TÉCNICO LISBOA

UNIVERSIDADE DE LISBOA
INSTITUTO SUPERIOR TÉCNICO

Privacy-Preserving Frameworks for Speech Mining

José Miguel Ladeira Portêlo

Supervisor:	Doctor Isabel Maria Martins Trancoso
Co-Supervisor:	Doctor Bhiksha Raj

Thesis approved in public session to obtain the PhD Degree in
Electrical and Computer Engineering

Jury final classification: Pass with Distinction

Jury

Chairperson:	Chairman of the IST Scientific Board
Members of the Committee:	Doctor Atta Badii
	Doctor Isabel Maria Martins Trancoso
	Doctor Bhiksha Raj
	Doctor Carlos Nuno da Cruz Ribeiro
	Doctor José Miguel de Oliveira Monteiro Sales Dias
	Doctor Alysson Neves Bessani
	Doctor Alberto Abad Gareta

2015

TÉCNICO LISBOA

UNIVERSIDADE DE LISBOA
INSTITUTO SUPERIOR TÉCNICO

Privacy-Preserving Frameworks for Speech Mining

José Miguel Ladeira Portêlo

Supervisor:	Doctor Isabel Maria Martins Trancoso
Co-Supervisor:	Doctor Bhiksha Raj

Thesis approved in public session to obtain the PhD Degree in
Electrical and Computer Engineering

Jury final classification: Pass with Distinction

Jury

Chairperson:	Chairman of the IST Scientific Board
Members of the Committee:	Doctor Atta Badii, Distinguished Professor, School of Systems Engineering, University of Reading, UK
	Doctor Isabel Maria Martins Trancoso, Professora Catedrática do Instituto Superior Técnico da Universidade de Lisboa
	Doctor Bhiksha Raj, Associate Professor, Language Technologies Institute, Carnegie Mellon University, USA
	Doctor Carlos Nuno da Cruz Ribeiro, Professor Associado do Instituto Superior Técnico da Universidade de Lisboa
	Doctor José Miguel de Oliveira Monteiro Sales Dias, Professor Associado Convidado da Escola de Tecnologias e Arquitectura do ISCTE - Instituto Universitário de Lisboa
	Doctor Alysson Neves Bessani, Professor Auxiliar da Faculdade de Ciências da Universidade de Lisboa
	Doctor Alberto Abad Gareta, Professor Auxiliar do Instituto Superior Técnico da Universidade de Lisboa

Funding Institutions
Fundação para a Ciência e a Tecnologia

2015

Resumo

As agências de segurança desejam muitas vezes monitorizar conversas telefónicas para determinar se alguma delas é importante para a segurança nacional, mas isto resulta numa invasão intolerável da privacidade de milhares de cidadãos comuns, o que é de facto considerado ilegal em muitas situações. A fala é considerada uma das formas de comunicação mais privadas, e as pessoas não gostam de ser escutadas ou gravadas sem permissão. A voz de uma pessoa contém informação sobre o que ela disse, o seu sexo, nacionalidade (pronúncia) e até mesmo estado emocional, tudo coisas que ela pode não querer revelar. Presumivelmente, as agências estão principalmente interessadas em detectar vozes específicas ou identificar certas frases-chave. No entanto, mesmo que elas o façam usando sistemas automáticos, as tecnologias actuais ainda precisam de acesso total às gravações originais. A única forma de não lhes conceder acesso à voz gravada é não utilizá-la de todo.

Esta tese propõe soluções de preservação de privacidade para tarefas de extração de informação latente em fala. O processamento é realizado sem ter acesso à voz. Aqui, "acesso à voz" refere-se a ter acesso a qualquer forma de fala que possa ser analisada para obter informações acerca dos oradores ou do que eles disseram. Usando uma combinação de ferramentas criptográficas e de computação multi-entidade segura, os algoritmos de processamento de voz são tornados seguros, de forma a que a privacidade de todas as partes envolvidas (oradores, sistema e qualquer outra entidade envolvida no processamento) seja preservada.

Em particular, esta tese propõe especificamente soluções de preservação de privacidade para tarefas de procura por exemplos em fala e verificação de orador. A solução para o primeiro problema usa uma técnica de *hashing* de preservação de distâncias para ocultar a informação das características, permitindo no entanto que as gravações que contêm a palavra-chave de interesse possam ser recuperadas usando um algoritmo de programação dinâmica. Para o

último problema são apresentadas duas soluções: uma que usa a mesma técnica de *hashing* de preservação de distâncias nas características e que em seguida usa um classificador de aprendizagem supervisionada sobre as *hashes* resultantes, a outra que usa uma técnica de computação multi-entidade segura para avaliar modelos probabilísticos. Ambas as soluções para o último problema procuram determinar se um orador é de facto quem ele/ela afirma ser.

Estas soluções de preservação de privacidade permitem atingir resultados semelhantes aos obtidos pelas suas soluções homólogas não seguras, permitindo simultaneamente garantir que nenhuma outra informação pode ser obtida a partir da voz. Apesar de não serem soluções perfeitas, uma vez que a eficácia depende da precisão do sistema de procura em fala ou de verificação de orador que foi protegido, estas soluções são ainda assim uma grande melhoria em relação à situação actual.

Embora esta tese aborde apenas técnicas de preservação de privacidade para processamento de fala, o impacto destas técnicas não é limitado a esta área. De facto, estas podem influenciar todos os tipos de documentos multimédia. Na Internet dos dias de hoje, é mais fácil do que nunca criar, consumir, partilhar e arquivar conteúdos multimédia. Este nível sem precedentes de acesso gera preocupações de segurança e privacidade, tais como proteger a criatividade via direitos de autor, prevenir corrupção e roubo de dados privados e garantir partilha legal de dados. Uma forma eficaz de resolver estas preocupações é realizar cálculos e processamento directamente nos sinais que tenham sido ofuscados quer com criptografia, partilha de segredos ou primitivas de preservação de privacidade. Em resposta a este ênfase em segurança e privacidade, tem havido um aumento rápido no interesse em aspectos fundamentais e de aplicação de processamento de sinais de forma segura, alimentado por investigadores nas áreas de criptografia, processamento de sinal e extração de informação latente em dados.

Abstract

Security agencies often desire to monitor telephonic conversations to determine if any of them are important to national security, but this results in an intolerable invasion of the privacy of thousands of regular citizens, which is in fact considered illegal in many situations. Speech is regarded as one of the most private forms of communication, and people do not like being eavesdropped on or recorded without permission. A person's voice contains information about what they spoke, their gender, nationality (accent) and even emotional state, all of which they may not want to reveal. Presumably, the agencies are mainly interested in detecting certain voices or identifying certain key phrases. However, even if they did so using automated systems, current technologies still require full access to the subjects' recordings. Not granting them access to one's voice is only possible if one does not use it at all.

This thesis proposes privacy-preserving frameworks for speech mining. Processing is performed without having access to the voice. Here, "access to voice" refers to having access to any form of the speech that can be analysed to obtain information about the talkers or what they spoke. Using a combination of cryptographic and secure multi-party computation tools, voice processing algorithms are rendered secure, so that the privacy of all parties (speakers, the system and any other entity involved in the processing) is preserved.

In particular, this thesis specifically proposes privacy-preserving solutions for the query-by-example speech search and speaker verification tasks. The solution for the first problem uses a distance-preserving hashing technique to hide the information from the features while still allowing for recordings containing the keyword of interest to be retrieved using a dynamic programming algorithm. For the latter problem, two solutions are presented: one using the same distance-preserving hashing technique on the features and then using a supervised learning classifier on the resulting hashes, another using a secure multi-party computation

technique to evaluate probabilistic models. Both solutions to the latter problem aim to determine if a speaker is indeed who he/she claims to be.

These privacy-preserving solutions enable obtaining results similar to the ones provided by their non-secure counterparts, while simultaneously ensuring that no further information may be obtained from the voice. Despite not being perfect solutions, since the efficacy depends on the accuracy of the actual speech search or speaker verification system that has been secured, these solutions are still a vast improvement over the current situation.

Although this thesis only addresses privacy-preserving techniques for speech processing, the impact of these techniques is not restricted to this area. In fact, they may affect all types of multimedia documents. In today's Internet era, it is easier than ever before to create, consume, share and archive multimedia content. This unprecedented level of access creates security and privacy concerns, such as protecting creativity via copyright, preventing corruption and theft of private data, and ensuring lawful sharing of data. A powerful way to address these concerns is to perform computation and processing directly on signals that have been obfuscated either by encryption, secret sharing or other privacy-preserving primitives. In response to this emphasis on security and privacy, there has been a surge of interest in fundamental and applied aspects of secure signal processing, fuelled by researchers from the cryptography, signal processing, and data mining communities.

Palavras-Chave
Keywords

Palavras-Chave

Privacidade de Dados
Criptografia
Correspondência de Músicas
Procura por Exemplos em Fala
Verificação de Orador

Keywords

Data Privacy
Cryptography
Music Matching
Query-by-Example Speech Search
Speaker Verification

Acknowledgements

This work has been carried out at the Spoken Language Systems Laboratory (L^2F) at INESC-ID. Several individuals and institutions contributed directly and indirectly to this thesis. I would like thank them all for their support.

First and foremost, I thank my scientific advisers Prof. Isabel Trancoso and Prof. Bhiksha Raj for their technical knowledge, vision, and suggestions regarding research directions and the fulfilment of this thesis. Additionally, I am grateful to my advisers for the freedom to make my own decisions concerning my work and for proof-reading the scientific documents produced throughout these years.

I would like to thank the support of the Portuguese research funding agency Fundação para a Ciência e a Tecnologia (FCT) through the PhD scholarship SFRH/BD/71349/2010 during the first four years of this work, as well as the FCT and Instituto Superior Técnico funding during the last year.

Next, I express my gratitude to the members of the *Comissão de Acompanhamento da Tese*, Prof. Isabel Trancoso, Prof. Bhiksha Raj, Prof. Carlos Ribeiro and Prof. Alberto Abad for their suggestions and recommendations, both in terms of scientific research and organization of this document.

I am deeply grateful to all my L^2F present and former colleagues for their help in multiple fields, specially:

- Alberto Abad, for valuable insight into the Speaker Verification (SV) and Query-by-Example Speech Search (QESS) tasks, as well as providing me with the feature extraction programs used in the baseline experiments on these tasks;

- David Matos, for the technical support regarding the usage of the HTCondor framework for distributed parallel computations and for helping me improve my programming skills;
- Hugo Meinedo and Ramón Astudillo, for helping me using Audimus, the hybrid speech recognizer developed at L^2F;
- João Miranda, Luís Marujo, Tiago Luís and Wang Ling, my office co-workers at L^2F, for helping me improve my programming skills;
- Luís Marujo, for valuable insight into the document summarization task, as well as obtaining the results for the baseline experiments on this task;

I would also like to thank Bhiksha Raj, Rita Singh and all the remaining members of the MLSP group at Carnegie Mellon University for the meaningful discussions regarding the work done in the scope of this thesis.

Finally, a very special thank goes to my family, for everything.

Contents

List of Figures

List of Tables

List of Acronyms

AKWS Acoustic Keyword Spotting

ASCII American Standard Code for Information Interchange

ASR Automatic Speech Recognition

BN Broadcast News

bpc bits per coefficient

CA Complex Addition

CCA1 Chosen-Ciphertext Attacks

CCA2 Adaptive Chosen-Ciphertext Attacks

CM Complex Multiplication

CPA Chosen-Plaintext Attacks

CPS Cross Power Spectrum

CPU Central Processing Unit

DCRA Decisional Composite Residuosity Assumption

DE decryption algorithm

DET Detection Error Tradeoff

DTW Dynamic Time Warping

EER Equal Error Rate

EM Expectation-Maximization

EN encryption algorithm

en-US American English

es-ES European Spanish

ETSI European Telecommunications Standards Institute

FCT Fundação para a Ciência e a Tecnologia

FFT Fast Fourier Transform

FHE Fully Homomorphic Encryption

fps frames per second

GC Garbled Circuit

gcd greatest common divisor

GMM Gaussian Mixture Model

GSV GMM Supervector

HE Homomorphic Encryption

HMM Hidden Markov Model

IFFT Inverse Fast Fourier Transform

IPR Important Passage Retrieval

ITT International Telephone & Telegraph

JFA Joint Factor Analysis

KG key generation algorithm

KWS Keyword Spotting

L^{2}F Spoken Language Systems Laboratory

lcm least common multiple

LDA Linear Discriminant Analysis

LSA Latent Semantic Analysis

LSB Least Significant Bits

LSH Locality-Sensitive Hashing

LVCSR Large Vocabulary Continuous Speech Recognition

MAP Maximum A Posteriori

MAUI Multi-purpose Automatic Topic Indexing

ME Modular Exponentiation

MFCC Mel-Frequency Cepstral Coefficients

MLP Multi-Layer Perceptron

MM Modular Multiplication

MMR Maximal Marginal Relevance

MSB Most Significant Bits

MSG Modulation SpectroGram

MTWV Maximum Term Weighted Value

NIST National Institute of Standards and Technology

OT Oblivious Transfer

PLA Piecewise Linear Approximation

PLDA Probabilistic Linear Discriminant Analysis

PLP Perceptive Linear Predictive

PPEDS Privacy-Preserving Extractive Document Summarization

PPMM Privacy-Preserving Music Matching

PPQESS Privacy-Preserving Query-by-Example Speech Search

PPSV Privacy-Preserving Speaker Verification

PRNG Pseudo-Random Number Generator

pt-BR Brazilian Portuguese

pt-PT European Portuguese

QESS Query-by-Example Speech Search

RASTA RelAtive SpecTrAl

RBF Radial Basis Function

RNG Random Number Generator

ROUGE Recall-Oriented Understudy for Gisting Evaluation

RQ Ramp Quantization

RSA Rivest-Shamir-Adler

SAMPA Speech Assessment Methods Phonetic Alphabet

SBE Secure Binary Embedding

SIP Secure Inner Product

SLOG Secure Logsum

SMAX Secure Maximum Index

SMH Secure Modular Hashing

SMPC Secure Multi-Party Computation

SQ Step Quantization

SRE Speaker Recognition Evaluation

STD Spoken Term Detection

STPC Secure Two-Party Computation

SUSPECT SecUre SPEeCh Technologies

SV Speaker Verification

SVAL Secure Maximum Value

SVM Support Vector Machine

SWS Spoken Web Search

TV Total Variability

UBM Universal Background Model

UQ Universal Quantization

VIPP Visual Information Processing and Protection

WAVE Waveform Audio File Format

WCCN Within-Class Covariance Normalization

1 Introduction

1.1 Privacy and Security

Traditionally, privacy concerns have been related to protection of data from unauthorized entities during communication and storage. The most popular solutions for protection of privacy are symmetric- and asymmetric-key encryption algorithms [118, 36] that use a published "public" key to encrypt the data. Only the intended recipient who has the "private" key can decrypt the encrypted data. These encryption techniques are virtually impregnable, requiring infeasibly long times to break even with powerful computers.

