_o \vert HT_s)$	0.0270	0.0886	0.0400
$P(ST_o \vert ST_s)$	0.3077	0.1429	0.2800
$P(HT_o \vert ST_s)$	0.6923	0.8571	0.7200
$P(HT_o \vert HT_s)$	0.9730	0.9114	0.9600

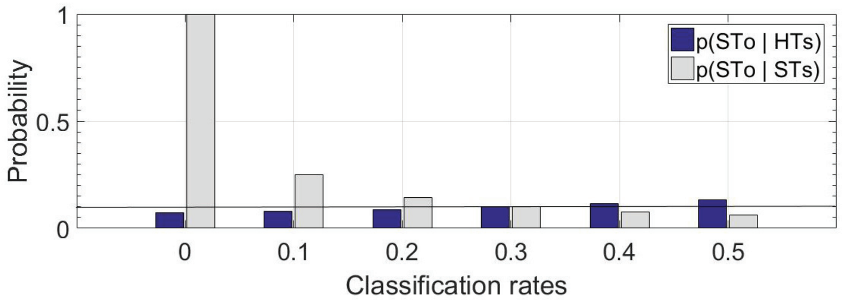


Figure 8.3: Comparison of probabilities $P(ST_o|ST_s)$ and $P(ST_o|HT_s)$ for a constant portion of sugar templates $r^{ST} = 0.3$ and different classification rates c^{ST}.

In Fig. 8.3 and 8.4 are shown the two groups of probabilities for different classification rates, ranging between $c^{HT} = 0.1$ and $c^{HT} = 0.5$, where $r^{ST} = 0.3$ is used. For values of the classification rate $c < r^{ST}$ the attacker has a higher confidence to use a ST_s instead of HT_s. For the system to be better protected the classification rate should be $c^{ST} = 1 - r^{ST}$, equal to 0.3 in this specific case.

In the case of observed honey templates, in both **Example 4** and **5**, it can be seen that $P(HT_o|HT_s)$ is higher than the baseline case, whereas $P(HT_o|ST_s)$ is lower. Now, again the defender system would be better protected if the classification rate c is less than 0.5 and the percentage parameter r^{ST} is equal to that specific value of c.

We conclude that the defender will be better protected if the system employs non-ideal classifiers. In fact the defender can improve the security compared to the

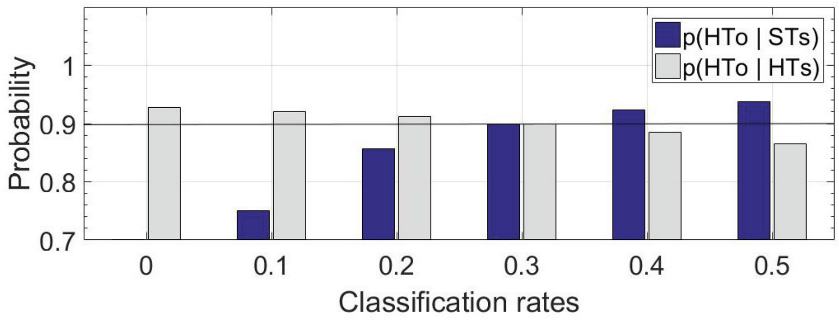


Figure 8.4: Comparison of probabilities $P\left(HT_o|ST_s\right)$ and $P\left(HT_o|HT_s\right)$ for a constant portion of sugar templates $r^{HT} = 0.7$ and different classification rates c^{HT}.

ideal case, so the answer to the question raised at the beginning of this section, related to real-life classifiers is that the defender system **is** better proteced if it deploys tuned generation of sugar and honey templates. The condition to achieve this is that the tuning parameter equals the classification rate.

8.4 Experimental setup and results: Algorithm VI

8.4.1 Defender/Attacker simulation

Databases

In order to illustrate the predictability methodology of biometric templates, experiments are carried out on *faces94* public database to simulate a Defender/Attacker scenario. This database comprises of three sets (*female*, *male*, *malestaff*). Two of the sets (female and malestaff comprising of 20 samples each) are equally divided between the two systems. Its third set (male, comprising of 113 subjects) is divided such that finally the defender database DB^D contains *70* subjects, and the attacker database DB^A comprises of 83 subjects.

In *faces94* database each subject has 20 samples, images background is plain green, there are minor variations in head turn, and there are considerable face expression changes. Their image resolution is 180×200 pixels. Before applying feature extraction images are processed further. They are converted in greyscale images, and having resolution of 92×112 pixels.

Feature vectors and templates

Figure 8.5: Samples of a subject in *faces94* database.

To generate the biometric features from each image, *Principal Components Analysis* (PCA) is used. It is a mathematical procedure that converts a set of correlated variables to a set of uncorrelated variables(55). It finds a reduced set of vectors which could serve as a basis from which all other vectors are generated as their linear combination. Face images are seen from the PCA technique under the same point of view. Their aim is to find the most informative base vectors, called *eigenfaces*, from which the PCA components for DB^A will be generated.

From experimental observation, in both systems, the number of PCA components bringing most of the information from image features is equal to 20. This means that biometric features vectors for sugar templates in both Defender and Attacker have real-number representations with dimensionality equal to 20.

The Defender system generates the *eigenfaces* by an auxiliary database DB^D_{aux}. It comprises of 40 subjects from publicly available images. The same does the Attacker. He owns DB^A_{aux} to generate his eigenfaces.

Honey features are generated by averaging a randomly picked number of *t*-images by respective databases DB^D and DB^A. On both sets of features, for both systems, the template protection mechanism of Random Projected is implemented (as described in Algorithms (1) and (2)).

Random projection is a technique which provides unlinkable templates by projecting a biometric feature vector in a set of orthonormal random vectors. To achieve this, the feature vector is multiplied by a random real-value matrix (111). In the Defender/Attacker simulation the dimensionality of the matrix is 20×40 meaning that the final representation of the templates will be real-number vectors of

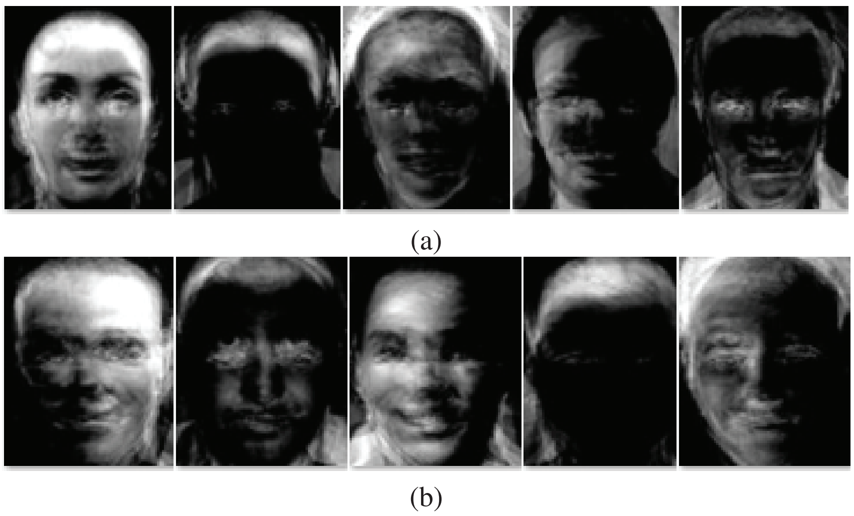


Figure 8.6: *Eigenfaces* from Defender (a) and Attacker (b).

dimensionality equals to 40.

In our experiments $k = 9$ honey templates are generated, meaning that in total 10 sweet templates for each sample. Thus, the biometric database of the defender contains 70 subjects $\times 20$ samples $\times 10$ sweet templates $= 1400$ templates, and that of the attacker 83 subjects $\times 20$ samples $\times 10$ sweet templates $= 1660$ templates.

8.4.2 Systems performance evaluation and results

Verification performance is analysed for both Defender and Attacker on:

A. *unprotected face verification system*

B. *Performance of protected face verification system, without honey templates*

C. *Performance of protected face verification system, with honey templates*.

The same protocol will be followed in all cases. The metrics used to evaluate the performance is Equal Error Rate (EER) which is a rate where false match rates are equal to false non match rates. The lower EER, the better is system recognition performing.

A. Performance for unprotected face verification system

In the unprotected system and without honey templates, genuine and imposter scores are computed as cross comparison between plain feature vectors of the same or different subjects, respectively. In Fig. 8.7 are shown performance curves of both systems. In Table 3, EERs values show that Defender and Attacker system have a similar recognition performance.

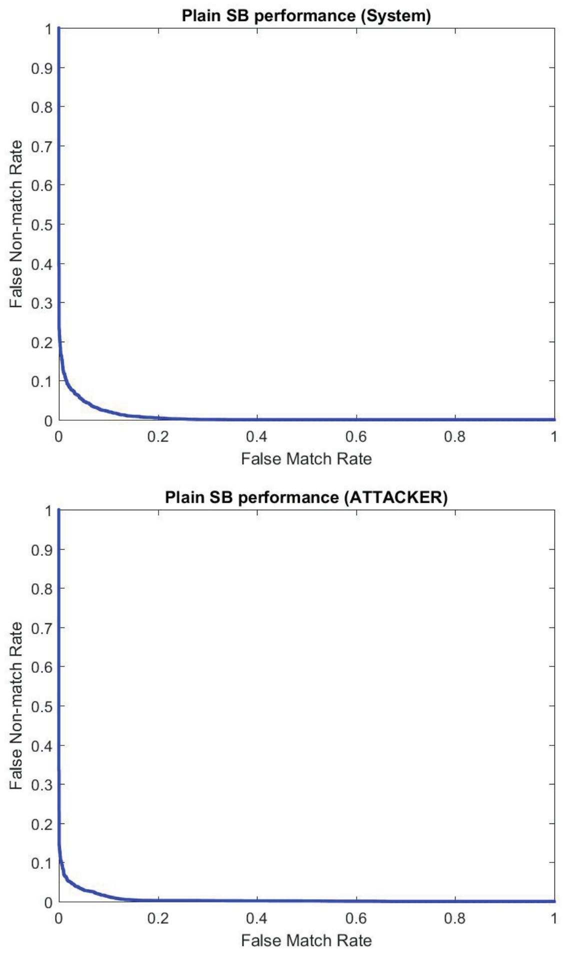


Figure 8.7: System recognition performance for Defender (up) and Attacker (down), after PCA feature extraction.

B. Performance of protected face verification system, without honey templates

In the protected system, and without honey templates, genuine and imposter scores are again computed as cross comparison between protected templates. In Fig. 8.8

Table 8.3: EER comparison after PCA feature extraction.

	Defender	*Attacker*
EER	0.0496	0.0377

are shown performance curves of both systems. In Table 4, EER values show similar recognition performances and a very low degradation compared to the unprotected system.

Table 8.4: EER comparison after Random Projection.