A different situation arises when two or more parties aim to engage in collaborative computation, but do not wish to expose their data to one another. An illustrative example is the so-called millionaire problem [138]: two millionaires desire to find out which of them is wealthier without revealing their own worth. Real-world examples are also readily found, *e.g.* distributed voting (where consensus must be determined without revealing individual votes), private bidding auctions, private information retrieval (where one does not desire to reveal one's query), privacy-maintaining collaborative filtering, etc.

Such problems can theoretically be solved through Fully Homomorphic Encryption (FHE) schemes that enable the computation of arbitrary functions directly on encrypted data [117]. However, despite recent developments [47], computationally-feasible FHE remains elusive. An alternative is to employ Secure Multi-Party Computation (SMPC) protocols. The general mechanism involves iterated exchange of encrypted or hidden partial information between parties according to a pre-specified protocol until one or more parties have the result. The protocols incorporate a variety of data-hiding methods such as Homomorphic Encryption (HE), data randomization, Oblivious Transfer (OT), etc. A protocol is deemed secure if no

participant can learn more from the computation than what can be inferred from the final result. SMPC protocols have been used for the privacy-preserving variants of several learning and classification tasks, such as k-means clustering [133], learning decision trees [129], Support Vector Machine (SVM) [141], computation of Hamming distances between bit strings [122], principal components analysis [29], singular value decomposition [54], etc. There is a large amount of literature on the subject; too large to describe here in any detail.

1.2 Speech and Audio Mining

Over the past few decades there has been an exponential increase in terms of production of audio data. Recurrent problems include not only how to store and access this data efficiently, but also being able to process it in order to extract relevant information. The raw data that is available can be noisy or otherwise distorted, meaning that some pre-processing techniques (*e.g.*: filtering, distortion compensation, etc.) may need to be applied. Given the huge diversity of audio sources and applications for the audio data, the type of pre-processing required may vary substantially from source to source.

Although there are many ways to separate audio into different categories according to its source, for the scope of this thesis it is convenient to consider just two major categories, speech and non-speech, and to divide non-speech into two sub-categories, music and audio events. An audio event is a much more recent concept than both speech and music, and refers to all audio that is non-speech and non-music, such as the one produced by animals, rivers, wind, traffic, power tools, etc. Speech mining tasks include, but are not limited to, speaker and language recognition, Keyword Spotting (KWS) and speech recognition, which make it possible to recognize the identity and language/accent of the person speaking and what he/she is saying. Additional information can be extracted from the speech data used by these techniques, such as the emotional state of the speaker, as well as a variety of other paralinguistic information and speaker states and traits[1]. A plethora of text mining techniques can be applied to the transcribed data. Music mining tasks include retrieval and plagiarism detection, which can

[1] http://compare.openaudio.eu/

be performed by analysing the melodic, harmonic and/or rhythmic structure of the music signal. Audio event detection is typically performed by designing specialized classifiers for each individual audio event.

Like in most signal processing areas, the first step for performing speech mining is to parameterize the speech signal by extracting features from it. Usually Mel-Frequency Cepstral Coefficients (MFCC) or Perceptive Linear Predictive (PLP) features are extracted, although in many scenarios features such as pitch, zero-crossing rate, phoneme duration might prove useful. In fact, the number of features extracted may be very high [40]. A variety of machine learning techniques such as deep learning, Gaussian Mixture Model (GMM), Hidden Markov Model (HMM) and SVM may be used for solving all of the speech and audio mining tasks mentioned above. At the risk of sounding repetitive, there is too much literature on the topic to describe here in any detail.

1.3 Previous Work on Privacy-Preserving Speech and Audio Mining

Security agencies often desire to monitor telephonic conversations to determine if any of them are of importance to national security, but this results in an intolerable invasion of the privacy of hundreds of regular citizens and, in fact, is considered illegal in many situations. Speech is regarded as one of the most private forms of communication and people do not like to be eavesdropped on, or recorded without permission. A person's voice contains information not just about what they said but also about their gender, nationality (accent) and even their emotional state, all of which they may not want to reveal. Presumably, the agency is mainly interested in detecting certain voices, or identifying certain key phrases. However, even if they did so using automated systems, current technologies would still provide them full access to the subjects' recordings. Not granting them access to one's voice is only possible if one does not use them at all.

Following the privacy-preserving solution to the millionaires' problem [138], many techniques

were developed for privacy-preserving data mining tasks, some of which were already mentioned in Section 1.1. Regarding speech and audio mining in particular, previous research on privacy-preserving approaches is very limited and therefore it is still a mostly untouched area of scientific research. To the best of our knowledge, previous work is limited to addressing simple tasks, such as musical database matching [126], building blocks required for more complex speech processing tasks, such as Linear Discriminant Analysis (LDA) [53], GMM [125] and HMM [128], or preliminary attempts to address the Speaker Verification (SV) task [98, 99].

We now present a small summary for each to these techniques. For the Privacy-Preserving Music Matching (PPMM) protocol, Alice has a music recording but does not know any of the metadata regarding it (*e.g.* song title, artist, album name), while Bob has an extensive database of songs together with their corresponding metadata. In order for Alice to gain the information she desires without revealing her music recording to Bob, and Bob being assured that Alice only accesses those specific pieces of information, they engage in several Secure Two-Party Computation (STPC) protocols [126]. The privacy-preserving computation of LDA from Alice's matrix A and Bob's matrix B is made using two separate protocols, secure matrix multiplication and secure inverse of matrix sum, which keeps their respective data hidden from each other, while the final result is shared between them using additive shares [53]. Regarding the privacy-preserving algorithms for learning and classification using GMM and HMM, if Alice has a set of feature vectors X and Bob has a GMM or HMM with parameters θ, the algorithm provides Alice and Bob with additive shares a and b, respectively, such that $a+b = \text{GMM}\,(X, \theta)$ or $a+b = \text{HMM}\,(X, \theta)$, the value of the GMM or HMM evaluated at X [125, 128]. Finally, the first approach to Privacy-Preserving Speaker Verification (PPSV) consists of using STPC to evaluate input vectors representing Alice's voice against GMM models owned by Bob (one generic and the other specific to Alice), while the second approach relies on a hashing scheme that hides the original information while allowing distance information to be preserved. This way, Alice can send Bob hashes computed from features representing her voice, which he then compares to hashes supplied by Alice during a registration stage [98, 99].

1.4 Main Contributions

The main contributions of this thesis are the design of novel privacy-preserving protocols for several audio and speech processing tasks, namely music matching, Query-by-Example Speech Search (QESS) and SV. In fact, despite extensive use of privacy-preserving techniques in different classification and matching tasks using image-based biometrics, such as fingerprint matching, iris recognition and face identification, there are very few examples of applying such techniques to audio or speech signals.

The work on this thesis was mainly developed in the scope of the SecUre SPEeCh Technologies (SUSPECT) project[2], sponsored by the Portuguese research funding agency Fundação para a Ciência e a Tecnologia (FCT) through grant PTDC/EIA-CCO/122542/2010. The goal of SUSPECT was to develop privacy-preserving frameworks for processing voice data. Processing is to be performed without having access to the voice, *i.e.*, access to any form of speech that can be analysed to obtain information about the person talking or what they said.

To this end, we combined different cryptographic and hashing algorithms with existing techniques for music matching, QESS and SV in order to obtain privacy-preserving versions of those techniques that possess the following properties:

- Effectively hide sensitive information.
- Achieve near negligible degradation in classification results.
- Have small overheads in terms of computation and execution time.

We first addressed the issue of performing music matching by computing operations in the encrypted domain using a partially homomorphic cryptosystem. Although our analysis was conducted on a toy corpus, it provided extremely useful insight into many of the problems that arise when data privacy is to be maintained while performing useful computations on that

[2] https://www.l2f.inesc-id.pt/wiki/index.php/SUSPECT_(SecUre_SPEeCh_Technologies)

same data. We also detected and corrected a privacy flaw in the original protocol formulation, highlighting the inherent difficulty of designed privacy-preserving protocols using STPC. The task of QESS follows on naturally from music matching, as it is also a form of pattern matching. In order to achieve data privacy in this task, we used a hashing method that preserves the notion of distance between the original feature vectors under certain conditions. The last task we addressed was SV. We considered two approaches for obtaining privacy: the first one used the same hashing scheme adopted for the QESS task and the second one was based on Garbled Circuit (GC). Each of the privacy-preserving approaches we considered required a different SV technique. In particular, the one considering GC led to the development of a novel technique for computing the logsum operation in a privacy-preserving manner.

1.5 Publication List

2015:

[112] **J. Portêlo**, B. Raj, and I. Trancoso. Logsum using Garbled Circuits. In *PLoS ONE*, Public Library of Science, March 26 2015.

[111] **J. Portêlo**, A. Abad, B. Raj, and I. Trancoso. Privacy-Preserving Query-by-Example Speech Search. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 1797–1801, Brisbane, Australia, April 19–24 2015.

[131] I. Trancoso, **J. Portêlo**, B. Raj, G. Chollet, N. Cannings, D. Petrovska-Delacrétaz, A. Badii, and J.-J. Quisquater. Privacy Preserving Speech Processing. *Poster presented at the CHIST-ERA Conference*, Lisbon, Portugal, June 16–18 2015.

[77] L. Marujo, **J. Portêlo**, W. Ling, D. Matos, J. Neto, A. Gershman, J. Carbonell, I. Trancoso, and B. Raj. Privacy-Preserving Multi-Document Summarization. In *ACM SIGIR Workshop 'Privacy-Preserving Information Retrieval'*, pages 1–4, Santiago, Chile, August 13 2015.

[61] A. Jiménez, B. Raj, **J. Portêlo**, I. Trancoso. Secure Modular Hashing. In *IEEE International Workshop on Information Forensics and Security (WIFS)*, Rome, Italy, November 16–19 2015.

2014:

[109] **J. Portêlo**, B. Raj, A. Abad, and I. Trancoso. Privacy-Preserving Speaker Verification using Secure Binary Embeddings. In *37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1268–1272, Opatija, Croatia, May 26–30 2014.

[76] L. Marujo, **J. Portêlo**, D. Matos, J. Neto, A. Gershman, J. Carbonell, I. Trancoso, and B. Raj. Privacy-Preserving Important Passage Retrieval. In *ACM SIGIR Workshop 'Privacy-Preserving Information Retrieval'*, pages 7–12, Gold Coast, Australia, July 11 2014.

[110] **J. Portêlo**, B. Raj, A. Abad, and I. Trancoso. Privacy-Preserving Speaker Verification using Garbled GMMs. In *22nd European Signal Processing Conference (EUSIPCO)*, pages 2070–2074, Lisbon, Portugal, September 1–5 2014.

2013:

[107] **J. Portêlo**, A. Abad, B. Raj, and I. Trancoso. Secure Binary Embeddings of Front-End Factor Analysis for Privacy Preserving Speaker Verification. In *14th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pages 2494–2498, Lyon, France, August 25–29 2013.

[108] **J. Portêlo**, B. Raj, P. Boufounos, I. Trancoso, and A. Abad. Speaker Verification using Secure Binary Embeddings. In *21st European Signal Processing Conference (EUSIPCO)*, Marrakech, Morocco, September 9–13 2013.

2012:

[106] **J. Portêlo**, B. Raj, and I. Trancoso. Attacking a Privacy-Preserving Music Matching Algorithm. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 1821–1824, Kyoto, Japan, March 25–30 2012.

[101] M. Pathak, **J. Portêlo**, B. Raj, and I. Trancoso. Privacy-Preserving Speaker Authentication. In *Information Security Conference (ISC)*, pages 1–22, Passau, Germany, September 19–21 2012.

2011:

[105] **J. Portêlo**, B. Raj, A. Abad, and I. Trancoso. On the Implementation of a Secure Musical Database Matching. In *19th European Signal Processing Conference (EUSIPCO)*, pages 1949–1953, Barcelona, Spain, August 29 – September 2 2011.

1.6 Structure of this Document

The remainder of this thesis is structured as follows. In Chapter 2 we describe the cryptographic and hashing techniques used in our privacy-preserving protocols. Chapter 3 presents our work in the scope of the PPMM task. In Chapter 4 we illustrate our approach to the problem of Privacy-Preserving Query-by-Example Speech Search (PPQESS). Chapters 5 and 6 contain two different ways of addressing the PPSV task, resorting to a hashing and cryptographic technique, respectively. This thesis ends with Chapter 7, where we present the conclusions of this work and present some directions for future work.

2 Cryptographic and Hashing Techniques

This chapter describes the cryptographic and hashing techniques considered throughout this thesis. Section 2.1 contains techniques used to perform computations directly on the encrypted domain, while Section 2.2 presents techniques that allow for distance-based classification using randomly generated hashes.

2.1 Secure Multi-Party Computation

Secure Multi-Party Computation (SMPC) enables a set of participants $p_1, p_2, \ldots, p_N$, possessing private data $d_1, d_2, \ldots, d_N$, to collaboratively compute a function $f(d_1, d_2, \ldots, d_N)$ such that no participant can infer anything about the data from any other participant. An interesting sub-problem of SMPC is the situation where $N = 2$, also known as Secure Two-Party Computation (STPC), which was introduced by Yao as the millionaire problem [138]. In short, the problem consists of two millionaires who are interested in knowing which of them is wealthier, but without revealing their actual wealth. Due to the similarities between SMPC and STPC, we will restrict ourselves to the latter, as it is easier to analyse.

2.1.1 Basic Primitives

The parties involved in the STPC scenario are usually referred to as Alice and Bob. The situation mentioned above simplifies to Alice and Bob wanting to compute $c = f(a, b)$ as if it was computed confidentially by a trusted third party, such that neither party learns anything about the other's data more than what can be inferred from the result. This is achieved through iterated exchange of encrypted or otherwise obfuscated partial results in elaborate protocols that employ Homomorphic Encryption (HE) [43], Oblivious Transfer (OT) [85] and

randomization [137]. Typically, every step of the protocol is expressed in terms of a few basic operations, also known as primitives, for which privacy preserving implementations already exist. Furthermore, the intermediate results must be distributed as random additive shares c_a and c_b between the parties, such that the actual outcome of the computation is given by $c = c_a + c_b$. Examples of typical primitives are:

- *Secure Inner Product (SIP)*: If Alice has a vector x and Bob has vector y, the SIP protocol produces two seemingly random numbers a and b such that $a + b = x^\top y$. Alice receives the share a and Bob receives b. These shares can either be propagated to subsequent steps of the algorithm directly or aggregated by a mutually acceptable party to obtain the final result. A variety of SIP algorithms have been proposed in the literature [58, 49].

- *Secure Maximum Index (SMAX)*: Alice has a vector $x = [x_1, x_2, \ldots, x_N]$ and Bob has $y = [y_1, y_2, \ldots, y_N]$. They wish to compute the index of the maximum of $x + y = [x_1 + y_1, x_2 + y_2, \ldots, x_N + y_N]$. The SMAX protocol [8, 13, 70] delivers randomly permuted additive shares a and b of $x + y$ to Alice and Bob. The maximum index is obtained through an additional processing step that also ensures that the maximum value is not exposed. Alternatively, the partial shares could be propagated as-is to subsequent steps of an algorithm.