	Defender	*Attacker*
EER	0.0488	0.0388

C. Performance of protected face verification system, with honey templates

For the honey-based systems the probe sample is compared to the full list of sweet templates, taking the minimum Hamming distance as the final score. In Fig. 8.9 and from EER values in Table 5, it can be concluded a degradation of performance, due to the inclusion of honey templates.

8.4.3 Classification tests

In the process of honey templates creation the plain feature vectors are transformed according to the Random Projection scheme. But as a second approach here is implemented a machine learning mechanism to iteratively generate templates. In our case, Support Vector Machines (SVM) classifier is trained by labelling sugar templates as '1' and those honey ones as '0'.

We chose SVM because of its accuracy in high-dimensional data. While its training is ready, the SVM classifier C can be used to test a sugar template to see if it can be classified as a honey one (labelled as '0'). If yes, this template can be saved in the biometric database. Two more tests were conducted:

D. Classification test of sugar and honey templates in a protected face verification system

E. Classification test of sugar and honey templates in a protected face verification system, with iterative classifier

D. Classification test of sugar and honey templates in a protected face verification system

The SVM classifier was trained with 200 protected templates (100 from sugar templates, and 100 from first honey templates of respective sugar subjects). The testing set was the remaining database. The same classification design was implemented for the Defender and Attacker systems. Results are presented in Table 6.

Table 8.5: EER comparison after Random Projection with honey templates.

	Defender	*Attacker*
EER	0.0766	0.0530

Table 8.6: Classification rates of sugar and honey templates after Random Projection.

	Defender	*Attacker*
c	0.2377	0.2239

Table 8.7: Classification rates of sugar and honey templates after Random Projection.

p	*Defender* (r)	*Attacker* (r)
10	0.2328	0.1969
15	0.3695	0.3193
20	0.2926	0.2692
25	0.2699	0.2865
35	0.2740	0.2547
50	0.2037	0.2143
65	0.2685	0.3679
75	0.3114	0.2989
85	0.3142	0.2144
95	0.3651	0.2643

E. Classification test of sugar and honey templates in a protected face verification system, with iterative classifier

From experimental observations templates' re-generation varies between $1-8$ iterations. Templates are generated for different percentages of sugar templates: 10, 15, 20, 25, 35, 50, 65, 75, 85, and 95. Classification rates are evaluated similarly to the simple template generation case. Training set is not included in the remaining

Table 8.8: Simulated system probabilities showing attacker's confidence with c_A^{ST}.

Probability	*Value*
$P\left(ST_o \vert HT_s\right)$	0.0911
$P\left(ST_o \vert ST_s\right)$	0.1296
$P\left(HT_o \vert ST_s\right)$	0.8704
$P\left(HT_o \vert HT_s\right)$	0.9089

Table 8.9: Simulated attacker's confidences on sugar and honey templates, calculated for different portions of ST.

Portion (%)	$conf_A^{HT}$ (ST)	$conf_A^{ST}$ (ST)	$conf_A^{HT}$ (HT)	$conf_A^{ST}$ (HT)
10	0.1141	0.0000	1.0000	0.8748
15	0.1085	0.0473	0.9527	0.8859
20	0.1028	0.0903	0.9097	0.8972
25	0.0970	0.1296	0.8704	0.9089
35	0.0851	0.1656	0.8344	0.9209
50	0.0668	0.1988	0.8012	0.9332
65	0.0477	0.2294	0.7706	0.9458
75	0.0346	0.2578	0.7422	0.9588
85	0.0210	0.2842	0.7158	0.9722
95	0.0071	0.3087	0.6913	0.9859

testing templates, which are generated according to the iterative process. Results are shown in Table 7.

8.4.4 Experimental attacker's confidence

Considering Case B of real-life classifiers, the confidence of an attacker on trying sugar or honey templates can be illustrated by using results of the simulated systems. In fact, having $c_D^{ST} = 0.2377$ and $c_A^{ST} = 0.2239$ and also almost equivalent performance rates between the two simulated systems, this means that the attacker has realised his goal in building a similar Defender system.

Supposing the attacker has somehow gotten into possession of DB^D with 70 subjects having 20 samples, and 10 sweet templates each sample. Since he has no knowledge on sugar and honey templates, he can test them with his classifier C^A and evaluate confidence levels.

From Table 9, it can be seen that the most equilibrated probability values (i.e. closer to the baseline case) are for portion $q = 20\%$. This proves the theoretical results from Case B, since this portion is closer in value to the classifier's rate C^A.

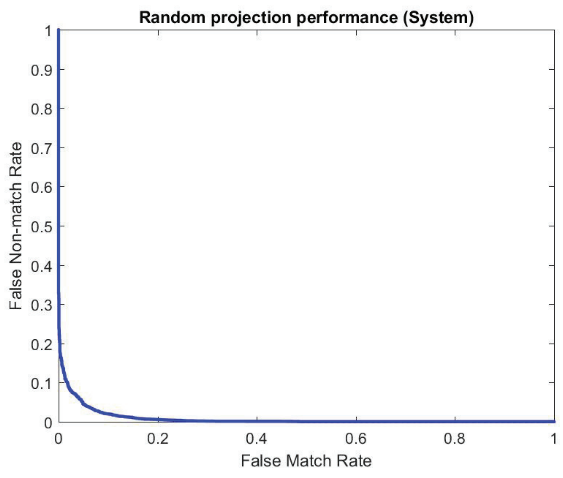
8.5 Conclusions

In this chapter we presented a methodology for the honey and sugar templates confidence. The proposed idea of honey templates, presented by (113), in 2015, describing the augmentation of a biometric database with *honey templates*. The aim of honey templates to camouflage the real template, may be circumvented if a knowledgeable attacker is able to distinguish honey from sugar templates found in a stolen biometric database. Even though the attacker is able to build and simulate a system very similar to the defender, the proposed methodology aims in improving the design of the Defender system.

There are two different kind of template generation schemes described in the chapter. Both of them use as a protection mechanism the Random Projection algorithm, which is a Feature Transformation BTP. The first kind is a simple honey template generator, and the second system generates templates according to a iterative classification process, which can be tweaked or not by a parameter q. The two main issues were: 1) will a defender be better protected if he artificially tunes the parameter q to tweak the distribution of observed sugar and honey templates, supposing both defender and attacker own an ideal classifier? And, 2) in real life applications where the classification rate is not ideal, will a defender be better protected when percentage is tuned?

8.6 Brief chapter summary

We prove in this chapter that a defender **is not** better protected when tuning the percentage of templates, and a defender **is** better protected when classification rates are not ideal, meaning that the methodology is applicable to real-life scenarios. Experimental results simulated a face verification biometric system, where a Defender/Attacker scenario is implemented in order to illustrate and prove theoretical results.



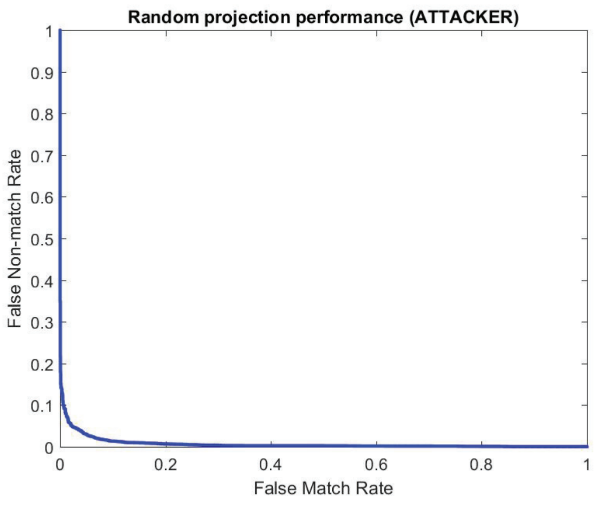


Figure 8.8: System recognition performance for Defender (up) and Attacker (down), after Random Projection.

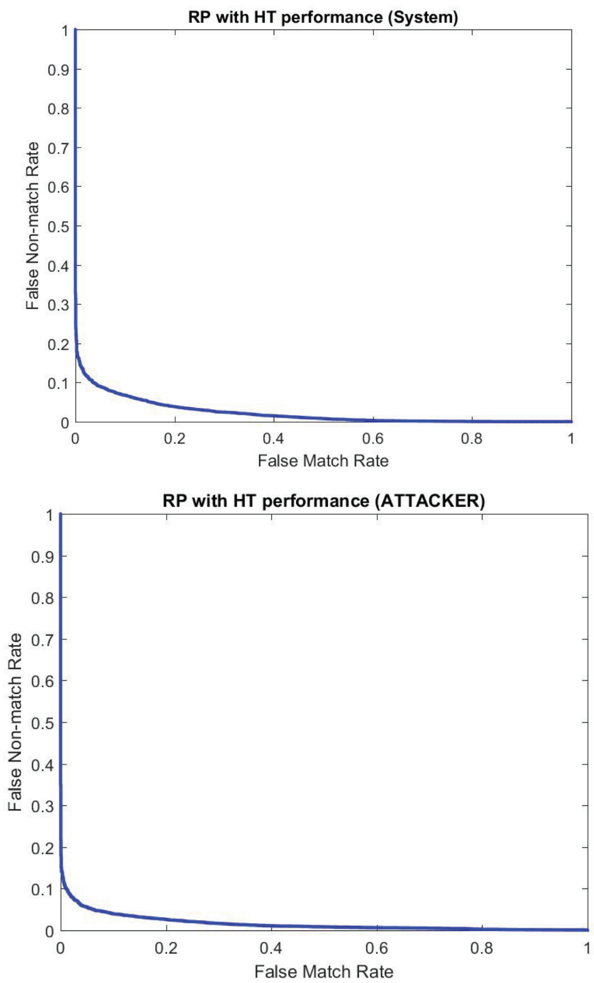


Figure 8.9: System recognition performance for Defender (up) and Attacker (down), after Random Projection, with honey templates.

Chapter 9

Thesis Conclusions

This thesis explored the possibility of augmenting a biometric system with synthetic templates in order to camouflage the real ones. Not only we aimed in proving the soundness of this newly presented technique, but we also included new protection functions, hybrid schemes, two-level security analysis, and mainly a new concept on synthetic templates generation, on how to make the system learn from the process. We can truthfully say this work presents an interdisciplinary attempt in the long scientific endevour which hopefully improves the security of biometric systems and contributes to other topics within information security too.

9.1 Thesis overview

Fig. 5.1 shows the architecture of a biometric system augmented with the honey templates approach, which is applicable to verification scenarios. At the enrolment phase (Fig. 5.1 left), features are extracted from the input biometric sample to generate the plain feature vector $\mathbf{T}_i$. The Application server then creates the sugar template and a set of honey templates (see Fig. 5.2 and Sect. 5.2.2), which are generated as random binary templates. To make them indistinguishable in the feature domain, the average of non-zero elements of real samples is computed (defined as *sparsity* $\hat{S}$), and then the same $\hat{S}$ is used for the k-randomly generated honey feature vectors. The protection mechanism is applied on both type of vectors, and as a result, a set of protected sweet templates is obtained: $PT_i = (AD_i, PI_i^1, PI_i^2, \ldots, PI_i^{K+1})$, where the auxiliary data AD_i is used to generate the sugar template's index $L_i \in [1, \ldots, K+1]$. The indices of the protected honey templates will be randomly allocated in the remaining memory space available for each enrolled subject i.

It should be noted that replay attacks (alias presentation attacks) cannot be carried out directly on the sensor: the stored references are irreversible templates from which the original biometric sample cannot be recovered. As a consequence, in order to perform a reply attack the eventual impostor should inject the intercepted protected template in the signal processing subsystem, specifically in the channel between the feature extractor and the Bloom filter template module (see Fig. 5.2), thus breaking the integrity of the system.

That way, an eventual attacker must compromise both systems to achieve his goal: identify the sugar template among the honey templates. The main advantage of the honey concept thus relies on the fact that the data that the system needs to secure is reduced to the index L_i, which is very concise.

After enrolling the subject, at the verification phase (see Fig. 5.1 right), a protected template PI^* is extracted from the newly presented plain feature vector $\mathbf{T}_i^*$. After a handshake sequence between the Application Server and the Biometric Database Server, the new template will be compared with the whole set of sweet templates retrieved from the Biometric Database. The best-match template index id_{xi} from this set will be compared to the index L_i retrieved from the Honey Checker Database:

- If $L_i = id_{xi}$, the identity claim is accepted and access to the system granted.
- If $L_i \neq id_{xi}$, the template matched a honey template and the system will hence consider this attempt as information leakage. As a consequence, an alarm will be set off, and the specific regulation, based on the system's security policy, will be followed (e.g., a partial/complete update of the templates).

Their main security requirement is that they should be *indistinguishable* from real templates.

Chapter 2

In this chapter we provide an overview of honey objects and their application, offering a generalized architecture, applicable at data level (files, passwords, templates, etc). We review this property under a cryptographic context, by drawing parallel lines between cryptosystems and honey-based systems. In fact, honey objects must comply to two main properties: (1) **indistinguishability**, honey and real objects must be hard to distinguish from each other (e.g. a real password from a generated password or a database entry of a real patient in a health system from a fake one); (2) **secrecy**, the real object must be secret and camouflaged among the honey objects. In the case of honeywords, if an intruder by some means gets access to the user' set of passwords, he can use/guess only one of them. The system intercepts that a honeyword is used, it will consider this as information leakage and proceed with further steps (set off an alarm and/or update the passwords set).

Chapter 3

In this chapter we provided an explanation of the honey-based idea applied in a biometric context. We applied the experiments on a new BTPS, on a verification system based on faces. A large set of tests was performed for both criteria of irreversability and recognition performance (including feature vectors, protected templates, and honey templates. This was the first proof-of-concept that honey templates are applicable in a new BTPS.

Chapter 4

In actual biometric verification systems, if an attacker steals the content of the biometric database, he could easily link the template to the subject, since they reside in the same space, and then he might reconstruct the biometric features from the protected templates by breaking their protection algorithm. In this Chapter we aim to break this link between the subject (and their respective $PersonalInformation$) and the biometric content the system holds, by designing a $game$ between the system $Defender$ and the $Attacker$ to better evaluate the later's efforts in being successful in his/her attempts. An attacker -except from the leaked database- should also own (or design, simulate, create) two separate sets of templates (honey and real) similar in terms of data parameters with those found in the leaked database, in order to build a reliable and accurate classifier for the division of the new set in two classes. In this chapter we designed the Defender system D, considering that the attacker is highly knowleadgeable about D and owns different classification tools, we concluded that an attacker can mimic D.

Chapter 5

In this chapter, a general biometric template protection scheme based on Honey Templates and Bloom filters is proposed, in order to grant privacy protection to the enrolled subject and detect the use of stolen templates. It covers one other implementation of Honey Templates among the Feature Transformation mechanisms. The performance and security evaluations show the soundness of the proposed scheme for facial verification. The benchmark is conducted with the publicly available BioSecure Multimodal DB and the free Bob image processing toolbox, so that research is fully reproducible.

Chapter 6

In this chapter is presented an application of Biometric Honey Templates in a hybrid protection setup, comprised of three components which generates synthetic (honey) templates by preserving the indistinguishability property. While in previous chapters we applied Random Projection and Bloom Filters categorized as *Feature Transformation* schemes where Honey Templates can be adopted, in this chapter we add *Fuzzy Commitment*, one of the protection mechanisms part of the *Biometric Cryptosystem* category.

Considering an attacker having full knowledge of the system, a face verification system and a *testing framework* are implemented. Results from the tests showed that they have a high level of dissimilarity, by preserving the system recognition performance. Another contribution to this work is the *Theorem of Augmentation*. It evaluates the maximum number of honey templates with which a system can be augmented.

Chapter 7

We here provided the benefits that could bring the adoption of honey templates as a security mechanism together with PCA. It can be used in template protection design as a feature transformation scheme, but, by using this technique, a possible attacker could reconstruct the templates and try to define if a given pre-image could be a sugar template, or one of the honey templates. For this reason, template indistinguishability has to be measured and strong metrics must be set off for practical implementation. To provide sound result we used two classification technique: one based on human classification and one on automatic classification. For the first type we constructed a platform where human testers could classify a set of random pre-images coming from sugar or honey templates. We compared the obtained result with an SVM classifier. We noticed that, compared to the SVM classifier, human testers showed better results in terms of classification distinguishability and from an attacker point of view. This means that with a trained subject an attacker could improve its chances of guessing a sugar template among a set of honey templates, after the pre-image classification.

Chapter 8

In this chapter we presented a methodology for the honey and sugar templates confidence. Even though the attacker is able to build and simulate a system very similar to the defender, the proposed methodology aims in improving the design of the Defender system. There are two different kind of template generation schemes described in the chapter. Both of them use as a protection mechanism the Random Projection algorithm, which is a Feature Transformation BTP. The first kind is a simple honey template generator, and the second system generates templates according to a iterative classification process, which can be tweaked or not by a parameter q. The two main issues were: 1) will a defender be better protected if he artificially tunes the parameter q to tweak the distribution of observed sugar and honey templates, supposing both defender and attacker own an ideal classifier? And, 2) in real life applications where the classification rate is not ideal, a defender will be better protected when percentage is tuned.

9.2 Research Questions Verification

In Chapter 1 we listed the research questions of the thesis. We bring them again here as they are.

- **RQ1.** Can Biometric Template Protection schemes be augmented with Honey Templates, that are indistinguishable from real ones and that has information leakage capabilities?

  RQ1.1. Does this augmentation deteriorate the system performance?

  RQ1.2. Are Equal Error Rates influenced in the same way for different schemes?

  RQ1.3. Can the indistinguishing property between templates be measured with reliable metrics?

  RQ1.4. Is there an upper limit in the number of Honey Templates that can be augmented in different schemes?

  RQ1.5. Is it possible for an attacker to mimic a honey-based system in terms of its parameters, and use it to classify leaked templates?

- **RQ2.** Can we provide a mechanism for the Honey Templates generation that learns and improves the templates indistinguishability property?

- **RQ3.** If face biometric templates, real or honey, were reconstructed and their images were tested to be distinguished by automatic classifiers or human testers, who would be more accurate?

- **RQ4.** Can we assert that Honey Templates are applicable in real-life scenarios?

- **RQ5*.** Can the mechanisms used in this thesis on biomtric templates be extended to other access control mechanisms: such as passwords, from where it emerged?

Throughout the chapters in the thesis and in the appendix we could answer them by means of extensive experimental frameworks.

Research Question 1.

In chapters 2, 3, 5, and 6 we provided an explanation of the honey-based idea applied in a biometric context. We can answer as Yes to **RQ1**, and assert that Biometric Template Protection Schemes can be augmented with Honey Templates, indistinguishable from real ones. We applied the experiments on a new BTPS, on a verification system based on faces, which comprises another contribution to this thesis. A large set of tests was performed for both criteria of irreversability and recognition performance (including feature vectors, protected templates, and honey templates.

Research Question 1.1

Performance is an important aspect of this work. Starting with Chapter 2, where a mask vector is applied as a BTP, the results from the tests show that the system performance is deteriorated with the honey templates augmentation, but this level is acceptable compared to the overall effect that BTP schemes tend to give in this aspect. Regarding **RQ1.1**, we answer that, yes, honey templates influence the system performance in this case (EER increases are of a minor influence). Whereas in Chapter 5 Honey Templates do not deteriorate the system performance. In this case the template protection mechanism are Bloom Filters.

Research Question 1.2

To state if EER are influenced in the same way for different schemes, we here provide a comparison of test results for all of the BTPs implemented in this thesis.

Algorithm I - Mask Vector Here honey templates are generated by first creating random feature vectors and then applying the BTP. Plain feature vectors are generated from a helping database of 40 public face images.

Performance evaluation. Since the scheme offered a variety of parametric results, the best system performance augmented with honey templates was found for $EER = 0,1285$.

Classification results. The best classification results were $FMR^{SVM} = 0,3433$

and $FNMR^{SVM} = 0,4050$.

Bloom Filters Tests were conducted on BioSecure Multimodal Database (Desktop dataset) (25) with 210 subjects having 4 samples each. For each sample 10 honey templates were generated. as random binary feature vectors. The size of the final templates is a binary matrix having dimensions 1024×32. The main challenge of this work was classification of templates. There were three different classifiers trained and tested after a feature selection process.

Performance evaluation. The system performance when augmented with honey templates was $EER = 0,063$.

Classification results. Three classifiers were used: *Support Vector Machine*, *Linear Regression*, and *Quadratic Discriminant Analysis*. Results are shown in Table 9.1, together with system performances.

Table 9.1: Performance results after Random Projection (Level II, left side), and after Binarization (Level III, right side) for 5 rounds.

	Mask vector	**Bloom Filter**
Performance	0.1285	0,063
$mean_{Diff}$	*0.13 (SVM)*	*0,08 (SVM)*
		0.12 (LR)
		0.10 (QDA)

And as we could get from Table 7.3:

Table 9.2: Classification rates for tests 6, 7, and 9. (Level II and III)

Test no.	FMR^{SVM}	$FNMR^{SVM}$	$mean_{Diff}$
6	0.1027	0.6850	0.2912
7	0.1036	0.6217	0.2591
9	0.5142	0.4967	0.0123

If we compare classification results of previous honey templates implementation, there is an improvement of $mean_{Diff}$ of the Hybrid scheme. From Table 9.1 and Table 9.2 we have respectively *0.13* (**Mask vector**), *0.08* (**Bloom Filter**'s best rate from SVM), and *0.0123* (**Hybrid** scheme), resulting in more indistinguishable templates. We can conclude that performances of systems augmented with honey templates are not influenced the same.