- *Secure Maximum Value (SVAL)*: Alice and Bob wish to compute the value of the largest component of $z = x + y$. This can be achieved using the SVAL protocol [8], the outcome of which is to provide Alice and Bob with additive shares of a and b of the maximum value, but no information about its index. The primitives in turn have two essential steps: first both Alice and Bob encrypt their own data using special cryptographic techniques [49]. The computation is performed on the encrypted data and results are distributed. When primitives are chained so that the output of one primitive is fed to the next, the computation can frequently be arranged so that repeated encryption is not necessary – the partial results from one computation can directly be input into the next.

- *Permute Protocol* [8]: Here Alice and Bob end up with arrays $\bar{a}$ and $\bar{b}$, such that $\bar{a}+\bar{b} = \pi(a) + \pi(b)$, where $\pi(\cdot)$ is a permutation.

- *Secure Logsum (SLOG)*: This rather unusual operation is a useful primitive for speech processing algorithms. Given that Alice and Bob hold vectors x and y, the goal is to compute $z = \log \sum_i e^{x_i+y_i}$. The SLOG protocol [128] once again provides additive shares of the result to Alice and Bob.

Other such primitives can be defined. Larger computations, such as regression, classification, matching and retrieval, can be obtained through judicious combination of these primitives.

2.1.2 Random Number Generation

A Random Number Generator (RNG) is a device that produces a sequence of numbers that lack any pattern. Random number sequences have a wide variety of applications, ranging from computer games to cryptography. Likewise, there are also many types of RNG, each providing different levels of randomness. A more detailed analysis on this topic can be found in the literature [123, 50].

For the privacy-preserving tasks considered in this work, real random sequences such as the ones obtained through measuring radioactive decay, thermal noise and other subatomic physics phenomena based on the laws of quantum mechanics are not needed. It is, however, crucial that one has a way to generate cryptographically secure pseudo-random sequences of numbers. Such sequences should both appear random and be unpredictable. In order for a sequence of numbers to appear random, it should pass all possible statistical tests available. Examples of these tests include the Diehard battery of the statistical tests [75] and a list compiled by Donald Knuth [64]. For a sequence of numbers to be unpredictable, it should be computationally prohibitive to predict the next number of that sequence. This should hold even if both all the previous numbers of the sequence and a complete knowledge of the generating algorithm are known in advance. Unless stated otherwise, we considered the BlumBlumShub [15] Pseudo-Random Number Generator (PRNG) as a source of crypto-

graphically secure sequences of numbers. Although this algorithm is relatively slow, it passes all the statistical tests on the Diehard battery.

2.1.3 Homomorphic Encryption

A public-key cryptosystem consists of a triple of probabilistic polynomial time algorithms for key generation (KG), encryption (EN) and decryption (DE).

- The KG algorithm generates a pair of private and public keys, sk and pk respectively.
- The EN algorithm creates a ciphertext (encryption) m^E of a plaintext (input) m using public key pk, $m^E = EN(m, pk)$.
- The DE algorithm computes the original input m given the ciphertext of that input and the private key sk, $m = DE(EN(m, pk), sk)$.

The two parts of the key pair are obviously different from each other, although they are mathematically linked. Also, neither of these keys alone can perform both the EN and DE algorithms.

A HE system [43] is a special type of public-key cryptosystems that allows for specific algebraic operations to be performed indirectly in the plaintext, by manipulating the ciphertext. Partially homomorphic cryptosystems can be characterized as being multiplicatively or additively homomorphic.

A multiplicatively homomorphic scheme (*e.g.* RSA [118]) only supports multiplication on plaintexts, which is equivalent to the following property:

$$EN(m_1, pk) \times EN(m_2, pk) = EN(m_1 \times m_2, pk) \tag{2.1}$$

An additively homomorphic scheme (*e.g.* Paillier [96]) only supports addition on plaintexts, which can also be stated as:

$$EN(m_1, pk) \times EN(m_2, pk) = EN(m_1 + m_2, pk) \tag{2.2}$$

$$(EN(m_1, pk))^{m_2} = EN(m_1 \times m_2, pk) \tag{2.3}$$

Finally, a Fully Homomorphic Encryption (FHE) scheme [47] supports both addition and multiplication. This is a very strong and interesting property, as it enables an untrusted party to run full programs using encrypted input and producing the corresponding encrypted output. This would be possible since the untrusted party has no need to decrypt any value at any point during the program execution. Unfortunately, and despite recent advances made on this subject [45, 124, 46, 30], at the time of the writing of this document no efficient implementation of a fully homomorphic cryptosystem exists.

Paillier Cryptosystem

The Paillier public-key cryptosystem [96] is based on the Decisional Composite Residuosity Assumption (DCRA), which is believed to be computationally intractable, *i.e.*, it lacks a polynomial-time solution. Given a composite n of two large prime numbers p and q, $n = p \cdot q$, and an integer z, this problem can be briefly explained as deciding whether z is a n-residue modulo n^2 or not. This is equivalent to trying to find an integer y such that $z \equiv y^n \mod n^2$. The original formulation provides semantic security against Chosen-Plaintext Attacks (CPA), but due to its properties the system is malleable and therefore does not offer the highest level of semantic security. Malleability is usually seen as desirable for certain applications such as secure electronic voting. Nevertheless, an improvement on the original formulation is shown to make the new system resilient to Chosen-Ciphertext Attacks (CCA1) and Adaptive Chosen-Ciphertext Attacks (CCA2) in the random oracle model [97].

The three algorithms for key generation, encryption and decryption for the Paillier system are performed as follows.

- *KG* algorithm: Generate randomly and independently two different large prime numbers p and q, such that the $\gcd(p \cdot q, (p-1) \cdot (q-1)) = 1$. Using these numbers, compute $n = p \cdot q$ and $\lambda = \text{lcm}(p-1, q-1)$. Next, generate a cryptographically secure random integer $g \in \mathbb{Z}^*_{n^2}$. Finally, check if the values of n and g are appropriate by testing the existence of the modular multiplicative inverse $\mu = (L(g^\lambda \mod n^2))^{-1} \mod n$, with $L(u) = \frac{u-1}{n}$. This process should be repeated until $\mu \neq 0$. At the end of this process the private and public keys are given by $sk = (\lambda, \mu)$ and $pk = (n, g)$, respectively.

- *EN* algorithm: Generate a cryptographically secure random integer $r \in \mathbb{Z}^*_n$. Then, take the plaintext message m and obtain the corresponding ciphertext m^E by computing $m^E = EN(m, pk) = g^m \cdot r^n \mod n^2$. The encryption process also adds a masking effect to the data, as each individual value in the plaintext will be represented as one of a set of n different possibilities in the ciphertext, depending on the value of r.

- *DE* algorithm: Take the ciphertext c and retrieve the original plaintext message m by computing $m = L(c^\lambda \mod n^2) \cdot \mu \mod n$.

The homomorphic properties expressed in Equations 2.2 and 2.3 can now be reformulated as follows:

$$DE(EN(m_1, pk) \cdot EN(m_2, pk) \mod n^2, sk) = m_1 + m_2 \mod n \tag{2.4}$$

$$DE(EN(m_1, pk)^{m_2} \mod n^2, sk) = m_1 \cdot m_2 \mod n \tag{2.5}$$

2.1.4 Oblivious Transfer

OT is a cryptographic method that enables privacy-preserving selection from sets. Different OT protocols can be defined depending on the size of the set and on the number of data values to be transmitted each time the protocol is run. Three different OT protocols exist, which can be summarized as follows:

- *1-out-of-2 OT* [39]: Bob sends one of two bits to Alice. Alice receives either one of these bits, with equal probability, and she knows which one she receives, while Bob does not

know which bit Alice received.

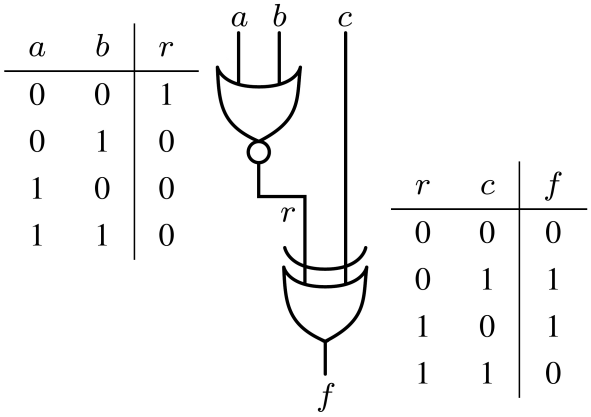
- *1-out-of-n OT* [85]: Similar to the 1-out-of-2 OT protocol, but in this case Bob transmits one of n different data values to Alice.
- *k-out-of-n OT* [21]: Generalization of the 1-out-of-n OT protocol, where transmits k data values instead of just one to Alice.

Chosen versions of this protocols also exist. The difference between them and their original formulation is that Alice chooses the bit (or data value) she wishes to receive. On one hand, this protocol allows Bob to transmit to Alice the information associated with her request while giving him the guarantee that Alice will not learn anything about the other piece of information. On the other hand, Alice is assured that Bob will not know what information was actually requested since he will not be aware of the request itself.

OT is very useful in a wide variety of privacy-preserving applications, including selling of digital goods [4], solving the list intersection problem [44], exchanging mutually authenticated keys [27], etc. Detailed explanations on how the different OT protocols work and numerous optimizations that can be performed while implementing them are available in the literature [72, 7, 20].

2.1.5 Garbled Circuits

The earliest described and possibly the most popular approach to STPC is to express the function to be evaluated as a *garbled* Boolean circuit, usually referred to as a Garbled Circuit (GC) [138]. In short, the idea is to encrypt (or garble) the nodes and transition paths of a Boolean circuit so that the party who evaluates it can only follow a single path, which is exclusively defined by the circuit itself and the input values. Given a target function $y = f(\mathbf{a}, \mathbf{b})$, where $\mathbf{a}$ are Alice's private inputs and $\mathbf{b}$ are Bob's private inputs, it is possible to represent y using a Boolean circuit. Alice and Bob are both interested in evaluating the circuit without disclosing each their private inputs to one another. At the end of the evaluation process, the value y can be made available to one or both parties.



a	b	r
0	0	1
0	1	0
1	0	0
1	1	0

r	c	f
0	0	0
0	1	1
1	0	1
1	1	0

Figure 2.1: Example of a logic circuit and respective truth tables.

a	b	r
K_a^0	K_b^0	$\mathcal{E}_{K_a^0}(\mathcal{E}_{K_b^0}(K_r^1))$
K_a^0	K_b^1	$\mathcal{E}_{K_a^0}(\mathcal{E}_{K_b^1}(K_r^0))$
K_a^1	K_b^0	$\mathcal{E}_{K_a^1}(\mathcal{E}_{K_b^0}(K_r^0))$
K_a^1	K_b^1	$\mathcal{E}_{K_a^1}(\mathcal{E}_{K_b^1}(K_r^0))$

r	c	f
K_r^0	K_c^0	$\mathcal{E}_{K_r^0}(\mathcal{E}_{K_c^0}(0))$
K_r^0	K_c^1	$\mathcal{E}_{K_r^0}(\mathcal{E}_{K_c^1}(1))$
K_r^1	K_c^0	$\mathcal{E}_{K_r^1}(\mathcal{E}_{K_c^0}(1))$
K_r^1	K_c^1	$\mathcal{E}_{K_r^1}(\mathcal{E}_{K_c^1}(0))$

Table 2.1: Example of garbled truth tables.

The way GC work is best explained by using an example. Alice has Boolean input a and Bob has Boolean inputs b and c. They wish to compute the function $f(a, b, c) = \overline{a \vee b} \oplus c$. The logic circuit for $f(a, b, c)$ and the truth tables for the gates in it are shown in Figure 2.1. However, Alice does not want to reveal a to Bob, and Bob does not want to reveal b or c to Alice. In order to still be able to compute $f(a, b, c)$, they must therefore *garble* the circuit. Bob starts by generating the logic circuit implementing $f(\cdot)$, garbling it, and sending it to Alice. During the garbling process he generates a pair of private keys for each bit value (0/1), for each input and intermediate value (a, b, c, r in the example), totalling eight keys: $K_a^{0/1}$, $K_b^{0/1}$, $K_c^{0/1}$, $K_r^{0/1}$. For gates that generate intermediate values ($r = \overline{a \vee b}$ in our example), he replaces the outputs of the truth table with the encryption of the key corresponding to the output, performed with the keys corresponding to the inputs. For example, for $a = 0$, $b = 1$ the output is $r = 1$; the corresponding input keys are K_a^0, K_b^1 and the encrypted output is $\mathcal{E}_{K_a^0}(\mathcal{E}_{K_b^1}(K_r^1))$. For the output gate ($f = r \oplus c$ in our example), he encrypts the output bit itself. The garbled values for our example are presented in Table 2.1. To compute the function, Bob transmits the keys $K_b^?$ and $K_c^?$ corresponding to his input bits to Alice. Alice

recovers the keys corresponding to her input bit, $K_a^?$, as well as the truth tables for the gates of the circuit from Bob using the chosen k-out-of-n OT protocol. However, for the specific case of GC where only bits are transmitted, this protocol is equivalent to performing several consecutive chosen 1-out-of-2 OT protocols [86]. Because of the properties of OT, Bob does not learn either the value of Alice's bits or the portion of the truth table she actually requires to perform her computations, *i.e.*, he learns nothing, either directly or indirectly, of Alice's inputs. To evaluate the circuit, Alice successively deciphers each of the gates in the circuit. The nature of the computation is such that for each gate, Alice will possess two keys, one for each input. She decrypts all four encrypted outputs in the truth table for the gate using the two keys. The keys will be inappropriate for three of the four outputs; therefore Alice can only correctly decipher one of the four encrypted values, which in turn will be one of the keys for the next gate. After repeating this process for all gates, she finally obtains the desired output value f.

To continue with our example, if Alice has input $a = 0$ and Bob has inputs $b = 1$ and $c = 0$, Bob transmits K_b^1 and K_c^0 to Alice. Since the circuit is garbled, Alice does not know whether these keys represent bit values 0 or 1. Alice obtains K_a^0 from Bob using OT. Alice decrypts all four entries in the truth table for $\overline{a \vee b}$ using K_a^0 and K_b^1. Three of the four entries will result in obviously meaningless decryptions. The circuit is designed to make the incorrectness of the decryptions apparent; for instance only correctly decoded outputs may have their Most Significant Bits (MSB) set to zero, whereas incorrectly decrypted ones will not, and consequently their representation falls outside the expected range of values. The remaining entry, $\mathcal{E}_{K_a^1}(\mathcal{E}_{K_b^0}(K_r^0))$, will be decrypted to recover K_r^0. Note that Alice does not realize that K_r^0 represents the value $r = 0$, since the table has been garbled. She subsequently decrypts all four entries of the table for $f = r \oplus c$ using K_r^0 and K_c^0. As before, only one of the four entries will be decrypted correctly. The result of the decryption will be $f = 0$ (or, if the output must remain hidden from Alice, an encryption of 0 which only Bob can decipher). Alice and Bob never learn each others' inputs. The entity designated to know the outcome of the computation receives this value; no more information is leaked.