Research Question Q1.3
From Chapter 2 and ongoing, we set the basis a set of metrics, necessary in verifying if the indistinguishability property. To our work, generating honey templates that are indistinguishable from real ones, means that the probability of guessing a real template from a synthetic one is equal to 0.5. Or, it means that classification tools are not able to differentiate them. So, if in other classifications in general researchers tend to have clear class separations, in this work classifiers should ideally provide a value close to 0.5.

Another contribution to RQ1.3 is presented in Chapter 3. Here we defined imporant terms that add to the metrics apparatus regarding honey templates, specifically the rates FSTR and FHTR, making the answer to the research question **RQ1.3** more complete. The values of these metrics show the attacker's and defender's confidence in their systems, allowing us to better define in the future our protection mechanisms. And finally, the property of indistinguishability is measured by the mean-of-differences metrics, presented in Chapter 3.

Research Question 1.4

In the first two algorithms presented in this thesis (Mask Vector and Bloom Filters) the number of honey templates is limited only by the system memory capacity and processing performance. Whereas in Chapter 6 regarding **RQ1.4**, we answer as Yes, there is an upper limit in the number of Honey Templates that can be augmented in this scheme. We provided the Theorem of Augmentation that proofs the number of honey templates that the ECC can handle.

Research Question 1.5
Chapter 3 extensively relates to this research quetion. From the Attacker-Defender game we simulated, we answer as Yes, it is possible for an attacker to mimic a honey-based system in terms of its parameters, and use it to classify leaked templates. The level of confidence of the attacker in the classification framework depends on the training sets, and algorithms.

Two "opposite" system perspectives are implemented and analyzed: a Defender system *D*, and an Attacker system *A*. The first aim of A is to mimic a targeting biometric system $BioSys^D$ knowing it is augmented with honey templates. Supposing the attacker knows well the system design, algorithm parameters, and owns different databases, his second attempt is to build a classifier which will enable him to distinguish the two categories of templates.

To answer if an attacker system is able to mimic D and classify templates two face verification biometric systems are implemented to simulate a Defender/Attacker scenario, and a generic multi-classification scheme (with 12 different classifiers). Following a large set of tests on both systems, we could conclude that an Attacker can be successful in mimicking the proposed Defender biometric system. He can build such a system with performance recognition rates similar enough to $BioSys^D$, and also a complex multi-classifier. While he is successful in his first aim, achieving a *system indistinguishability* of $\varepsilon_r = 1,4 \times 10^{-3}$ and even though he tries all the known parameters in the classification mechanism, he still has a low confidence in honey templates indistinguishability.

Research Question 2

There are two different kind of template generation schemes described in the proposed algorithms. Expect the Mask Vector BTP, two more mechanisms are implemented. Both of them use as a protection mechanism the Random Projection algorithm, which is a Feature Transformation BTP. The first kind is a simple honey template generator, and the second system generates templates according to a iterative classification process, which can be tweaked by a parameter. All results show that the solution learns from the generated templates, and can improve the indistinguishability of generated templates.

Research Question 3

We provided the benefits that could bring the adoption of honey templates as a security mechanism in biometric systems. For this research question are analyzed the different threats and system vulnerabilities and provided how these could be relieved by using honey templates. We argue that PCA can be used in template protection design as a feature transformation scheme, but, by using this technique, a possible attacker could reconstruct the templates and try to define if a given pre-image could be a sugar template, or one of the honey templates. For this reason, template indistinguishability is measured and strong metrics must be set off for practical implementation.

To provide sound result we used two classification technique: one based on human classification and one on automatic classification. For the first type we constructed a platform where human testers could classify a set of random pre-images coming from sugar or honey templates. We compared the obtained result with an SVM classifier. We noticed that, compared to the SVM classifier, human testers showed better results in terms of classification distinguishability and from an attacker point of view.

This means that with a trained subject an attacker could improve its chances of guessing a sugar template among a set of honey templates, after the pre-image classification. The accurate tester we had with the minimum error rate equal to 0,23 and other testers who achieved a similar result could be subject to further studies regarding image perception and human memory classification processes.

Research Question 4

For the verification of RQ4, we present a methodology for the honey and sugar templates confidence. The two main issues were: 1) will a defender be better protected if he artificially tunes the parameter q to tweak the distribution of observed sugar and honey templates, supposing both defender and attacker own an ideal classifier? And, 2) in real life applications where the classification rate is not ideal, will a defender be better protected when percentage is tuned?

We prove in this work that a defender **is not** better protected when tuning the percentage of templates, and a defender **is** better protected when classification rates are not ideal, meaning that the methodology is applicable to real-life scenarios. Experimental results simulated a face verification biometric system, where a Defender/Attacker scenario is implemented in order to illustrate and prove theoretical results.

Research Question 5*

In Appendix A, we analyzed the feasibility of PassGAN for honeywords generation method. Besides, we also investigated some possible strategies that can be used by the attacker and the defender in a PassGAN based honeywords system. The defender uses the generator model of PassGAN to generate high-quality fake passwords while the attacker is assumed can manage to steal and crack all the hashed password data and then uses the discriminator model of PassGAN to distinguish the real passwords from the fake ones. Based on the experiment results, PassGAN is feasible for honeywords generation strategy. The best strategy that can be used by the attacker is to use the same dataset as the defender 's dataset. Besides, the attacker can also use a large number of iterations as the strategy. The more number of iterations is proven to be able to increase the attacker 's success rate even though it is very small. From the defender's perspective, the strategy of using a large number of iterations also benefits the defender to reduce the attacker 's success rate. We could so, integrate successfully some of the concepts we applied in biometric templates to passwords.

9.3 Future work

In the future works, other several strategies for the attacker and the defender can also be examined. Despite its ability to generate high quality artificial templates in some of our algorithms, there are still many building blocks of this system that needs to be thoroughly tested.

Honey templates has a practical advantage, it can be applicable to verification applications in physical and logical access control, such as ATM, health records, etc. An important part of our work was to provide the security evaluation of our BTPS, focused on the irreversibility of the biometric sugar and honey templates. The low levels of correlations between different feature or template sets show the effectiveness of the scheme.

As a further work we will continue with the system evaluation in terms of unlinkability, as the difficulty of classifying the protected templates over time and across applications [16]. Inuma in [18] has mathematically proven the relationship between the two notions of irreversibility and linkability in a biometric system. We proved in our work that if an attacker possesses the set of protected sugar and honey templates, he is not able to recover the feature element and then pretend to be that user. But what if he possesses two sets of protected templates of user i, coming from two different applications: is the attacker able to identify that these template sets belong to the same characteristic? This process should be computationally hard and we expect to evaluate the notion of linkability in our future work.

We can finally conclude that while use of honey templates in biometrics is still a new direction to explore, we believe both the BTPS method and the honey template construction method have wide room to improve in security and recognition performance aspects in the future. We hope this work can provoke thoughts and discussions in this field.

Appendix A

PassGAN for Artificial Passwords Generation

The main challenge in a honeywords system is how to generate artificial passwords (honeywords) that are indistinguishable from the genuine ones (sugarword). It is straightforward to consider the PassGAN technique for generating honeywords from the defender's perspective. In this chapter, we analyze a game situation between the defender and the attacker assuming the two parties exploit the PassGAN for their own competing advantage, i.e., the defender uses the generator model of PassGAN to generate honeywords and the attacker uses the discriminator model of PassGAN to detect the sugarword. In this game, we investigate the feasibility of PassGAN as a honeywords generation strategy and the possible strategies that can be used by the defender and the attacker to reach their goal.

A.1 Introduction

Password remains the most widely used identity authentication method due to its simplicity and familiarity to users and developers (108). Unfortunately, there have been many password data leaks recently including data from well-known organizations like Rock-you (96), Dropbox (43), Yahoo (42), etc. Even worse, the leaks show that most users prefer poor passwords for their accounts that enable attackers to guess them easily using some cracking techniques such as dictionary attack despite they are saved in the hashed form (32, 34). Most of these breaches were often only discovered months or even years after it first happened. During the period, the attacker surely had exploited most of these passwords, some even had published or sold it online. Therefore, it is prominent to have not only a mechanism to improve security but also a mechanism to detect the password data leaks quickly (33).

Some approaches have been proposed to handle the password data breach (12, 84). A more promising one is introduced by Juels and Rivest called honeywords (59). In this system, the mixture of real passwords (sugarwords or sugars) and decoy passwords (honeywords or honeys) are stored in the password file. If an attacker manages to steal the file containing the password and successfully resolves all hash values in the file, she or he still has to be able to distinguish the real password from the artificial ones. When an attacker tries to log in using a honeyword, the system will detect and provide an alarm to denote that there is an attack on the password file. This technique is relatively more practical because it only needs a few changes on the system, both server and client side (36).

The main challenge for the system administrator in this approach is how to generate honeywords that are difficult to distinguish from the real ones. Some prior methods for honeyword generation include random-replacement (59), utilizing existing passwords (30), questionnaire-based (22), text and image-based non-realistic honeyword generation (7), and paired distance protocol (PDP) (21). One of the newest potential methods that can be utilized is PassGAN (45).

PassGAN is designed to be a password guessing method (45) that utilizes a Generative Adversarial Network (GAN) (39) to learn from the real passwords as training data and generate password guesses that have high similarities as the real passwords. PassGAN does not require initial knowledge or intuition from experts about what type of passwords often chosen by users since it will autonomously learn from the real passwords. Besides, PassGAN for honeywords generation is classified as a legacy-UI method because it does not require user intervention. This kind of method is preferred due to usability advantage (59). Following this idea, PassGAN can be a good tool for honeyword generation from the defender's perspective.

On the other hand, attackers are also getting smarter and trying to find ways to pick the right password. The attackers could use some password guessing tools and machine- learning techniques to distinguish the real passwords from the decoys. The worst case for the defender is when the attacker also uses PassGAN to determine the correct password. This condition raises a question of how is the feasibility of PassGAN as a honeyword generation strategy and what are the best strategies for the defender and the attacker in this situation.

In this chapter, we assume that the attacker has managed to crack the hashed password file that is leaked from the defender, and could use PassGAN to distinguish between the sugarword from the honeywords. We will analyze the feasibility of PassGAN for honeywords generation in this case. We will also investigate some strategies that can be used by the attacker to make this distinguishing process more accurate and some strategies that can be used by the defender (e.g., the system administrator) to better secure their honeywords system against this kind of attack.

Most previous researches on honeywords (e.g. (30, 22, 21, 59)) only evaluate the honeywords generation strategies heuristically. In this study, we will conduct an evaluation of PassGAN for honeywords generation empirically with some near real-world scenarios and use datasets from some real systems (e.g. leaked password data).

A.2 Background and Related Works

A.2.1 PassGAN

PassGAN is a deep learning based password guessing method developed by Hitaj et al. (45). PassGAN uses Generative Adversarial Network (GAN) to autonomously learn from the real password dataset and then generate some passwords that have the similar distribution to the dataset. GAN is a deep learning approach introduced by Goodfellow et al. (39) for estimating generative models through an adversarial process.

GAN comprised of two neural networks that are pitted against each other: a generative model G and a discriminative model D. G studies the distribution of training data and creates new data instances or samples with similar distributions, while D compute the probabilities whether each sample is from the real training data or generated by G. G and D are trained simultaneously where the goal of G is to maximize the probability of D making a mistake while in contrast, D aims to successfully detect the fake samples. Competition in this game drives both G and D to improve their methods until the generated samples cannot be distinguished from the real data. Recently, there are many methods to improve GAN. One of the promising ones is IWGAN (40) due to its ability to make training more stable. IWGAN also successfully implemented for text generation and gives improvements in the performance.

In the PassGAN, a generator model is trained to learn about the characteristics and structures of passwords from training data (e.g. leaked password data). The generative model then generates some fake passwords based on the training model. At the same time, the discriminator model is simultaneously trained to distinguish the real password from the fake ones.

A.3 Our Work: Algorithm VII

A.3.1 Game Situation

In this work, we simulate a game between the defender and the attacker. As seen in Figure A.1, three datasets are used for the game: the defender's dataset for her or his PassGAN training, the attacker's dataset for her or his PassGAN training, and the system dataset as the real passwords (sugarwords). The game flowchart can be divided into two parts: the defender's work and the attacker's work.

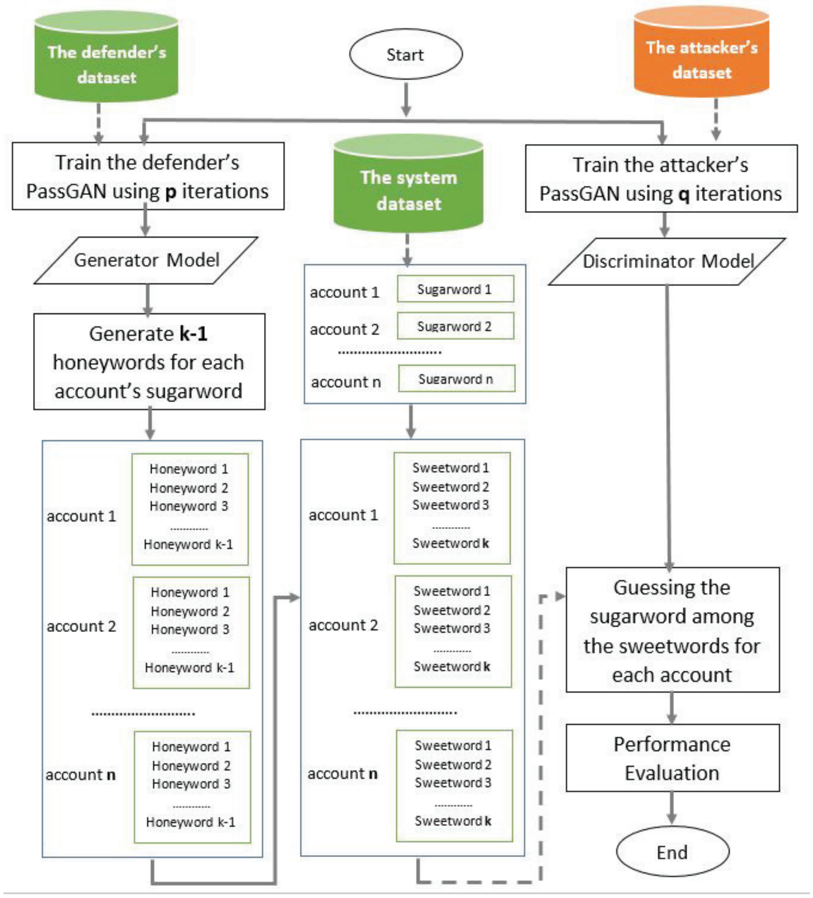

The defender's dataset
Start
The attacker's dataset
Train the defender's PassGAN using p iterations
The system dataset
Train the attacker's PassGAN using q iterations
Generator Model
Discriminator Model
account 1 Sugarword 1
account 2 Sugarword 2
account n Sugarword n
Generate k-1 honeywords for each account's sugarword
account 1 Honeyword 1 Honeyword 2 Honeyword 3 Honeyword k-1
account 2 Honeyword 1 Honeyword 2 Honeyword 3 Honeyword k-1
account n Honeyword 1 Honeyword 2 Honeyword 3 Honeyword k-1
account 1 Sweetword 1 Sweetword 2 Sweetword 3 Sweetword k
account 2 Sweetword 1 Sweetword 2 Sweetword 3 Sweetword k
account n Sweetword 1 Sweetword 2 Sweetword 3 Sweetword k
Guessing the sugarword among the sweetwords for each account
Performance Evaluation
End

Figure A.1: The attacker and the defender game flow.

The defender and the attacker train their own PassGAN model separately using their own dataset. The defender uses p iterations to train her or his PassGAN while the attacker uses q iterations to train her or his PassGAN. Both the attacker and the defender can use any number of iterations for their training process as it is one strategy that can be exploited by each party. After the training complete, the defender takes the generator model of her or his PassGAN, which is originally comprised of generator and discriminator model, and then uses it to generate $k-1$ honeywords for each account's sugarword in the system dataset. These generated honeywords are then combined with each sugarword from the system dataset to compose k sweetwords for each account.

Meanwhile, the attacker takes the discriminator model of her or his PassGAN and then uses it to determine the sugarword among the sweetwords for each account. The guessing process works as follows. For each account, the attacker uses the discriminator model to compute the probability of each sweetwords becoming a sugarword. Then, the sweetwords are sorted in a descending order based on the probability value. The sweetword with the highest probability is then considered as the real password of the account and will be submitted to the system by the attacker.

The guessed passwords are then evaluated. The attacker is considered successful if she or he can guess the correct password for each account. Otherwise, the attacker is considered fail. The success rate of the attacker will be used to evaluate both the attacker and defender strategies.

The objective of the defender in this game is to generate indistinguishable honeywords that can make the attacker's success rate low. On the contrary, the attacker aims to get the best discriminator performance so that it can maximize the probability of guessing a correct password and get a high success rate. In this game situation, we analyze the best strategy the two parties can exploit for their own competing advantage.

A.3.2 Dataset

Datasets used for this work are leaked passwords from Rock-you (16), Dropbox (2), and Linkedin (3). The Rock-you, Dropbox, and Linkedin dataset contains 21,315,673, 7,884,855, and 10,000 passwords respectively. These three passwords dataset are quite similar because they are leaked from online media platforms that have similar users, from various professions and mostly aged 18-34, and the rule for the password in these three datasets is very similar, e.g., there is no need for special characters or a combination of upper and lower case letters.

A.3.3 Experiment Setup

In this experiment, we have some assumptions as follows:

- The defender uses the generator model of PassGAN to generate honeywords.

- The defender uses public password dataset (e.g. leaked passwords) as her or his training data.
- The attacker manages to steal the data and crack all the hashed passwords.
- The attacker uses the discriminator model of PassGAN to guess the sugarword.
- In the real world, the attacker most likely does not know and does not have access to the defender's training dataset. Therefore, the attacker is likely to use a different dataset from the defender's dataset for the PassGAN training. However, we also conduct an experiment when the attacker and the defender use the same dataset to simulate a worst case for the defender.

The dataset used by the defender in this work is the Rock-you (RY) dataset while the attacker uses the Dropbox (Db) dataset. In a scenario when the attacker and the defender use the same dataset, the RY dataset is used for the experiment. The Linkedin (LI) dataset containing 10,000 account's real password is used as the system dataset's sugarwords ($n = 10000$). For each account's sugarword, 9 honeywords are generated and then they are combined to compose 10 sweetwords ($k = 10$).

The following is the detail of the two scenarios:

- Scenario 1: The defender and the attacker use the same number of iterations ($p = q$). A various number of iterations are used for the experiment ($p = q = \{5000, 10000, ..., 195000\}$). In this scenario, two dataset variations (both parties use a different dataset and both parties use the same dataset) are also examined.
- Scenario 2: The attacker and the defender use a different number of iterations. The attacker uses a fixed number of iteration ($p = 100000$), while the defender uses several numbers of iterations ($q = \{5000, 10000, ..., 195000\}$). In this scenario, two dataset variations (both parties use a different dataset and both parties use the same dataset) are also examined.

A.3.4 Performance Evaluation

Juels and Rivest (59) proposed flatness measurement to evaluate honeywords generation strategies. A honeywords generation method is called ϵ-flat when the attacker that is given a one-time opportunity to choose a correct password has a maximum success rate ϵ. A generation strategy is called "perfectly flat" if the attacker's maximum success rate $\epsilon = 1/k$. If the success rate ϵ is not much greater than $= 1/k$, it is called "approximately flat". Therefore, the goal of both the defender and the attacker in this experiment is measured using the attacker's success rate. The formula is as follows:

$$\epsilon = \frac{NoC}{NoA} \tag{A.1}$$

where ϵ is the attacker 's success rate, NoC is the number of accounts whose passwords have been guessed correctly and NoA is the total number of accounts. In this experiment, since the number of sugarwords used is 10,000, then the number of accounts (NoA) is always 10,000. Besides, we will also calculate the attacker's success rate when the attacker is given the opportunity more than once to choose the correct password in case there are honeywords systems that allow more than one guesses.

A.4 Experiment results and conclusions

Scenario 1

The experiment results from scenario 1 are shown in Figure **??**. Generally, the attacker's success rate in scenario 1 tends to be very low. It means that the generated honeywords are hard to distinguish from the sweetwords so that the attacker discriminator model can only guess a few numbers of passwords correctly. Based on Figure A.2, as expected, the attacker's success rate given only one guess when both parties use the same dataset is higher by almost 0.1 than when they use a different dataset. The use of the same dataset makes the attacker's discriminator model can learn from the same defender's dataset so that it can distinguish the honeywords better. When the attacker and the defender use a different dataset, the success rate is relatively stable even though the number of iterations increases. It only fluctuates in a very small margin between 0.1 and 0.15. Meanwhile, when the attacker and the defender use the same dataset, the success rate increase as more iterations used. The highest success rate (0.21) is obtained when both parties use 195000 iterations.

Based on Figure A.3, the success rate when both parties use the same dataset is also higher than the use of different datasets. However, the difference is relatively small and the success rates in these two conditions are not much greater than the "perfectly flat" method. Therefore, this honeywords generation method can be considered as "approximately flat".