For a long time it was believed that GC were of purely theoretical interest, but recent advances have made GC much more efficient and practical to use. For instance, all the OT for transferring the input ciphers may be pre-computed [9], meaning that all the computationally expensive operations regarding the OT are performed offline, and only an additional yet simple operation per OT is performed while evaluating the GC. Furthermore, all the OT may be computed efficiently by using shorter ciphers [87, 4], which can be implemented using elliptic curve cryptography [65]. In the gate decryption phase, the point-and-permute technique [73] may be used to reduce the total number of required decryptions from 4 to 1 by associating a permutation bit to the input ciphers chosen using OT.

Other "shortcuts" may be obtained from the nature of the gates. NOT computations impose no additional expense, as we noted in our example (where we computed $r = \overline{a \vee b}$ directly, rather than the sequence $d = a \vee b;\quad r = \overline{d}$). All the XOR gates of a GC may be evaluated for "free" [66], *i.e.*, without significant computational cost, as the XOR gate does not need a garbled table and its evaluation consists of XOR-ing its garbled input values. Consequently, NXOR gates are "free" as well. Additional reductions in execution time, bandwidth and energy use are possible if the AND gates are broken into two half-gates [140]. Custom designing circuit operations to preferentially utilize these gates will result in faster evaluation of the resulting GC [67].

On top of these implementation tricks, other useful properties of GC have been found. For example, it is possible to reuse the same GC several times under some conditions [51] by using functional encryption [121, 16], which means that if the same operation must be computed several times we only need a single circuit instead of one circuit for each instance that we want to compute that function. It is also possible to provide security against malicious adversaries using the cut-and-choose technique [71]. Several other privacy and security guarantees regarding GC have also been proven [19, 60, 10, 83].

In implementing the basic operations using GC, some issues different from the ones faced when using hardware circuits arise. Decision making operations such as comparisons (greater/lesser-than, maximum/minimum) and multiplexing can be implemented similarly to digital circuits.

However, branching (if-then-else) and cycles/recursion (for loop, do-while) should be avoided. The first requires building and evaluating circuits for all possibilities in order to hide which one was actually followed. The second requires explicitly expanding the cycles, one circuit per iteration, which may result in a huge intractable circuit. Another problem arises when computing non-linear functions, since the usual trick of computing the initial elements of the corresponding Taylor series is impractical to implement using GC. An efficient work-around is to replace these operations with Piecewise Linear Approximation (PLA), whose coefficients can be stored in look-up tables. An additional restriction exists on the representation of the ciphered values. Operations on floating-point values usually require some normalization on them before they are performed, introducing an additional and unnecessary computational cost. To counter this, a fixed-point representation is preferred, despite a possible loss of precision in representing those same values.

Finally, given the popularity of GC, a variety of software implementations of all the required protocols and encryption systems are available. One of the first practical implementations of GC was Fairplay [73], and since then a number of different implementations have been developed in order to further improve their efficiency and practicality [57, 103, 11].

2.2 Distance-Preserving Hashing Techniques

Distance-preserving hashing techniques are dimensionality reduction mapping functions such that the distance relationships between the original input data points are preserved and transferred (under certain conditions) to their corresponding hashes. This means that if two data points are located close to each other, their hashes will also be close to each other; if these data points are far apart, no information regarding the distance between them can be inferred from their hashes. We will now address two such techniques.

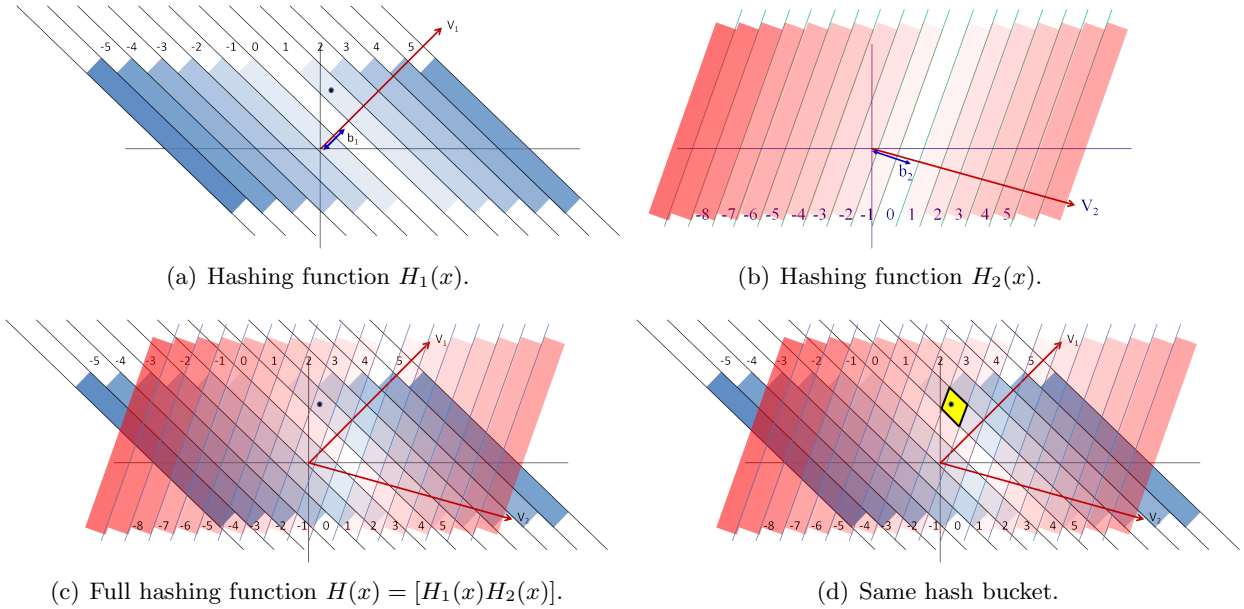
2.2.1 Locality-Sensitive Hashing

Locality-Sensitive Hashing (LSH) is a method of performing probabilistic dimension reduction of high-dimensional data. The LSH algorithm relies on the existence of locality-sensitive functions $H(\cdot)$, which apply random transformations to a data vector x, projecting it to a vector $H(x)$ in a lower dimensional space, which we refer to as the LSH key or bucket. A set of data points that map to the same key are considered as approximate nearest neighbours. As a single LSH function does not group the data points into fine-grained clusters, hash keys obtained by concatenating the output of n LSH functions are normally considered. This n-bit LSH function $H(x) = [H_1(x)H_2(x)\cdots H_n(x)]$ maps a d-dimensional vector into a n-length bit string. If the Euclidean norm is considered, then each of the individual hashing function is computed by

$$h_i(x) = h_i(x; V_i, b_i) = \left\lfloor \frac{x^\top V_i + b_i}{w} \right\rfloor \tag{2.6}$$

where V_i is a random normal vector, b_i is a random uniform value between 0 and w, and w is the quantization width. Also, h different LSH keys are computed over the same input to achieve better recall. Two data vectors x and x' are said to be neighbours if at least one of their h keys, each of length n, matches exactly. In order to better illustrate the concept of LSH we describe a simple 2-D example, presented in Figure 2.2. Let a data vector be represented on the 2-D space by the small black dot. In Figure 2.2(a) a possible hashing function H_1 is presented, defined by vector V_1 and scalar b_1. Notice that the regions are perpendicular to V_1 and the reference region is defined by b_1. Figure 2.2(b) contains a different hashing function H_2, together with its corresponding vector V_2 and scalar b_2. For this example we only consider $n = 2$ projections, so the final hashing function is given by $H(x) = [H_1(x)H_2(x)]$, as presented in Figure 2.2(c). Finally, all the vectors located within the same bucket will be represented by the same hash. The bucket corresponding to the data vector is highlighted in yellow in Figure 2.2(d).

We now define LSH in a more formal manner. Let $\mathcal{H}$ be a family of hash functions mapping $\mathbb{R}^d$ to some universe U. For any two points p and q, consider a process in which we choose a



(a) Hashing function $H_1(x)$.

(b) Hashing function $H_2(x)$.

(c) Full hashing function $H(x) = [H_1(x)H_2(x)]$.

(d) Same hash bucket.

Figure 2.2: 2-D example using an LSH projection.

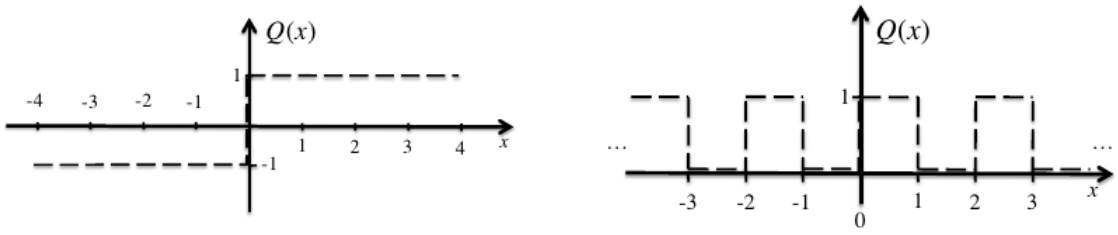
function $H(\cdot)$ from $\mathcal{H}$ uniformly at random, and analyse the probability that $H(p) = H(q)$. Let R and c be the radius and the control parameter for a c-approximate R-near-neighbour query. A family $\mathcal{H}$ is called (R, cR, P_1, P_2)-sensitive if, for any two points p, $q \in \mathbb{R}^d$, it satisfies the following conditions:

- if $\|p - q\| \leq R$ then $Pr_{\mathcal{H}}[H(p) = H(q)] \geq P_1$
- if $\|p - q\| \geq cR$ then $Pr_{\mathcal{H}}[H(p) = H(q)] \leq P_2$

In order for a family of LSH hash functions to be useful, it has to satisfy $P_1 > P_2$.

2.2.2 Secure Binary Embeddings

Secure Binary Embedding (SBE) [18] is a scheme for converting vectors to bit sequences using band-quantized random projections. It produces a LSH method with an interesting property: if the Euclidean distance between two vectors is lower than a threshold, then the Hamming distance between their hashes is proportional to the Euclidean distance between the vectors; if it is higher, then the hashes provide no information regarding the true distance between the



(a) Conventional scalar 1-bit quantization. (b) Universal 1-bit quantization.

Figure 2.3: 1-bit quantization functions.

two vectors. This scheme is based on the concept of Universal Quantization (UQ) [17], which redefines scalar quantization by forcing the quantization function to have non-contiguous quantization regions.

Given an L-dimensional vector $\mathbf{x} \in \mathbb{R}^L$, the UQ process converts it to an M-bit binary sequence, where the m-th bit is given by

$$q_m(\mathbf{x}) = Q\left(\frac{\langle \mathbf{x}, \mathbf{a}_m \rangle + w_m}{\Delta}\right) \tag{2.7}$$

Here $\langle , \rangle$ represents a dot product. $\mathbf{a}_m \in \mathbb{R}^L$ is a random projection vector comprising L i.i.d samples drawn from $\mathcal{N}(\mu = 0, \sigma^2)$, Δ is a precision parameter, and w_m is a random dither drawn from a uniform distribution over $[0, \Delta]$. $Q(\cdot)$ is a quantization function given by $Q(x) = \lfloor x \mod 2 \rfloor$. We can represent the complete quantization into M bits compactly in vector form:

$$\mathbf{q}(\mathbf{x}) = Q\left(\mathbf{\Delta}^{-1}(\mathbf{A}\mathbf{x} + \mathbf{w})\right) \tag{2.8}$$

where $\mathbf{q}(\mathbf{x})$ is an M-bit binary vector, which we will refer to as the *hash* of $\mathbf{x}$, $\mathbf{A} \in \mathbb{R}^{M \times L}$ is a matrix composed of the row vectors $\mathbf{a}_m$, $\mathbf{\Delta}$ is a diagonal matrix with entries Δ, and $\mathbf{w} \in \mathbb{R}^M$ is a vector composed from the dither values w_m. The universal 1-bit quantizer of Equation 2.7 maps the real line onto 1/0 in a banded manner, where each band is Δ_m wide. Figure 2.3 compares the conventional scalar 1-bit quantization function with the equivalent universal 1-bit quantization function. If we consider once again the example described in Figure 2.2, but replace the quantization function in Figure 2.3(a) with the one in Figure

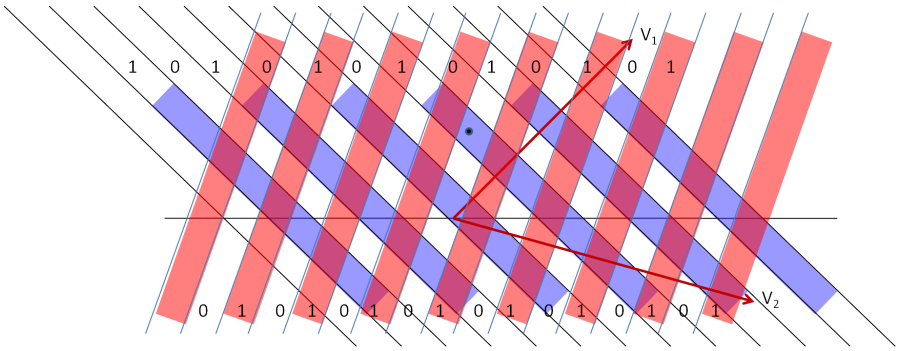


Figure 2.4: 2-D example using an SBE projection.

2.3(b), we obtain the corresponding result for SBE, presented in Figure 2.4. The binary hash generated by the UQ of Equation 2.8 has the following properties [18]: the probability that the i^{th} bits, $q_i(\mathbf{x})$ and $q_i(\mathbf{x}')$ of hashes of two vectors $\mathbf{x}$ and $\mathbf{x}'$, respectively, are identical depends only on the distance $d = \|\mathbf{x}-\mathbf{x}'\|$ between the vectors and not on their actual values. As a consequence, the following relationship can be shown [18]: given any two vectors $\mathbf{x}$ and $\mathbf{x}'$ with a Euclidean distance d, with probability at most e^{-2t^2M} the normalized (per-bit) Hamming distance $d_H(\mathbf{q}(\mathbf{x}), \mathbf{q}(\mathbf{x}'))$ between the hashes of $\mathbf{x}$ and $\mathbf{x}'$ is bounded by:

$$\frac{1}{2} - \frac{1}{2}e^{-\left(\frac{\pi\sigma d}{\sqrt{2}\Delta}\right)^2} - t \leq d_H(\mathbf{q}(\mathbf{x}), \mathbf{q}(\mathbf{x}')) \leq \frac{1}{2} - \frac{4}{\pi^2}e^{-\left(\frac{\pi\sigma d}{\sqrt{2}\Delta}\right)^2} + t \tag{2.9}$$

where t is the control factor. The importance of the above bound is that the Hamming distance $d_H(\mathbf{q}(\mathbf{x}), \mathbf{q}(\mathbf{x}'))$ is correlated to the Euclidean distance $\|\mathbf{x} - \mathbf{x}'\|$ between the two vectors if the Euclidean distance between the two vectors is below a threshold (which depends on Δ). Specifically, for small d, $E[d_H(\mathbf{q}(\mathbf{x}), \mathbf{q}(\mathbf{x}'))]$, the expected Hamming distance, can be shown to be bounded from above by $\sqrt{2\pi^{-1}}\sigma\Delta^{-1}d$, which is linear in d. However, if the distance between $\mathbf{x}$ and $\mathbf{x}'$ is greater than this threshold, $d_H(\mathbf{q}(\mathbf{x}), \mathbf{q}(\mathbf{x}'))$ is bounded by $0.5 - 4\pi^{-2}exp\left(-0.5\pi^2\sigma^2\Delta^{-2}d^2\right)$, which rapidly converges to 0.5 and effectively gives us no information whatsoever about the true distance between $\mathbf{x}$ and $\mathbf{x}'$.

In order to illustrate how this scheme works, we randomly generated samples in a high-dimensional space ($L = 1024$) and plotted the normalized Hamming distance between their hashes against the Euclidean distance between the respective samples. This is presented

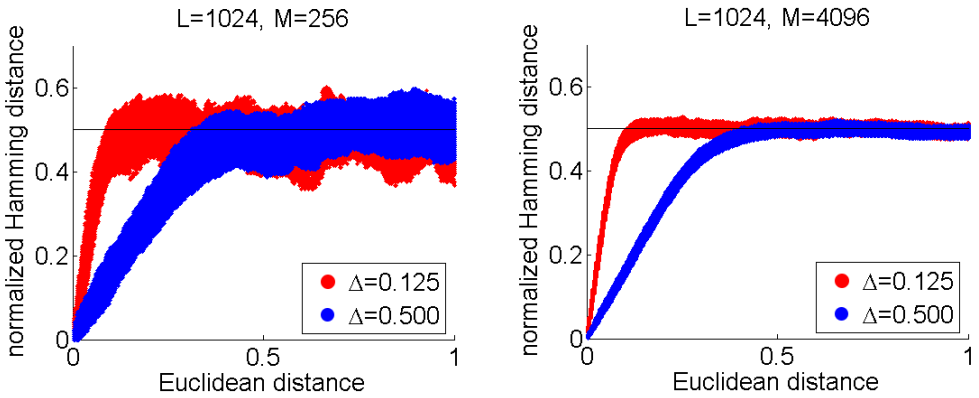


Figure 2.5: Embedding behaviour for different values of Δ and different amounts of measurements M.

in Figure 2.5. The number of bits in the hash is also shown in the figures. We note that in all cases, once the normalized distance exceeds Δ, the Hamming distance between the hashes of two vectors ceases to provide any information about the true distance between the vectors. We will find this property useful in developing our Privacy-Preserving Speaker Verification (PPSV) system. We also see that changing the value of the precision parameter Δ allows us to adjust the distance threshold until which the Hamming distance is informative. Also, increasing the number of bits M leads to a reduction of the variance of the Hamming distance.

A converse property of the embeddings is that for all $\mathbf{x}'$ except those that lie within a small radius of any $\mathbf{x}$, $d_H(\mathbf{q}(\mathbf{x}), \mathbf{q}(\mathbf{x}'))$ provides little information about how close $\mathbf{x}'$ is to $\mathbf{x}$. It can, in fact, be shown that the embedding provides information theoretic security beyond this radius, if the embedding parameters $\mathbf{A}$ and $\mathbf{w}$ remain unknown to the potential eavesdropper. Any algorithm attempting to recover a signal $\mathbf{x}$ from its embedding $\mathbf{q}(\mathbf{x})$ or to infer anything about the relationship between two signals sufficiently far apart using only their embeddings will fail to do so. Furthermore, even in the case where $\mathbf{A}$ and $\mathbf{w}$ are known, it seems computationally intractable to derive $\mathbf{x}$ from $\mathbf{q}(\mathbf{x})$ unless one can guess a starting point very close to $\mathbf{x}$. In effect, it seems infeasible to invert the SBE without strong a priori assumptions about $\mathbf{x}$.

There are two parameters that control the behaviour of the SBE hashes: the quantization step size Δ and the number of bits M. The value of M by itself is not a useful number, as

different values of L require different values of M; hence we report our results as a function of bits per coefficient (*bpc*), computed as $bpc = M/L$. The *bpc* allows us to govern the variance of the UQ.

Leakage

In the following chapters we will define *leakage* as the fraction of utterances containing the target keywords whose SBE hashes have a normalized Hamming distance below the threshold at which Hamming distance d_H is proportional to the Euclidean distance d with respect to any utterance in which we want to detect keywords. This is an important concept, as it is possible that some information regarding the original features may be extracted by analysing the distances between their corresponding hashes. This will greatly depend on the type of features and task being considered, and therefore each situation should be analysed separately.

3 Privacy-Preserving Music Matching

This chapter covers the work carried out in the scope of Privacy-Preserving Music Matching (PPMM). Section 3.2 presents the approach considered for the music matching problem. Section 3.3 describes the setup of the performed experiments and Section 3.4 shows the obtained results. Finally, Section 3.5 contains an analysis on the privacy and security achieved by the algorithm.

3.1 Introduction

The music matching problem can be quickly described as considering a party (Alice) that has access to a music recording and wishes to find out relevant information about it (*e.g.* song title, artist, year of release, etc.) by querying an on-line database (Bob). An easy and efficient way of doing this can be achieved by Alice providing Bob with a small segment of her song, for instance a 5 seconds segment, which he then compares with the entire collection on his database. This can be achieved, for example, by performing a cross-correlation between the signals. Notice that this segment matching approach closely relates to Query-by-Example Speech Search (QESS), as it also consists of searching for a pre-specified pattern in a given set of recordings.

The PPMM problem is basically the same problem but with the restriction that both Alice and Bob have privacy concerns about the query. Alice may wish Bob not to know which song she is querying, and Bob may not wish his database to be open to direct access queries. This restriction leads to some important changes regarding the original problem. First, there is an increase in the complexity of the equations that represent each individual algorithm used for solving the music matching problem. Secondly, all the computations must be performed

using ciphered values, which means considering only operations based on modular arithmetic using a homomorphic cryptosystem. Finally, the communication between the two parties has to be made through Secure Multi-Party Computation (SMPC) protocols.

3.2 Music Matching

There are many techniques that can be used for performing music matching. The simplest approach is to perform a cross-correlation at the audio level between the target music snippet and the labelled music files contained in a music database. This technique has very little resistance to noise but it is effective under controlled conditions. Other approaches include performing melodic matching [132], computing cross-correlations at the feature level [37] and comparing music hash fingerprints [134]. The first technique consists of performing melody extraction and standardization[1], followed by computing similarity measures. The second technique extracts features from the audio, performs frame alignments based on beat detection and then computes cross-correlations among the aligned sets of frames. Finally, the last technique was popularized by the music identification services such as Shazam [135]. It starts by extracting a set of interest points from the audio spectrogram and then generates a combinatorial hash using those points, which are used for searching and scoring operations.

In this work we will consider the technique that computes cross-correlations at the audio level. Although this technique is only useful for tackling the music matching problem for very specific situations, it is the easiest to adapt to a privacy-preserving context and therefore enables a clear analysis of its performance using a variety of metrics.

3.3 Experimental Setup

This section describes how Alice and Bob can perform an online music query in such a way that they both preserve the privacy of their data. A theoretical description on how this should

[1]Standardisation refers to rewriting the musical sequence in a standard form that preserves the "feel" of the melody but eliminates performance-specific characteristics.

be achieved using a cross-correlation approach [126]. In this section we will focus primarily on the implementation aspects. In Section 3.5 we highlight a flaw in the original protocol formulation and a way to correct it. As in the work of Shashanka and Smaragdis [126], we will also structure our implementation in steps, and for each step we will provide details on both the required operations and privacy-preserving algorithms we considered. The first step describes the computation of the cross-correlation between two signals, the second step shows how to obtain the maximum element of a given vector, the third step describes how to find the index of the maximum element of a vector and finally the fourth step shows how to obtain the information in a specific database entry.

Throughout this section all equations will be presented in the ciphertext domain. We will consider the Paillier cryptosystem [96], which has additively homomorphic properties, as stated previously in Equations 2.2 and 2.3. In short, this means that every addition in the plaintext domain is converted into a multiplication in the ciphertext domain, and that every multiplication by a scalar in the plaintext domain is converted into an exponentiation in the ciphertext domain.

We consider both Alice and Bob to behave as *honest-but-curious* entities (*i.e.*, they follow the protocol honestly but each of them attempts to learn additional information from the ciphertexts he or she receives from the other). Because of the way the PPMM protocols work, Alice and Bob cannot perform Chosen-Ciphertext Attacks (CCA1) or Adaptive Chosen-Ciphertext Attacks (CCA2), as neither of them can chose arbitrary ciphertexts and have access to its corresponding plaintext unless they originally already own that plaintext. Therefore, they will only be able to perform Chosen-Plaintext Attacks (CPA), against which the original formulation of the Paillier cryptosystem is resilient [96].

3.3.1 Step 1: Cross-Correlation of the Music Signals

In the first step we will perform the cross-correlation between Alice's music segment and each of the songs in Bob's database. A visual representation of the process for a given music with index k in Bob's database is presented in Figure 3.1. Alice starts by generating a private and

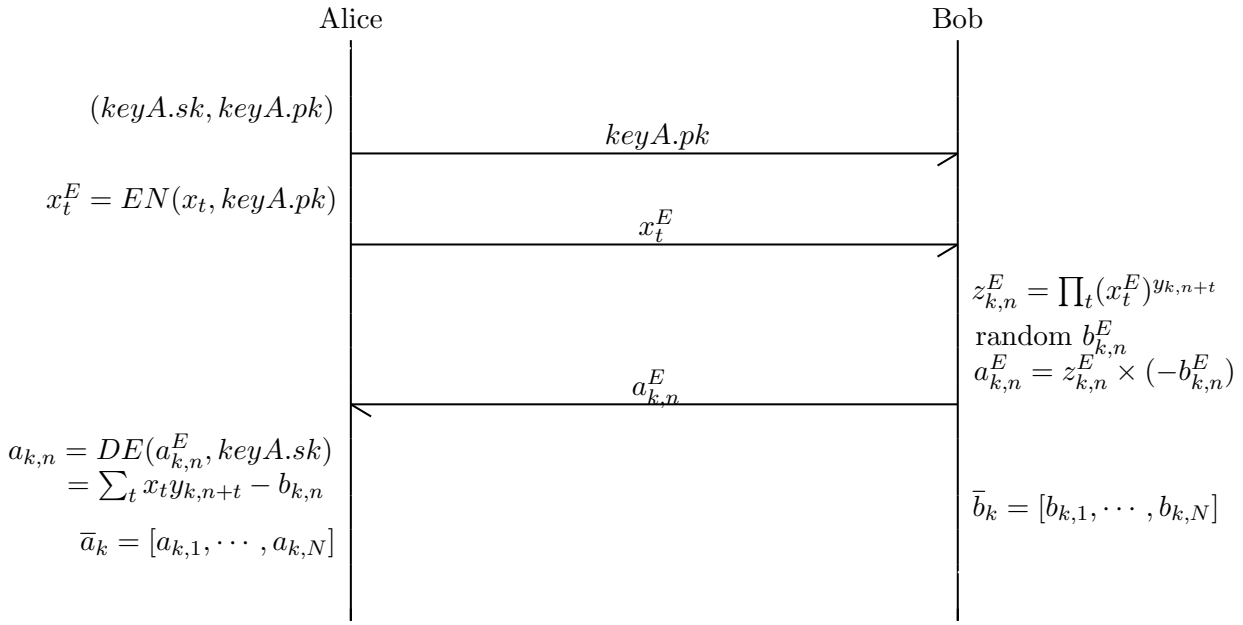


Figure 3.1: Message sequence chart for Step 1.

public key pair (sk,pk) and sends the public key to Bob. She then encrypts each element of her music segment x_t with pk, obtaining $x_t^E = EN(x_t, pk)$ for $t = \{1, \ldots, T\}$, where T is the number of samples in her music segment, and sends them to Bob. He can now compute the cross-correlation in a privacy-preserving manner.

In order for Bob to compute the cross-correlation between Alice's values and his own values y_t, there are two possible options: he can compute it either in the time domain or in the spectral domain. The first option is easier to implement, as the cross-correlation can be computed by:

$$z_n^E = \prod_{t=1}^{T} (x_t^E)^{y_{n+t}} \tag{3.1}$$

which is the ciphertext version of the expected equation for the cross-correlation. This solution is not usually adopted, because it is very time consuming. The number of operations required to compute the full cross-correlation in the time domain is $\mathcal{O}(T^2)$. The second option consists of computing the Fast Fourier Transform (FFT) of both Alice's segment and each song in Bob's database, performing the Cross Power Spectrum (CPS) on the resulting signals and finally computing the Inverse Fast Fourier Transform (IFFT). In terms of ciphertext this is

represented by:

$$z^E = IFFT_{secure}\big((FFT_{secure}(x^E))^{FFT(y)}\big) \tag{3.2}$$

where FFT_{secure} denotes the privacy-preserving version of the FFT. This solution is less straightforward but faster than the first one, as the number of operations required to compute the full cross-correlation in the frequency domain (*i.e.*, using the FFT) is $\mathcal{O}(T\log_2(T))$.

In the case of desiring one sample precision in the computation of the cross-correlation between Alice's music segment and the songs in Bob's database, we would definitely implement the second solution. However, there is no need for such precision when computing the cross-correlation between two music segments that are sufficiently long, as consecutive cross-correlation results are not significantly different. Thus, we can choose a fixed larger window shift for the computation of consecutive cross-correlations and save a great deal of computational time. In particular, it would be necessary to perform the same number of operations for computing the cross-correlation either in the time domain or in the frequency domain when $T/(\text{window shift}) = \log_2(T)$. This happens because increasing the window shift effectively reduces the number cross-correlations N_{cc} that need to be computed in the time domain, but does not affect the computation of the FFT. An analysis on the choice of value for the window shift is presented in Section 3.4. There is also a problem with the complexity inherent to performing an FFT in a privacy-preserving way, for which an interesting analysis is presented in [12]. Computing the FFT in ciphertext not only requires many more operations than computing its equivalent in plaintext, but also each operation is more complex, which further contributes to avoid this solution. These two reasons led us to implement the privacy-preserving cross-correlation in the time domain.

After computing the cross-correlation, Bob generates cryptographically secure random numbers b_i, one for each cross-correlation value, which will become his part of the random additive shares. Using these numbers, he then computes Alice's shares of the cross-correlation a_i such that $z_i = a_i + b_i$, and sends them to her.