Scenario 2

The experiment results from scenario 2 are shown in Figure A.4 and Figure A.5. Generally, the use of a different number of iterations does not show significant

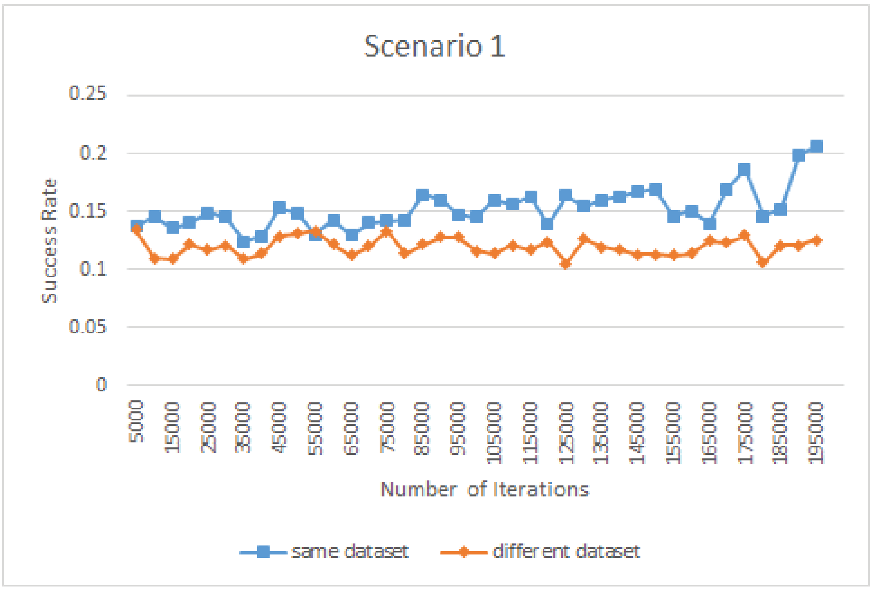


Figure A.2: The attacker's success rate given only 1 opportunity to guess the correct password in scenario 1. In this figure, the attacker and the defender use several iteration variations.

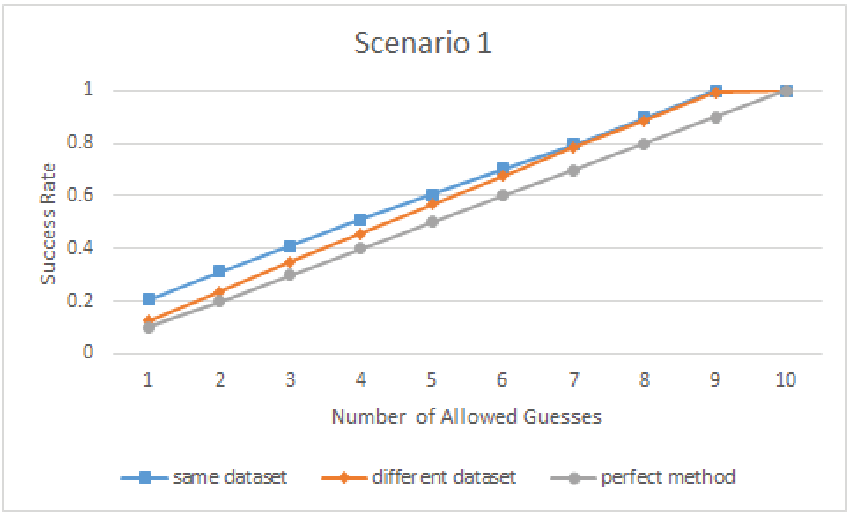


Figure A.3: The attacker's success rate given more than 1 opportunity to guess the correct password in scenario 1. In this figure, the defender and the attacker use 195000 iterations.

differences as the attacker's success rate in scenario 2 also tends to be very low. Based on Figure A.4, the use of the same dataset also give a higher success rate for the attacker than the use of a different dataset. In the same dataset condition, the success rate decrease, even just a little, as the number of defender's iterations increase and become larger than the attacker 's number of iterations. Meanwhile, when the attacker uses a different dataset from the defender 's dataset, the success rate is relatively stable as it only fluctuated in a very small margin.

Based on Figure A.5 and Figure A.6, the use of a different number of iterations by the defender does not show significant success rate differences. The highest success rate is obtained when the defender use 5000 iterations (less than the attacker's number of iterations) and the lowest success rate is obtained when the defender uses 195000 iterations (more than the attacker's number of iterations). However, the difference is hard to be noticed because it is very small.

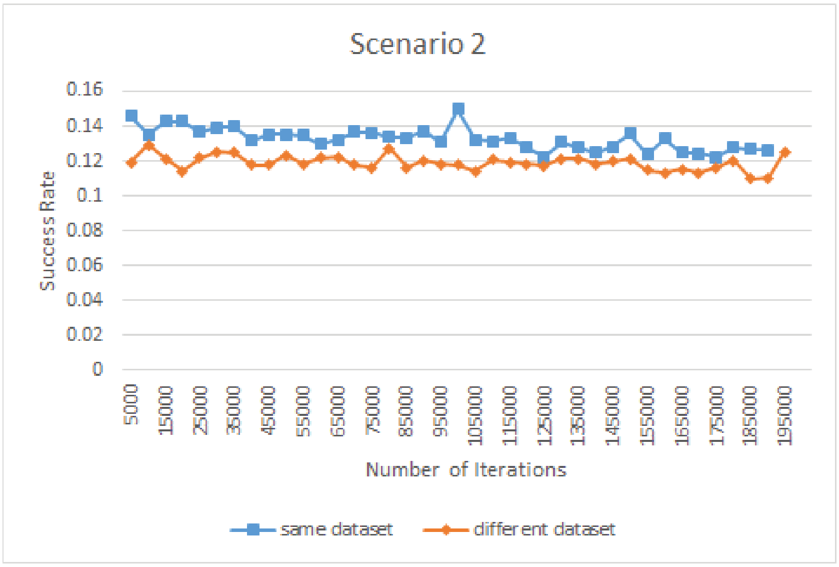


Figure A.4: The attacker's success rate given only 1 opportunity to guess the correct password in scenario 2. In this figure, the attacker uses a fixed number of iterations (100000), while the defender uses several numbers of iterations.

A.5 Brief chapter summary

In this chapter, we analyzed the feasibility of PassGAN for honeywords generation method. Besides, we also investigated some possible strategies that can be used by the attacker and the defender in a PassGAN based honeywords system. The

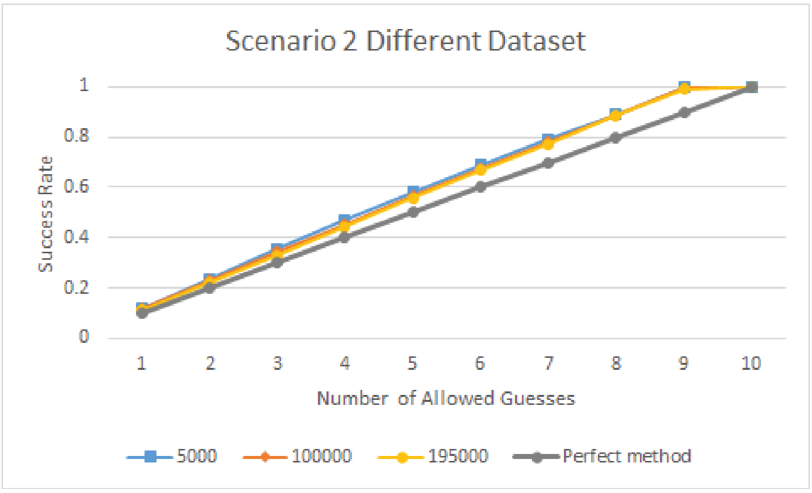


Figure A.5: The attacker and the defender use different datasets.

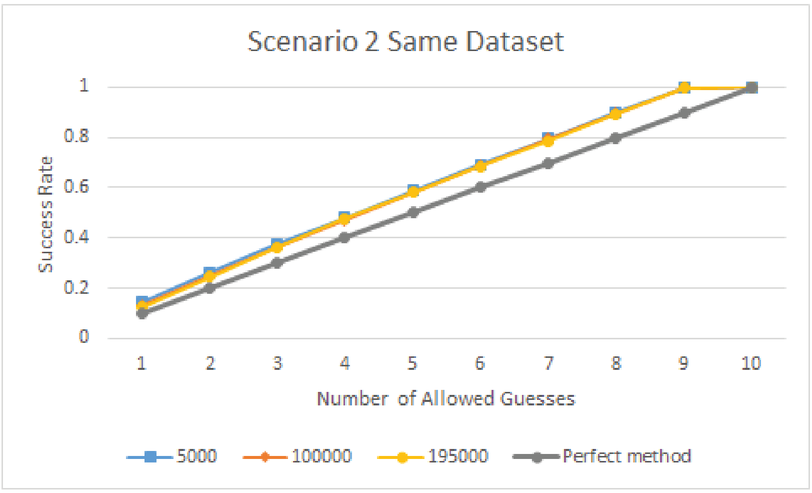


Figure A.6: The attacker and the defender use the same datasets.

defender uses the generator model of PassGAN to generate high-quality fake passwords while the attacker is assumed can manage to steal and crack all the hashed password data and then uses the discriminator model of PassGAN to distinguish the real passwords from the fake ones. Based on the experiment results, PassGAN is feasible for honeywords generation strategy. PassGAN is "approximately flat" even when the attacker also uses PassGAN to distinguish the sugarword from the honeywords.

The best strategy that can be used by the attacker is to use the same dataset as the defender 's dataset. Besides, the attacker can also use a large number of iterations as the strategy. The more number of iterations is proven to be able to increase the attacker 's success rate even though it is very small. From the defender's perspective, the strategy of using a large number of iterations also benefits the defender to reduce the attacker 's success rate.

Bibliography

[1] Cis threshold, http://groups.csail.mit.edu/cis/cis-threshold.html. accessed november 1, 2019. 121

[2] Leaks dropbox, https://hashes.org/leaks.php?id=91. last accessed: June 24, 2019. 175

[3] Leaks linkedin, https://hashes.org/leaks.php?id=68. last accessed: July 25, 2019. 175

[4] Understanding biometrics, http://www.griaulebiometrics.com/en-us/book/understanding-biometrics/types/attacks. accessed april, 2015. 44

[5] 24745, I. Information technology, security techniques, biometric information protection. 2010. 17, 41, 119

[6] ABE, N., YAMADA, S., AND SHINZAKI, T. Irreversible fingerprint template using minutiae relation code with bloom filter. In *Proc. Int. Conf. on Biometrics: Theory, Applications and Systems (BTAS)* (2015). 86, 87

[7] AKSHAYA, K., AND DHANABAL, S. Achieving flatness from non-realistic honeywords. In *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)* (2017), IEEE, pp. 1–3. 172

[8] ALMESHEKAH, M. H., AND SPAFFORD, E. H. Planning and integrating deception into computer security defenses. In *Proceedings of the 2014 workshop on New Security Paradigms Workshop* (2014), ACM, pp. 127–138. 14

[9] ANJOS, A., SHAFEY, L. E., ET AL. Bob: a free signal processing and machine learning toolbox for researchers. In *Proc. ACM Int. Conf. on Multimedia, MM* (2012), pp. 1449–1452. 86, 90

[10] BLACK, P. *Fisher-Yates shuffle.* In Dictionary of Algorithms and Data Structures [online]. Available from: http://www.nist.gov/dads/HTML/fisherYatesShuffle.html, (accessed: September 2015). 31

[11] BLOOM, B. Space/time tradeoffs in hash coding with allowable errors. *Comm. of the ACM 13*, 7 (1970), 422–426. 86