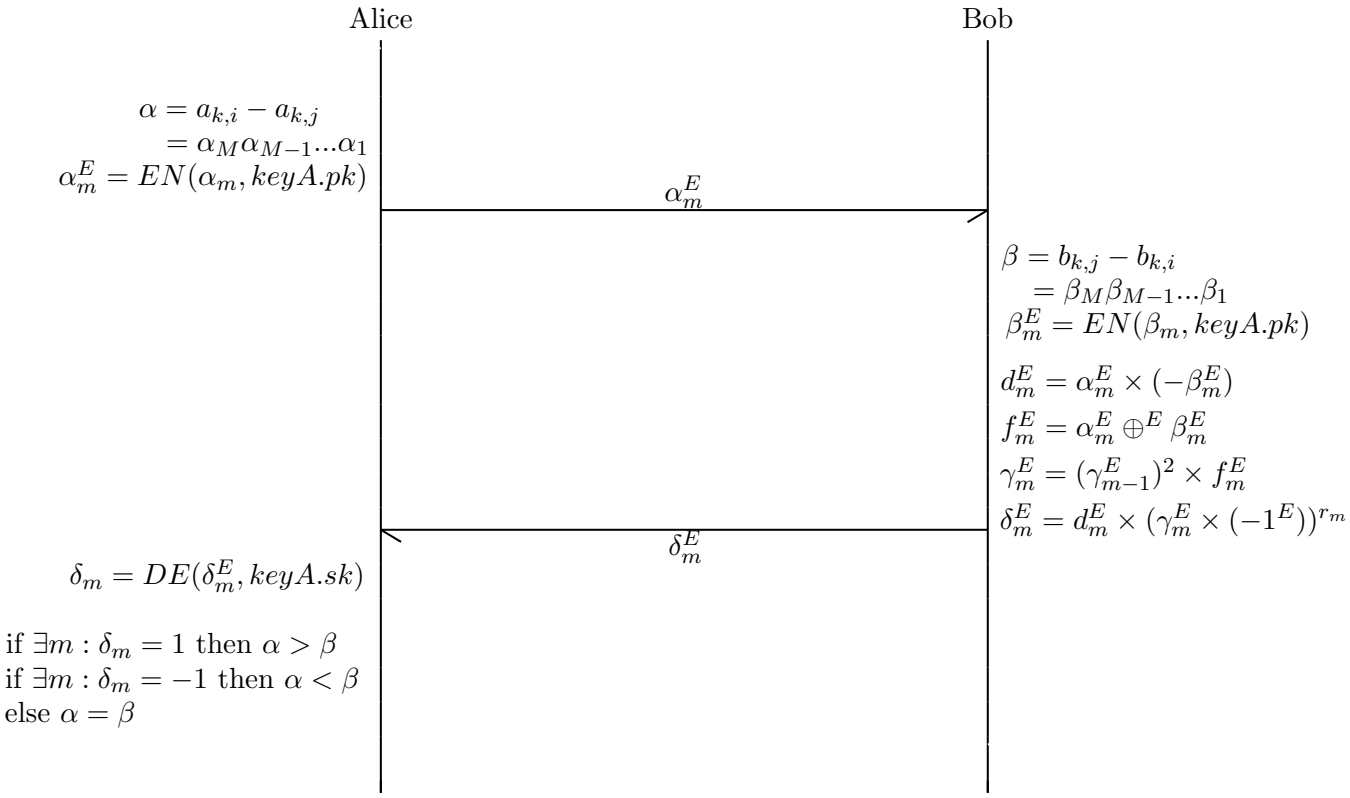


Figure 3.2: Message sequence chart for Step 2.

3.3.2 Step 2: Obtaining the Cross-Correlation Peaks

In the second step, Bob must compute the cross-correlation peak for each song in his database. Figure 3.2 illustrates this process. Since the output is randomly distributed between him and Alice, he cannot perform greater-than comparisons directly. He must take advantage of the fact that $z_i \geq z_j \Leftrightarrow (a_i - a_j) \geq (b_j - b_i)$, with z_i and z_j representing two cross-correlations of different segments of the same song with another song in the database. To perform the comparisons in a privacy-preserving manner, one can implement any solution to the Yao's millionaire problem [138]. Here we implemented an adaptation of the Blake-Kolesnikov algorithm [13]. This algorithm works by comparing two values at the bit level. Consider the bit representation of values $\alpha = a_i - a_j$ and $\beta = b_j - b_i$ to be $\alpha = \alpha_M \alpha_{M-1} \ldots \alpha_1$

and $\beta = \beta_M \beta_{M-1} \dots \beta_1$ respectively. For each bit m, $m = 1, \dots, M$, Bob computes:

$$d_m^E = \alpha_m^E \times (-\beta_m^E) \tag{3.3}$$

$$f_m^E = \alpha_m^E \times ((\alpha_m^E)^{\beta_m})^{-2} \times \beta_m^E \tag{3.4}$$

$$\gamma_m^E = (\gamma_{m-1}^E)^2 \times f_m^E,\ \gamma_0^E = 0^E \tag{3.5}$$

$$\delta_m^E = d_m^E \times (\gamma_m^E \times -1^E)^{r_m} \tag{3.6}$$

with r_m a cryptographically secure random number. The plaintext equivalent of the equations written above is:

$$d_m = \alpha_m - \beta_m \tag{3.7}$$

$$f_m = \alpha_m \oplus \beta_m = \alpha_m - 2\alpha_m\beta_m + \beta_m \tag{3.8}$$

$$\gamma_m = 2\gamma_{m-1} + f_m \tag{3.9}$$

$$\delta_m = d_m + r_m(\gamma_m - 1) \tag{3.10}$$

The δ_m^E's are the output of the algorithm, and they contain the encrypted information on which number is larger. Bob permutes all of them and sends them to Alice. She can now learn which number is larger from the encrypted values she receives. If the two numbers are different from each other, then one and only one of the δ_m's will be either 1 if $\alpha > \beta$ or -1 if $\alpha < \beta$, and the rest will be random numbers. More formally, if $\exists m : \delta_m = 1$, then $\alpha_m > \beta_m \Rightarrow \alpha > \beta$; if $\exists m : \delta_m = -1$, then $\alpha_m < \beta_m \Rightarrow \alpha < \beta$; else $\alpha = \beta$.

3.3.3 Step 3: Finding the Most Likely Music Index

In this step Alice wishes to know which index in Bob's database corresponds to her song. The communication protocol between Alice and Bob is illustrated in Figure 3.3. This index is the one for which the vector of cross-correlation peaks has its maximum. Alice can obtain this information without Bob knowing which song she has by means of a permute protocol

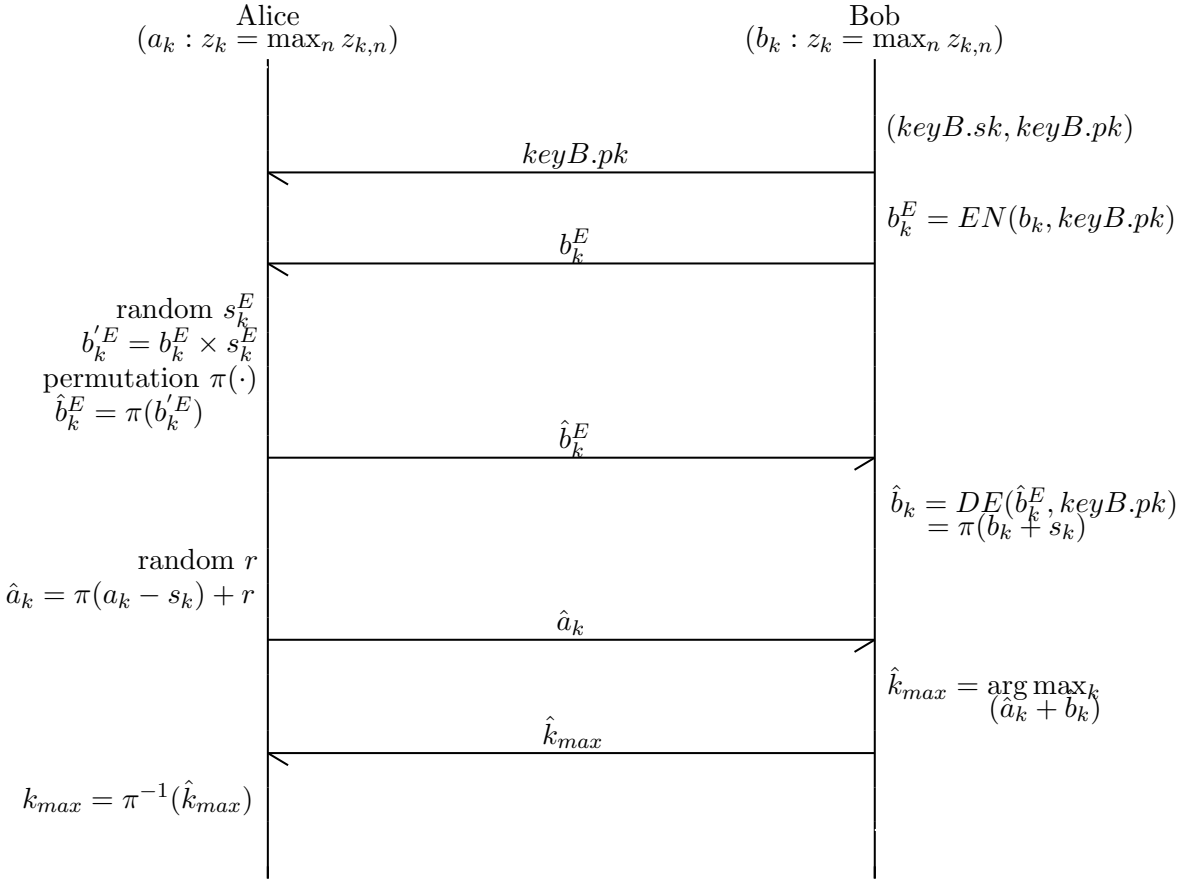


Figure 3.3: Message sequence chart for Step 3.

[8]. The main idea is that given Alice's and Bob's additive secret share vectors a and b, they can compute different secret share vectors $\overline{a}$ and $\overline{b}$ such that $\overline{a}+\overline{b} = \pi(a)+\pi(b)$, where $\pi(\cdot)$ is a random permutation. The permutation we implemented was the Fisher-Yates shuffle [42].

The permute protocol works as follows. After Bob generates a private and public key pair (sk,pk), he encrypts each of his random additive shares corresponding to the cross-correlation peaks and sends them to Alice. She generates a cryptographically secure random number s_k for each value from Bob and computes $b'_k = EN(b_k, pk) \times EN(s_k, pk)$ and $a'_k = EN(a_k, pk) \times EN(-s_k, pk)$. She then generates another cryptographically secure random number r and computes $a''_k = a'_k \times EN(r, pk)$. She generates a random permutation $\pi(\cdot)$ and computes $\overline{a} = \pi(a'')$ and $\overline{b} = \pi(b')$. Finally, she sends $\overline{a}$ and $\overline{b}$ to Bob. Notice that Bob now has access to $\tilde{z} = \pi(a) + \pi(b) + r$ in the plaintext for each song, but he cannot know any of the values of the cross-correlations because he does not know r and he does not know which index corresponds to which song in his database because of the permutation π. He can compute the maximum of these values and send the permuted index back to Alice. After she undoes the permutation, she now knows the index of her song in the database.

3.3.4 Step 4: Obtaining the Desired Information

In the final step, Alice finally retrieves the information about her song from Bob's database. If the database was public, she could just browse it until she found her music index. Figure 3.4 contains an illustration of this protocol. Since Bob wants to preserve the privacy of his database, except of course the information on Alice's song, they must exchange the information using Oblivious Transfer (OT) [85]. Consider l to be the index of Alice's song on the database and $\mathbf{u} = \{u_1, \ldots, u_K\}$ a vector representing the relevant information of the K songs in the database. If Alice encrypts her music index and sends it to Bob, he can just generate cryptographically secure random numbers r_k and compute:

$$v_k = EN(u_k, pk) \times (EN(l, pk) \times EN(-k, pk))^{r_k} \tag{3.11}$$

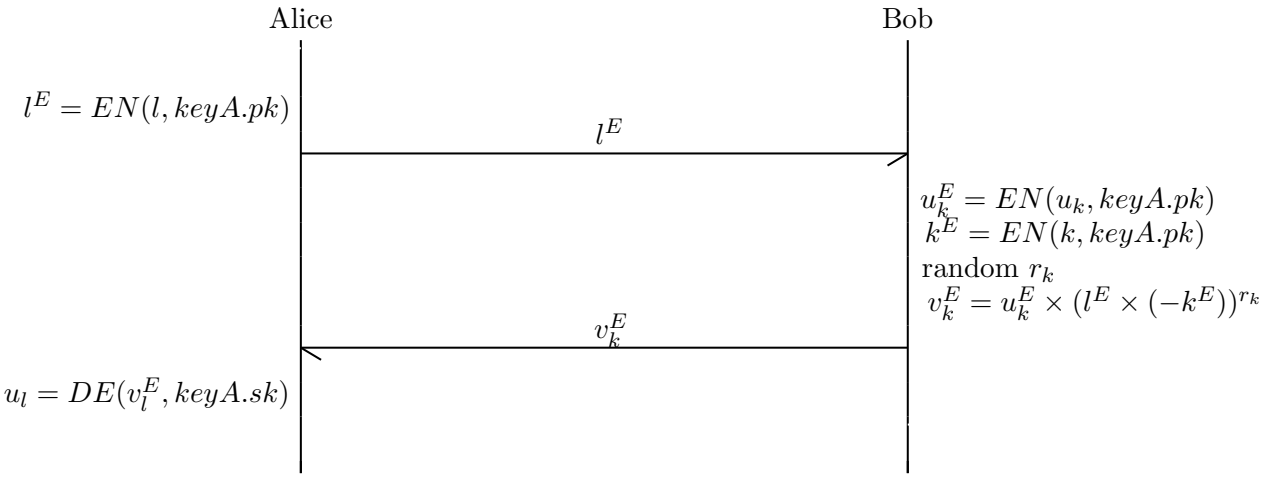


Figure 3.4: Message sequence chart for Step 4.

with $k = 1, \ldots, K$, which in the plaintext is equivalent to:

$$v_k = u_k + r_k(l - k) \tag{3.12}$$

He then sends them to Alice. Notice that the privacy of the database is preserved, because even if Alice decrypts every message she receives from Bob, all but the information corresponding to her song will appear as random numbers to her.

3.4 Results

In this section we describe the experiments we performed in order to obtain an implementation of a music matching algorithm. This implementation should be fast enough to be used as an on-line application without compromising the correctness of the music matching process. We focus on the following aspects: communication between the two parties, complexity and time consumption of the algorithms and correctness of the matching result.

The results presented in this section were obtained using a music database of $K = 50$ 16-bit WAVE files of different genres, including celtic, pop and metal, with an average duration of 3 minutes and 30 seconds. In order to reduce the size of the songs in terms of number

of samples, since it greatly affects the execution time of the algorithm, all the songs were downsampled to $F_s = 8000$Hz.

3.4.1 Communication Between the Two Parties

Regarding the communication exchanges between Alice and Bob, there is a major difference between the scenario where the music matching occurs without the use of privacy-preserving techniques and the scenario where cryptography is used. In the first scenario, communication between the parties occurs only at the start of the interaction, when Alice sends her music segment to Bob, and at the end of that interaction, when Bob returns the information corresponding to her song. As can be observed, the total amount of communication in this scenario is very small. In the second case, mainly due to the use of cryptography and random additive shares at each computation step, the need for communication is much larger. This scenario will now be analysed in more detail.

There are three classes of values being exchanged between Alice and Bob: control values used for exchanged messages synchronization, such as the size of Alice's music segment, etc., cryptographic keys, and the encrypted values themselves. Since the third class overwhelms the other two in terms of total amount of communication required, it is the only one which will be examined.

As in the non-private scenario, performing the music matching in a private way starts by Alice sending her music segment to Bob. The difference is that in this case every sample has to be encrypted. Let S be the number of samples of Alice's segment. She has to send $2N_{bits}S$ bits, instead of the original $16S$ bits. It is necessary to consider $2N_{bits}$ bits per message instead of just N_{bits} because the Paillier cryptosystem performs the following mapping: $p \in Z^*_{N_{bits}} \rightarrow c \in Z^*_{N^2_{bits}}$, $p-$plaintext message, $c-$ciphertext message.

For each cross-correlation Bob computes, he has to send a random additive share to Alice. The average length of the songs in Bob's database is much larger than the length of Alice's music segment, so he has to compute on average N_{cc} cross-correlations for each song, where

scenario	no privacy	with privacy
Step 1	$16S$	$2N_{bits}S + 2N_{bits}KN_{cc}$
Step 2	-	$2(2N_{bits}^2KN_{cc})$
Step 3	-	$3(2N_{bits}K)$
Step 4	N_{bits}	$2N_{bits}K$

Table 3.1: Summary of communication analysis, results presented in number of bits.

the value of N_{cc} will depend on the chosen window shift between cross-correlations. This means that at this stage he has to send a total of $2N_{bits}KN_{cc}$ bits to Alice. For obtaining the cross-correlation peaks, Alice has to send to Bob the encryption of each bit of each of her random additive shares for the cross-correlation, and Bob has to send back the greater-than result for each comparison he performs. This means that the total number of bits exchanged in this step is $2(2N_{bits}^2KN_{cc})$. For finding the most likely song index, there are three rounds of encrypted random additive shares exchanges between Alice and Bob. In this step, they exchange between themselves $3(2N_{bits}K)$ bits. Finally in the last step, Bob sends the information on his database to Alice. Assuming that the information of each song can be represented in a single number of N_{bits} bits, he sends $2N_{bits}K$ bits to Alice. The summary of the communication required between the two parties is presented in Table 3.1.

3.4.2 Computational Complexity

Another aspect where there is a significant difference between the two implementation approaches is the computational complexity, represented by the type and number of operations required for each algorithm. The following abbreviations are considered: CA for complex addition, CM for complex multiplication, MM for modular multiplication and ME for modular exponentiation. Notice that MM is the ciphertext equivalent of CA in the plaintext and that ME is the ciphertext equivalent of CM in the plaintext. Also, CA is composed of 2 additions and CM is composed of 2 additions and 4 multiplications. The computational complexity of the algorithms is summarized in Table 3.2.

scenario	no privacy	with privacy
cross-correlation	$T(1CM + 1CA)$	$T(4ME + 2MM)$
greater-than	1	$N(6MM + 4ME)$

Table 3.2: Summary of computational complexity analysis, results presented in number of operations.

T	cross-correlation	greater-than
8000	4-5	1-2
40000	18-20	1-2
80000	35-40	1-2

Table 3.3: Summary of execution times for privacy-preserving algorithms, results presented in seconds.

3.4.3 Execution Time

The analysis made in terms of execution time focuses on two main algorithms, which are the privacy preserving versions of the cross-correlation and greater-than algorithms. All the execution times presented were obtained running the algorithms on an Intel Core2 Quad CPU Q6600 @ 2.40GHz.

Three different values for the number of samples in Alice's music segment are considered, $T = 8000$, $T = 40000$ and $T = 80000$, in order to evaluate the obtained results for music samples of 1, 5 and 10 seconds, respectively. The execution time, in seconds, for each algorithm is presented in Table 3.3. For comparison, we also present the execution times, in seconds, for the non-private implementations of the same algorithms in Table 3.4. The results in Table 3.3 immediately raise some concerns. Considering the case of matching Alice's music segment with a single song in Bob's database, it is necessary to compute N_{cc} privacy-preserving cross-correlations and N_{cc} privacy-preserving greater-than comparisons.

T	cross-correlation	greater-than
8000	$\ll$0.01	$\sim$0
40000	$<$0.01	$\sim$0
80000	0.01	$\sim$0

Table 3.4: Summary of execution times for non-private algorithms, results presented in seconds.

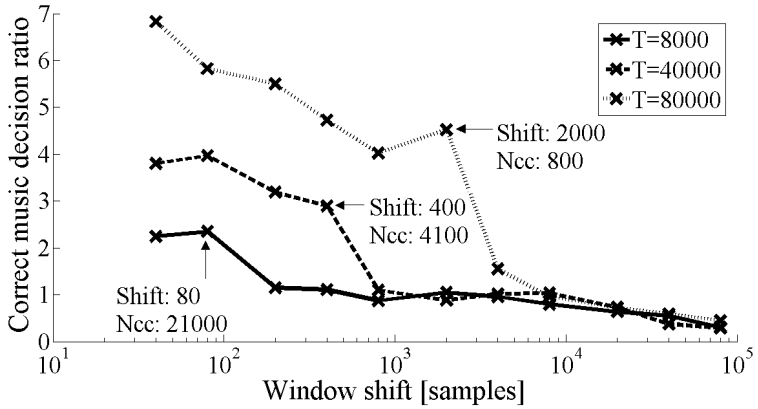


Figure 3.5: Music matching results as a function of window shift.

3.4.4 Correctness

The correctness of the matching results is also a very important aspect to take into account, as the whole music matching process is useless if it does not provide a correct result. In order to find an adequate value for the window shift between the cross-correlations we chose several random music segments of 1, 5 and 10 seconds from the songs in the database and used them to perform the query using different values for the window shift between cross-correlations in order to find out when a visible degradation of the results occurred. This was evaluated by analysing the ratio between the maximum of the cross-correlation for the correct song and the maximum of the cross-correlation for all the remaining songs. A ratio lower than 1 means a wrong identification. This ratio was computed over a wide range of values for the window shift, and the results we obtained are presented in Figure 3.5. By analysing the results, we can see that adequate values for the window shift between cross-correlations are 80, 400 and 2000 samples for $T = 8000$, $T = 40000$ and $T = 80000$, respectively. We can now compute the average number of cross-correlations we have to compute for each song in the database. These results are presented in Table 3.5. As expected, for larger music segments, one can compute much fewer cross-correlations.

T	window shift [samples]	N_{cc}
8000	80	21000
40000	400	4100
80000	2000	800

Table 3.5: Number of cross-correlations as a function of window shift.

3.5 Privacy Analysis and Other Practical Issues

In this section we analyse the privacy properties of the PPMM algorithm. We start by pointing out a security flaw in the original formulation, then we describe how it can be exploited, and finally we present a set of possible solutions for correcting this flaw, each providing different levels of privacy and performance.

3.5.1 Attack on the Privacy of the Algorithm

Until now it has been assumed that, since each of the component primitives are provably secure, the overall algorithm must be secure as well. Alice and Bob only exchange encrypted data, and rely on secure two party computations to achieve their goals. This should suffice for them to have the guarantee that their privacy is preserved. However, this is a fallacious assumption. The problem resides neither in the homomorphic cryptosystem nor in the privacy properties guaranteed by any of the secure two party computations, but in the fact that the secure computation chosen to perform Step 3, described in Section 3.3.3, is not adequate for solving the problem at hand. The problem arises due to two factors: a) the same operations are repeated on all songs in Bob's catalogue, and b) Bob can validly assume that in the case of a match, Alice's snippet will be very similar to a corresponding segment from one of his songs. The correlation between any other segment and the matching segment will not be significantly different from its correlation to Alice's snippet.

In Step 2, described in Section 3.3.2, Alice and Bob obtain additive shares a_i and b_i of the peak correlation $z_i = a_i + b_i$ of the i^{th} song in Bob's data. After Step 3, Bob possesses permuted values $\{\tilde{z}_i\} = \{\pi(z_i)\} + r$; the permutation $\pi(\cdot)$ and noise r are intended to hide the data from Bob. However, Bob can compute differences over all pairs of $\tilde{z}$'s, *i.e.*, $\tilde{z}_2 - \tilde{z}_1 = \pi(z_2) - \pi(z_1)$,

$\tilde{z}_3 - \tilde{z}_2 = \pi(z_3) - \pi(z_2)$, etc., thus removing the random number r. The complexity of this process is $\mathcal{O}(K^2)$. Next Bob computes z_{uvw}, which is the peak cross-correlation between the segment u of song v and the song w, for all u, v, and w with computational complexity $\mathcal{O}(\eta^2 K^2)$, where η is the average number of segments of the same length as Alice's snippet in a song. He then tries iteratively to find u_1, v_1, w_1 and u_2, v_2, w_2 such that $z_{u_1 v_1 w_1} - z_{u_2 v_2 w_2} \approx \pi(z_i) - \pi(z_j)$, for all i, j, with complexity $\mathcal{O}(\eta K^2)$. Once Bob completes this, he can reverse the permutation and find the index of Alice's song in his database. In practice, he can potentially speed this up by several orders of magnitude using branch-and-bound algorithms. Additionally, since Bob has access to all $\tilde{z}$'s at the same time, he does not require an exact match between Alice's snippet and any song. He can unravel her information in polynomial time, even if her snippet is somewhat corrupted by noise.

3.5.2 Securing the Algorithm

One way of preventing Alice from leaking information regarding her song is to make a more extensive use of the secure greater-than comparison, thus preventing Bob from having access to Alice's data in plaintext. We analyse three different ways to tackle this problem, each providing different levels of privacy and performance in terms of execution time: the first aims at optimum performance, the second aims at optimum privacy preservation and the third uses the best characteristics of the other two. We also analyse the execution time of each approach in comparison with T_{Step2}, which is the total execution time of Step 2. This step is also the bottleneck of the original algorithm in terms of execution time.

Maximal Performance Approach

The first approach uses only the minimum amount of comparisons that one must make in order to find the maximum value from a set of K numbers, which is $K-1$. The time required to do this is $T^1_{\text{Step3}} = \frac{1}{\eta} \cdot T_{\text{Step2}}$, since now Bob only needs to compute one secure greater-than comparison for each song in his database. However, if he computes only this minimum amount, after the last comparison he knows that Alice's song corresponds to one of the two

indexes used in the last comparison, which means that he has a 1/2 probability of making a correct guess. This approach preserves almost no privacy on Alice's side, and therefore it is not suitable for our purpose either.

Maximal Privacy Preservation Approach

The second approach, which provides the maximum privacy possible, requires all cross-correlation values to be compared with each other, implying a total of $\frac{K(K-1)}{2}$ comparisons. If Bob does this, he will have no idea of which song Alice has, but he would require $T^2_{\text{Step3}} = \frac{K}{2\eta} \cdot T_{\text{Step2}}$, which is much larger than T_{Step2} since $K >> \eta$. Using this approach leads to an unacceptable increase in the execution time of the overall algorithm when compared with its original formulation, and therefore it is not suitable for our purpose.

Compromise Approach Between Performance and Privacy

From the two previous approaches we realize that it is possible for Alice to get the index she wants from Bob's database either in a completely privacy preserving manner or a time efficient way, but not both at once. This leads us to devise a compromise approach that only leaks partial information, so that Bob cannot learn anything he is not supposed to, while also not turning this step of the algorithm into a bottleneck. The approach we developed achieves this using $\frac{K}{2} \log K$ secure greater-than comparisons, which means that its execution time is $T^3_{\text{Step3}} \approx \frac{\log K}{2\eta} \cdot T_{\text{Step2}}$, and therefore smaller than T_{Step2} for reasonable values of K and η. The basic idea is to perform $\log K$ rounds of comparisons with $\frac{K}{2}$ comparisons each. We will show how we obtained these numbers.

3.5.3 Analysis of the Compromise Approach

Example with $K = 8 = 2^3$

We start with an example of how our approach works for $K = 8$. Consider the cross-correlation values z_i, $i = 1, ..., 8$, such that $z_1 > z_2 > ... > z_8$. Any other combination of

greater-than orderings between these numbers does not affect our analysis, which means that any conclusions we take from this specific ordering will also hold for all the other possible orderings.

The basic idea behind our approach is to iteratively perform greater-than comparisons between the maxima of two adequately selected comparisons from the previous round and greater-than comparisons between the minima of the same two comparisons from the previous round. For the first round of comparisons we use the original sequence split into pairs. The sequence of comparisons one needs to make is as follows:

$$1^{\text{st}} : (z_1, z_2)\boxed{1}, (z_3, z_4)\boxed{2}, (z_5, z_6)\boxed{3}, (z_7, z_8)\boxed{4}$$
$$2^{\text{nd}} : (z_1, z_5)\boxed{5}, (z_2, z_6)\boxed{6}, (z_3, z_7)\boxed{7}, (z_4, z_8)\boxed{8}$$
$$3^{\text{rd}} : (z_1, z_3)\boxed{9}, (z_5, z_7)\boxed{10}, (z_2, z_4)\boxed{11}, (z_6, z_8)\boxed{12}$$

For any possible greater-than ordering of cross-correlation values, the previous sequence of comparisons generalizes to:

$$1^{\text{st}} : (z_1, z_2)\boxed{1}, (z_3, z_4)\boxed{2}, (z_5, z_6)\boxed{3}, (z_7, z_8)\boxed{4}$$
$$2^{\text{nd}} : (\max\boxed{1}, \max\boxed{3})\boxed{5}, (\min\boxed{1}, \min\boxed{3})\boxed{6}, (\max\boxed{2}, \max\boxed{4})\boxed{7}, (\min\boxed{2}, \min\boxed{4})\boxed{8}$$
$$3^{\text{rd}} : (\max\boxed{5}, \max\boxed{7})\boxed{9}, (\min\boxed{5}, \min\boxed{7})\boxed{10}, (\max\boxed{6}, \max\boxed{8})\boxed{11}, (\min\boxed{6}, \min\boxed{8})\boxed{12}$$

Notice that in each step one can start by comparing either the maxima or the minima of the previous round of comparisons. Getting back to our example, using the information learned

from the previous round, at the beginning of each round Bob knows:

$$1^{\text{st}} : -$$

$$2^{\text{nd}} : \alpha_1 \leftarrow ((z_1 > z_2 \text{ and } z_5 > z_6) \text{ and } (z_3 > z_4 \text{ and } z_7 > z_8)) \text{ or}$$

$$\beta_1 \leftarrow ((z_2 > z_1 \text{ and } z_6 > z_5) \text{ and } (z_4 > z_3 \text{ and } z_8 > z_7))$$

$$3^{\text{rd}} : \gamma_1 \leftarrow ((z_1 > z_5 \text{ and } z_3 > z_7) \text{ and } (z_2 > z_6 \text{ and } z_4 > z_8)) \text{ or}$$

$$\delta_1 \leftarrow ((z_5 > z_1 \text{ and } z_7 > z_3) \text{ and } (z_6 > z_2 \text{ and } z_8 > z_4))$$

He can try to guess which song Alice has, by assuming any of the 4 possible combinations of partial greater-than relationships between the cross-correlation values. For each combination Bob obtains:

$$\alpha_1 + \gamma_1 : z_1 >^{z_2}_{z_5}> z_6, z_3 >^{z_4}_{z_7}> z_8 \longrightarrow z_1 \text{ or } z_3$$

$$\alpha_1 + \delta_1 : z_5 >^{z_6}_{z_1}> z_2, z_7 >^{z_8}_{z_3}> z_4 \longrightarrow z_5 \text{ or } z_7$$

$$\beta_1 + \gamma_1 : z_2 >^{z_1}_{z_6}> z_5, z_4 >^{z_3}_{z_8}> z_7 \longrightarrow z_2 \text{ or } z_4$$

$$\beta_1 + \delta_1 : z_6 >^{z_5}_{z_2}> z_1, z_8 >^{z_7}_{z_4}> z_3 \longrightarrow z_6 \text{ or } z_8$$

After the final round Bob does not learn anything new because he does not get any more greater-than comparison requests from Alice. Also, because of the ambiguity introduced by letting Alice decide whether Bob starts the next round of comparisons using the maxima or the minima from the previous round, he cannot make a correct guess on what song Alice has with probability P more than $1/8$, even with partial information he has regarding the correct greater-than ordering of the z_i's. Generalizing this to any $K = M = 2^m$, Bob cannot correctly guess Alice's song with $P > 1/K$.