[12] BOJINOV, H., BURSZTEIN, E., BOYEN, X., AND BONEH, D. Kamouflage: Loss-resistant password management. In *European symposium on research in computer security* (2010), Springer, pp. 286–302. 172

[13] BOLLE, R., CONNELL, J., PANKANTI, S., RATHA, N., AND SENIOR, A. Guide to biometrics, springer-verlag. 44

[14] BOSE, R. C., AND RAY-CHAUDHURI, D. K. On a class of error correcting binary group codes. *Information and control 3*, 1 (1960), 68–79. 103

[15] BOWEN, B. M., HERSHKOP, S., KEROMYTIS, A. D., AND STOLFO, S. J. Baiting inside attackers using decoy documents. In *International Conference on Security and Privacy in Communication Systems* (2009), Springer, pp. 51–70. 30

[16] BRANNONDORSEY. Passsgan, https://github.com/brannondorsey/passgan/releases/download/data/rockyou-train.txt. last accessed: May 28, 2019. 175

[17] BREEBAART, J., BUSCH, C., GRAVE, J., AND KINDT, E. A reference architecture for biometric template protection based on pseudo identities. *BIOSIG 2008* (2008), 25–37. 43, 45

[18] BRINGER, J., MOREL, C., AND RATHGEB, C. Security analysis of bloom filter-based iris biometric template protection. In *Proc. Int. Conf. on Biometrics, ICB* (2015), pp. 527–534. 87, 91, 92, 97

[19] BRODER, A., AND MITZENMACHER, M. Network applications of bloom filters: A survey. *Internet Mathematics 1*, 4 (2005), 485–509. 86

[20] CALDER, A. J., BURTON, A. M., MILLER, P., YOUNG, A. W., AND AKAMATSU, S. A principal component analysis of facial expressions. *Vision Research 41* (2001), 1179. 123

[21] CHAKRABORTY, N., AND MONDAL, S. On designing a modified-ui based honeyword generation approach for overcoming the existing limitations. *Computers & Security 66* (2017), 155–168. 172, 173

[22] CHAKRABORTY, N., SINGH, S., AND MONDAL, S. On designing a questionnaire based honeyword generation approach for achieving flatness. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)* (2018), IEEE, pp. 444–455. 172, 173

[23] CHANDRA, E., AND KANAGALAKSHMI, K. Cancelable biometric template generation and protection schemes: A review. In *2011 3rd International Conference on Electronics Computer Technology* (2011), vol. 5, IEEE, pp. 15–20. 48

[24] CHATTERJEE, S., AND DAS, M. P. L. Property preserving symmetric encryption revisited. In *International Conference on the Theory and Application of Cryptology and Information Security* (2015), Springer, pp. 658–682. 37

[25] DATABASE, B. M. Publicly available at:. 165

[26] DAUGMAN, J. How iris recognition works. *IEEE Trans. on Circuits and Systems for Video Technology 14*, 1 (2004), 21–30. 86

[27] DICHTL, M. Cryptographic shuffling of random and pseudorandom sequences. *Symmetric Cryptography* (2007). 31

[28] DRAPER, B. A., BAEK, K., BARTLETT, M. S., AND BEVERIDGE, J. R. Recognizing faces with pca and ica. *Computer vision and image understanding 91*, 1 (2003), 115–137. 107

[29] DRAPER, B. A., BAEK, K., BARTLETT, M. S., AND BEVERIDGEA, J. R. Recognizing faces with pca and ica. *Computer Vision and Image Understanding 91* (2003), 115–137. 123

[30] ERGULER, I. Achieving flatness: Selecting the honeywords from existing user passwords. *IEEE Transactions on Dependable and Secure Computing 13*, 2 (2015), 284–295. 172, 173

[31] FACES94. Collection of facial images: Available at: http://cswww.essex.ac.uk/mv/allfaces/faces94.html, (accessed: October 2015). 109

[32] FLORENCIO, D., AND HERLEY, C. A large-scale study of web password habits. In *Proceedings of the 16th international conference on World Wide Web* (2007), ACM, pp. 657–666. 171

[33] FLORÊNCIO, D., HERLEY, C., AND VAN OORSCHOT, P. C. An administrator's guide to internet password research. In *28th Large Installation System Administration Conference (LISA14)* (2014), pp. 44–61. 171

[34] FURNELL, S. M., DOWLAND, P., ILLINGWORTH, H., AND REYNOLDS, P. L. Authentication and supervision: A survey of user attitudes. *Computers & Security 19*, 6 (2000), 529–539. 171

[35] GADAT, S., AND YOUNES, L. A stochastic algorithm for feature selection in pattern recognition. *The Journal of Machine Learning Research 8* (2007), 509–547. 94

[36] GENC, Z. A., KARDAŞ, S., AND KIRAZ, M. S. Examination of a new defense mechanism: Honeywords. In *IFIP International Conference on Information Security Theory and Practice* (2017), Springer, pp. 130–139. 172

[37] GOMEZ-BARRERO, M., RATHGEB, C., GALBALLY, J., BUSCH, C., AND FIERREZ, J. Unlinkable and irreversible biometric template protection based on bloom filters. *Information Sciences* (2016). Submitted. 87, 92

[38] GOMEZ-BARRERO, M., RATHGEB, C., GALBALLY, J., FIERREZ, J., AND BUSCH, C. Protected facial biometric templates based on local gabor patterns and adaptive bloom filters. In *Proc. Int. Conf. on Pattern Recognition, ICPR* (2014), pp. 4483–4488. 85, 86, 87, 90

[39] GOODFELLOW, I., POUGET-ABADIE, J., MIRZA, M., XU, B., WARDE-FARLEY, D., OZAIR, S., COURVILLE, A., AND BENGIO, Y. Generative adversarial nets. In *Advances in neural information processing systems* (2014), pp. 2672–2680. 172, 173

[40] GULRAJANI, I., AHMED, F., ARJOVSKY, M., DUMOULIN, V., AND COURVILLE, A. C. Improved training of wasserstein gans. In *Advances in neural information processing systems* (2017), pp. 5767–5777. 173

[41] GÜNTHER, M., WALLACE, R., AND MARCEL, S. An open source framework for standardized comparisons of face recognition algorithms. In *Proc. European Conf. on Computer Vision, ECCV* (2012), vol. 7585 of *LNCS*, pp. 547–556. 90

[42] HACKETT, R. Yahoo raises breach estimate to full 3 billion accounts, by far biggest known. 171

[43] HEIM, P. Resetting passwords to keep your files safe. 171

[44] HERMANS, J., MENNINK, B., AND PEETERS, R. When a bloom filter is a doom filter: Security assessment of a novel iris biometric. In *Proc. Int. Conf. of the Biometrics Special Interest Group, BIOSIG* (2014). 87

[45] HITAJ, B., GASTI, P., ATENIESE, G., AND PEREZ-CRUZ, F. Passgan: A deep learning approach for password guessing. In *International Conference on Applied Cryptography and Network Security* (2019), Springer, pp. 217–237. 172, 173

[46] HOMEPAGE, T. S. S. *http://http://srp.stanford.edu/*, 2011. 17

[47] HONEYPOTS. Available at http://www.honeypots.org/, (accessed: September 2015). 29

[48] ILEANA BUHAN, P. H. The state of the art in abuse of biometrics. Technical report, Centre for Telematics and Information Technology, University of Twente, December, 2005. 43

[49] ISO/IEC JTC1 SC27 SECURITY TECHNIQUES. *ISO/IEC 24745:2011. Information Technology - Security Techniques - Biometric Information Protection*. International Organization for Standardization, 2011. 13, 43, 46, 47, 85, 87, 91, 114

[50] ISO/IEC TC JTC1 SC37 BIOMETRICS. *ISO/IEC 19795-1:2006. Information Technology – Biometric Performance Testing and Reporting – Part 1: Principles and Framework*. International Organization for Standardization and International Electrotechnical Committee, Mar. 2006. 13

[51] JAIN, A. K., NANDAKUMAR, K., AND NAGAR, A. Biometric template security. *EURASIP J. Advanced Signal Process 2008* (2008), 1–17. 17, 45, 47, 48

[52] JAIN, A. K., NANDAKUMAR, K., AND NAGAR, A. *Biometric template security*. EURASIP, 2008. 42, 49

[53] JEONG, D. H., ZIEMKIEWICZ, C., RIBARSKY, W., AND CHANG, R. Understanding principal component analysis using a visual analytics tool. *Charlotte Visualization Center* (2009). 55, 107

[54] JEONG, D. H., ZIEMKIEWICZ, C., RIBARSKY, W., CHANG, R., AND CENTER, C. V. *Understanding Principal Component Analysis Using a Visual Analytics Tool*. Charlotte Visualization Center, UNC Charlotte, 2009. 123

[55] JEONG, M., LEE, C., KIM, J., CHOI, J.-Y., TOH, K.-A., AND KIM, J. Changeable biometrics for appearance based face recognition. In *2006 biometrics symposium: special session on research at the biometric consortium conference* (2006), IEEE, pp. 1–5. 149

[56] JUELS, A. A bodyguard of lies: The use of honey objects in information security. In *Proc. ACM Symposium on Access Control Models and Technologies* (2014), pp. 1–4. 15

[57] JUELS, A., AND RIVEST, R. L. *Honeywords: Making Password-Cracking Detectable*. SA Labs, Cambridge. 120

[58] JUELS, A., AND RIVEST, R. L. Honeywords: Making password-cracking detectable. In *Proc. ACM SIGSAC Conference on Computer & Communications Security* (2013), pp. 145–160. 15, 17, 18, 30, 31

[59] JUELS, A., AND RIVEST, R. L. Honeywords: Making password-cracking detectable. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security* (2013), ACM, pp. 145–160. 172, 173, 176

[60] JUELS, A., AND SUDAN, M. A fuzzy vault scheme. *Designs, Codes and Cryptography 38*, 2 (2006), 237–257. 18, 47, 102

[61] JUELS, A., AND WATTENBERG, M. A fuzzy commitment scheme. *ACM Conf. on Computer and Communications Security* (1999), 28–36. 18

[62] JUELS, A., AND WATTENBERG, M. A fuzzy commitment scheme. A. C. on Computer and C. Security, Eds., pages 286. 48, 49

[63] JUELS, A., AND WATTENBERG, M. A fuzzy commitment scheme. *ACM Conf. on Computer and Communications Security* (1999), 28–36. 107

[64] JUELS, A., AND WATTENBERG, M. A fuzzy vault scheme. In *Proc* (2002), IEEE Int. Symposium on Information Theory. 48, 49

[65] K, J., J, M., J, K., AND YP, L. Learning support vectors for face verification and recognition. In *Automatic Face and Gesture Recognition* (2000). 78

[66] KAUR, M., AND SOFAT, S. Template and database security in biometrics systems: A challenging task. *International Journal of Computer Applications 975* (July 2010). 44

[67] KHANDELWAL, S., GUPTA, P. C., AND MANTRI, K. Survey of threats to the biometric authentication systems and solutions. *International Journal of Computer Applications 61.17* (2013), 39–43. 44

[68] KONG, A., CHEUNGA, K.-H., ZHANGA, D., KAMELB, M., AND YOUA, J. An analysis of BioHashing and its variants. *Pattern Recognition 39*, 7 (2006), 1359–1368. 47

[69] LI, G., YANG, B., RATHGEB, C., AND BUSCH, C. Towards generating protected fingerprint templates based on bloom filters. In *Proc. Int. Workshop on Biometrics and Forensics (IWBF)* (2015). 86, 87

[70] MALTONI, D., MAIO, D., JAIN, A., AND PRABHAKAR, S. Handbook of fingerprint recognition. Springer Science and Business Media, 2006. 44

[71] MARTINEZ-DIAZ, M., FIERREZ-AGUILLAR, J., ALONSO-FERNANDEZ, F., ORTEGA-GARCIA, J., AND SIGUENZA, J. A. Hill-climbing and brute-force attacks on biometric systems: A case study in match-on-card fingerprint verification. *in Proc IEEE Intl. Carnahan Conf. on Security Technology, ICCST, Lexington, USA* (2006), 151–159. 44, 59

[72] MARTIRI, E., YANG, B., AND BUSCH, C. Protected honey face templates. *9th IEEE International conference, BIOSIG 2015*. xv, 119, 120, 123, 124, 125, 127

[73] MARTIRI, E., YANG, B., AND BUSCH, C. Protected honey face templates. In *Proc. BIOSIG* (2015). 86, 107, 136

[74] MASSEY, J. L. An introduction to contemporary cryptology. *Proceedings of the IEEE 76*, 5 (1988), 533–549. 33

[75] MATHWORKS. Documentation on bch codes. available from http://se.mathworks.com/help/comm/ref/bchenc.html, (accessed: September 2015). 103, 109

[76] MENEZES, A. J., VAN OORSCHOT, P. C., AND VANSTONE, S. A. *Handbook of applied cryptography*. CRC press, 1996. 33

[77] NANDAKUMAR, K., AND JAIN, A. K. Biometric template protection: Bridging the performance gap between theory and practice. *IEEE Signal Processing Magazine - Special Issue on Biometric Security and Privacy* (2015), 1–12. 45

[78] NANDAKUMAR, K., JAIN, A. K., AND PANKANTI, S. Fingerprint-based Fuzzy Vault: Implementation and Performance. *IEEE Trans. on Information Forensics and Security 2* (2007), 744–757. 47

[79] NANDAKUMAR, K., AND SYSTEMS:, M. Fusion strategies and template security, phd dissertation, michigan state university. 2008. 43, 49

[80] NAZARIO, J. Phoneyc: A virtual client honeypot. *LEET 9* (2009), 911–919. 29

[81] OF FACES, T. D. online at:. 55, 122, 123

[82] ORTEGA-GARCIA, J., FIERREZ, J., ET AL. The multi-scenario multi-environment BioSecure multimodal database (BMDB). *IEEE Trans. on Pattern Analysis and Machine Intelligence 32* (2010), 1097–1111. 86, 90

[83] PALLABI, P. Face recognition using multiple classifiers. In *International Conference on Tools with Artificial Intelligence* (pp. 179-186, 2006.). 78

[84] RAO, S. Data and system security with failwords, July 20 2006. US Patent App. 11/039,577. 172

[85] RATHA, N., CONNELL, J., AND BOLLE, R. Enhancing security and privacy in biometrics-based authentication systems. *IBM Systems Journal 40*, 3 (2001), 614–634. 17, 47

[86] RATHA, N. K., CHIKKERUR, S., CONNELL, J. H., AND BOLLE, R. M. Generating cancelable fingerprint templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence 29(4)* (April 2007), 561–572. 49

[87] RATHA, N. K., CONNELL, J., AND CHIKKERUR, S. Generating cancelable fingerprint templates. *IEEE Trans. on Pattern Analysis and Machine Intelligence 29*, 4 (2007), 561–572. 47, 48

[88] RATHGEB, C., BREITINGER, F., AND BUSCH, C. Alignment-free cancelable iris biometric templates based on adaptive bloom filters. In *Proc. Int. Conf. on Biometrics, ICB* (2013), pp. 1–8. 85, 86

[89] RATHGEB, C., BREITINGER, F., BUSCH, C., AND BAIER, H. On the application of bloom filters to iris biometrics. *IET Biometrics 3*, 1 (2014). 87

[90] RATHGEB, C., AND BUSCH, C. Cancelable multi-biometrics: Mixing iris-codes based on adaptive bloom filters. *Elsevier Computers and Security 42*, 0 (2014). 87

[91] RATHGEB, C., GOMEZ-BARRERO, M., BUSCH, C., GALBALLY, J., AND FIERREZ, J. Towards cancelable multi-biometrics based on bloom filters: A case study on feature level fusion of face and iris. In *Proc. Int. Workshop on Biometrics and Forensics, IWBF* (2015), pp. 1–7. 86, 87, 90

[92] RATHGEB, C., AND UHL, A. A survey on biometric cryptosystems and cancelable biometrics. *EURASIP Journal on Information Security 2011*, 3 (2011). 17, 47, 102

[93] RATHGEB, C., AND UHL, A. *A survey on biometric cryptosystems and cancelable biometrics*. EURASIP Journal on Information Security, 2011. 48, 49

[94] SAL STOLFO, STEVEN M. BELLOVIN, D. E. Measuring security. *IEEE Security & Privacy 3* (2011), 60–65. 15

[95] SHPILRAIN, V. Decoy-based information security. *Groups Complexity Cryptology 6.2* (2014), 149–155. 14, 53

[96] SIEGLER, M. One of the 32 million with a rockyou account? you may want to change all your passwords. like now. 171

[97] SIMOENS, K., YANG, B., ZHOU, X., BEATO, F., BUSCH, C., NEWTON, E., AND PRENEEL, B. Criteria toward metrics for benchmarking template protection algorithms. In *2012 5th IAPR International Conference on Biometrics* (2012), IEEE. 41, 119, 122

[98] SIMOENS, K., YANG, B., ZHOU, X., BEATO, F., BUSCH, C., NEWTON, E., AND PRENEEL, B. Criteria towards metrics for benchmarking template protection algorithms. In *Proc. Int. Conf. on Biometrics, ICB* (2012), pp. 498–505. 17

[99] SIMOENS, K., YANG, B., ZHOU, X., BEATO, F., BUSCH, C., NEWTON, E., AND PRENEEL, B. Criteria towards metrics for benchmarking template protection algorithms. In *Proc. Int. Conf. on Biometrics, ICB* (2012), pp. 498–505. 47, 55

[100] SPITZNER, L. *Honeypots: Tracking Hackers*. Addison Wesley, 2002. 15, 29

[101] SUTCU, Y., LI, Q., AND MEMON, N. Secure biometric templates from fingerprint-face features,. *Conference on Computer Vision and Pattern Recognition, CVPR '07. IEEE 2007*. 48, 49

[102] SUTCU, Y., LI, Q., AND MEMON, N. *Protecting biometric templates with sketch: Theory and practice*. IEEE Transactions on Information Forensics and Security, 2007. 18, 48

[103] TEOH, A. B. J., GOH, A., AND NGO, D. C. L. Random multispace quantisation as an analytic mechanism for biohashing of biometric and random identity inputs. *IEEE Trans. on Pattern Analysis and Machine Intelligence 28*, 12 (2006), 1892–1901. 17

[104] TEOH, A. B. J., GOH, A., AND NGO, D. C. L. Random multispace quantisation as an analytic mechanism for biohashing of biometric and random identity inputs. *IEEE Trans. on Pattern Analysis and Machine Intelligence 28*, 12 (2006), 1892–1901. 47, 49

[105] TREVOR, H., TIBSHIRANI, R., FRIEDMAN, J., AND FRANKLIN, J. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer 27, no. 2 (2005): 83-85.*. 137

[106] ULUDAG, U., AND JAIN, A. K. Attacks on biometric systems: a case study in fingerprints. *Proc. SPIE-EI Security, Seganography and Watermarking of Multimedia Contents VI, San Jose, CA, pp. 622?33 2004* (2004). 44

[107] ULUDAG, U., PANKANTI, S., PRABHAKAR, S., AND JAIN, A. K. Biometric cryptosystems: issues and challenges. *Proc. of the IEEE 92*, 6 (2004), 948–960. 47

[108] WANG, D., CHENG, H., WANG, P., YAN, J., AND HUANG, X. A security analysis of honeywords. In *NDSS* (2018). 171

[109] WU, Y., AND QIU, B. Transforming a pattern identifier into biometric key generators. In *Proc. Int. Conf. on Multimedia and Expo, ICME* (2010), pp. 78–82. 47

[110] Y, X., ZHU, Z, F., AND ET AL., Q. M. Coarse to fine k nearest neighbor classifier. *Pattern recognition letters.* (2013 Jul 1;34(9):980-6.). 78

[111] YANG, B., HARTUNG, D., SIMOENS, K., AND BUSCH, C. Dynamic random projection for biometric template protection. In *In Proceedings of the 4th IEEE International Conference on Biometrics: Theory Applications and Systems (BTAS)* (2010). 55, 149

[112] YANG, B., HARTUNG, D., SIMOENS, K., AND BUSCH, C. Dynamic random projection for biometric template protection. In *Biometrics: Theory Applications and Systems (BTAS), 2010 Fourth IEEE International Conference on* (2010), IEEE, pp. 1–7. 73, 107

[113] YANG, B., AND MARTIRI, E. Using honey templates to augment hash based biometric template protection. In *Proc. Int. Workshop on Secure Identity Management in the Cloud Environment (SIMICE)* (2015). xiv, 85, 88, 135, 136, 155

[114] YANG, B., AND MARTIRI, E. Using honey templates to augment hash based biometric template protection. In *proceedings of the 39th IEEE International Computers* (2015), Software and Applications Conference. 30, 119, 120

[115] YANG, B. H., SIMOENS, D., BUSCH, K., AND DYNAMIC, C. random projection for biometric template protection. In *Proceedings of the 4th IEEE International Conference on Biometrics: Theory Applications and Systems (BTAS* (2010). 122, 125

[116] YUILL, J., ZAPPE, M., DENNING, D., AND FEER, F. Honeyfiles: deceptive files for intrusion detection. In *Proc. SMC Information Assurance Workshop (IAW)* (2004), pp. 116–122. 15

[117] ZHANG, W., SHAN, S., ET AL. Local gabor binary pattern histogram sequence (LGBPHS): a novel non-statistical model for face representation and recognition. In *Proc. Int. Conf. on Computer Vision, ICCV* (2005), vol. 1, pp. 786–791. 86, 90

ISBN 978-82-326-6107-7 (printed ver.)
ISBN 978-82-326-6970-7 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)