Generalization for Any K Songs

Finally, all it remains is to study our approach for the most general case, where K can be any positive integer. Considering for instance the limit case of $K = 5 = 2^2 + 1$ and any possible ordering of the z_i's, the sequence of comparisons requested by Alice would be:

$$1^{\text{st}} : (z_1, z_2)\boxed{1}, (z_3, -)\boxed{2}, (z_5, -)\boxed{3}, (z_7, -)\boxed{4}$$

$$2^{\text{nd}} : (\max\boxed{1}, z_5)\boxed{5}, (\min\boxed{1}, z_5)\boxed{6}, (z_3, z_7)\boxed{7}, (z_3, z_7)\boxed{8}$$

$$3^{\text{rd}} : (\max\boxed{5}, \max\boxed{7})\boxed{9}, (\min\boxed{5}, \min\boxed{7})\boxed{10}, (\max\boxed{6}, \max\boxed{8})\boxed{11}, (\min\boxed{6}, \min\boxed{8})\boxed{12}$$

In this case, using the information learned from the previous round, at the beginning of each round of comparisons Bob learns:

$$\begin{aligned}
&1^{\text{st}} : - \\
&2^{\text{nd}} : \alpha_2 \leftarrow ((z_1 > z_2 \text{ and } 1) \text{ and } (1 \text{ and } 1)) \text{ or} \\
&\qquad \beta_2 \leftarrow ((z_2 > z_1 \text{ and } 1) \text{ and } (1 \text{ and } 1)) \\
&3^{\text{rd}} : \gamma_2 \leftarrow ((z_1 > z_5 \text{ and } z_3 > z_7) \text{ and } (z_2 > z_5 \text{ and } z_3 > z_7)) \text{ or} \\
&\qquad \delta_2 \leftarrow ((z_5 > z_1 \text{ and } z_7 > z_3) \text{ and } (z_5 > z_2 \text{ and } z_7 > z_3))
\end{aligned}$$

As before, Bob can assume any of the 4 possible combinations:

$$\begin{aligned}
&\alpha_2 + \gamma_2 : z_1 > z_2 > z_5, z_3 > z_7 \longrightarrow z_1 \text{ or } z_3 \\
&\alpha_2 + \delta_2 : z_5 > z_1 > z_2, z_7 > z_3 \longrightarrow z_5 \text{ or } z_7 \\
&\beta_2 + \gamma_2 : z_2 > z_1 > z_5, z_3 > z_7 \longrightarrow z_2 \text{ or } z_3 \\
&\beta_2 + \delta_2 : z_5 > z_2 > z_1, z_7 > z_3 \longrightarrow z_5 \text{ or } z_7
\end{aligned}$$

Notice that in this case Bob also has no idea of what song Alice has, but because K is not a power of 2, the probabilities of him guessing correctly depend on actual index of the

right song. This means that, in the current scenario, the probability of guessing Alice's song correctly is 1/4 if it has the index 3, 5 or 7 and it is 1/8 if it has the index 1 or 2. Like before, this scenario can be easily expanded to any value of K.

3.6 Summary

This chapter presented an implementation of a PPMM algorithm, showing how privacy is achieved at the cost of computational complexity and execution time. We presented not only implementation details but also an analysis of the obtained results in terms of communication between the two parties, computational complexity, execution time and correctness of the matching algorithm. Furthermore, we detected a privacy flaw in the original design of the algorithm, for which we described a possible solution, analysing the resulting tradeoff between privacy and computational complexity. Although in this chapter we focused on a music matching application, the principles can be easily adapted to perform other tasks, such as Keyword Spotting (KWS) or Speaker Verification (SV).

4 Privacy-Preserving Query-by-Example Speech Search

This chapter contains the work performed on the Privacy-Preserving Query-by-Example Speech Search (PPQESS) task. Section 4.2 presents an overview on the Query-by-Example Speech Search (QESS) problem. Section 4.3 contains the details on the experimental setup and Section 4.4 illustrates our approach with some experimental results. Finally, Section 4.5 shows an analysis on the level of privacy achieved by our approach.

4.1 Introduction

With the development of data recording and storage capabilities, it has become possible for many entities, both public and private, at the individual or corporate level, to store thousands of hours of speech audio for a relatively low cost. Such speech collections may be composed of emergency calls or witnesses testimonies in the case of law enforcement agencies, business calls between service providers and their clients, doctors notes taken during medical appointments, examinations or surgeries, etc. Public entities and large private companies usually store these recordings locally in their own servers, but an individual person or small companies might prefer to store them online. Online storage is also the preferred choice when only limited local storage capacities are available or when the owners of such recordings do not possess the know-out required to extract relevant information from them. Since the information contained by these databases allows for trend analysis and estimation, quality control and several other forms of service improvement, it is of vital importance to mine it. However, for these mining tasks to be performed, it is necessary to have full access to the audio of all recordings and, in particular, to know what is being said by which party.

A typical approach is to have a professional transcriber write down everything that was said or

to run an Automatic Speech Recognition (ASR) system over the database. A simpler, more robust and less expensive approach that is often employed is to simply search for specific terms or phrases that relate to specific topics of interest – inferences may be derived simply from the frequency of occurrence of these patterns. One version of this approach that is particularly relevant is the QESS approach, in which the key phrase patterns to be searched for are specified through actual examples.

In all scenarios, however, privacy concerns arise. The data being mined frequently contain private data. The identity of the speakers are generally considered private and must not be revealed. Speakers also often reveal private information, such as account or social security numbers, demographic information, health information, etc., all of which are clearly private and must not be publicly accessible. The only parties that should be granted access to the recordings are the people whose voice is recorded and the entities who are legally authorized to such access, but only *in the context* these recordings are intended to be heard. Any access to this information outside of this setting is clearly unacceptable. In all scenarios, the identity of the people engaged in the conversations must be kept hidden, particularly the people who are not employed or similarly related to the entity storing the recordings.

4.2 Query-by-Example Speech Search

One of the main approaches to mining large amounts of speech data is searching it for pre-specified keywords or phrases. In many situations, it is useful, or even necessary, to specify these key terms through *spoken* examples, rather than through other means of representation. In this "query-by-example" scenario, instances of these key terms in the test data to be mined are discovered through their acoustic match to the provided spoken examples. QESS is a particularly relevant problem for low-resource languages, and has recently gained significant research interest partially due to the success of the Spoken Web Search (SWS) task in the MediaEval evaluation series [81, 6]. In practice, the query-by-example task can be considered as a sort of generalization of the problem of speech search based on text queries. When the specific queries are known in advance of searching through a speech corpus, Keyword Spotting

(KWS) approaches can be applied. On the other hand, when the data collection has to be processed without prior knowledge of the queries, the so-called Spoken Term Detection (STD) task, a more elaborate procedure comprising a first indexing stage and a second search stage [74, 82] must be employed. The STD task has received considerable research attention in the recent past, partially due to the series of evaluations organized by the National Institute of Standards and Technology (NIST) [59, 94, 95].

One common limitation of KWS and STD is that they are language-dependent. In both cases, conventional approaches rely somehow on a well-trained ASR system or on a set of phonetic models trained for the particular language of the speech collection. In well-resourced languages where such resources are available, even the query-by-example problem can be converted to KWS or STD: a simple straightforward approach would consist of an initial speech-to-text conversion of the query, followed by application of any of the methods used in text-query-based speech search. However, QESS is more appropriate in situations where specific acoustic or phonetic models may not be assumed for the language, such as in low-resource situations. It is also relevant in other situations, *e.g.* surveillance tasks or code-switched situations, where one may be looking for specific terms which are not well transcribed in the expected language of the corpus. In the particular case of QESS, some of the most recent approaches are based on template matching methods, such as different flavours of Dynamic Time Warping (DTW) of posterior derived features [55, 119]. As an alternative to the widespread template matching approaches, other systems use Acoustic Keyword Spotting (AKWS) [130, 2], exploiting in several ways acoustic models in multiple languages. A review of these and other methods can be found in [81, 6]. The baseline QESS systems considered in this work are based on DTW template-matching of posterior features provided by different language-dependent acoustic networks [55, 119].

4.3 Query-by-Example Speech Search using SBE

The application of the Secure Binary Embedding (SBE) to QESS is straightforward: instead of evaluating the DTW algorithm using a distance metric appropriate for original arrays of

features, the normalized Hamming distance between the corresponding SBE hashes is used, therefore hiding any relevant information from the party computing the algorithm.

The implementation of a PPQESS system is as follows: a party that possesses a collection of audio recordings which contain sensitive or otherwise private information extracts features from the audio, computes the corresponding SBE hashes, and stores them in a free, public-access datacentre. Later, when it is necessary to recover recordings containing a specific key phrase, that party records one (or many) utterances of that key phrase, extracts features from it, computes the corresponding SBE hashes with the same parameters $\mathbf{A}$, $\mathbf{w}$ and $\boldsymbol{\Delta}$ that were used before and performs the DTW algorithm over the entire collection.

A potential scenario where a party other than the user that wants to query the database should also be considered. Here, two possibilities exist: either the user trusts this party and grants him access to his feature extraction tools and SBE parameters, allowing him to query the database in a similar manner, or he does not. The second possibility is similar to considering this new party as a potential attacker, and we will address this in Section 4.5.

For all scenarios, we consider the server to be *honest-but-curious*, which means it can analyse the hashes it receives from the users and try to learn information regarding the original data from them. No attack model is considered, as the server cannot request any information from the users.

4.3.1 DTW based Detector

DTW [84] is a time series alignment algorithm developed originally for speech recognition, which aims to align two sequences of feature vectors by warping the time axis iteratively until an optimal match between the two sequences is found. The two sequences can be arranged on the sides of the grid, one on the top and the other on the left hand side, as it is shown in Figure 4.1. Inside each cell a distance metric is computed, comparing the corresponding elements of the two sequences. In order to find the best alignment between these two sequences, it is necessary to find a path through the grid which minimizes the total distance between them.

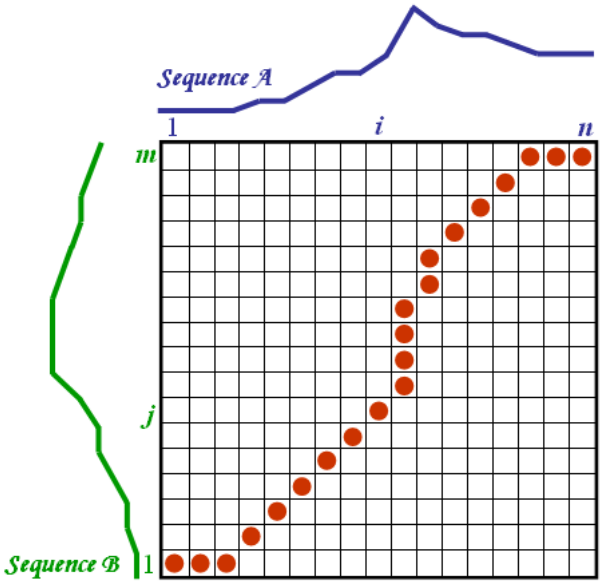


Figure 4.1: Example of a possible alignment of two sequences using the DTW algorithm.

This involves finding all possible paths through the grid and for each of them compute the overall accumulated distance. Since different paths may have different lengths, a weighting function is usually used to normalize them. A common weighting function is the length of the path being analysed. In order to force the optimal path to have an acceptable shape, a number of optimizations and constraints are usually used, such as monotonicity, continuity, maximum/minimum slope steepness, etc. In our implementation of the DTW algorithm, we just consider paths that advance only one cell at a time, either to the right, diagonally or up, thus enforcing monotonicity and continuity. Instead of using a sliding window approach for spotting a specific query in a collection of long sequences, we allow for the alignment between the query sequence and the long sequence to start and end at any arbitrary position of the long sequence. To do this, we impose some conditions to the computation of the best path length-normalized accumulated distances, and we store all possible paths and the corresponding accumulated distance matrices. When the last cell is analysed, we perform a backward step to find the minimum distance crossing path and the start and end frame of the query-term searched [119]. In this work, we produce a single score per each query-file pair corresponding to the negative length-normalized accumulated distance of the best crossing

path match, given by

$$\text{Score}(KW, TS, p) = \min_{p \in \mathcal{P}} \sum_{p=1}^{P} \frac{d(KW, TS, p)}{\text{length}(p)} \tag{4.1}$$

where KW are the frames corresponding to the target keyword, TS are frames corresponding to the target sentence, p is a possible path for aligning KW and TS, and $d(KW, TS, p)$ is the distance between KW and TS along path p. Thus, the speech search task considered here is more similar to spoken document retrieval than to a simple query detection. Finally, we also apply a per-query zero-mean and unit-variance normalization (q-norm) to the detection scores.

4.4 Results

In this section we present the results obtained on the QESS task using a DTW approach, both for the normal scenario and the privacy-preserving one. For this task, all the results are presented in terms of Maximum Term Weighted Value (MTWV) and Detection Error Tradeoff (DET) curves, as they are commonly used in the NIST STD evaluations [59]. We consider a prior that approximately fits the empirical prior ($P_{target} = 0.01$), and two more suitable false alarm and miss error costs to our application scenario ($C_{fa} = 1$ and $C_{miss} = 100$). The corresponding β is 0.99.

4.4.1 Feature Extraction

The feature extraction process is based on language-dependent phonetic networks that obtain posterior features exploiting Multi-Layer Perceptron (MLP) networks that are part of our in-house hybrid connectionist ASR systems for European Portuguese (pt-PT), Brazilian Portuguese (pt-BR), European Spanish (es-ES) and American English (en-US). The phonetic class posterior probabilities are in fact the result of the combination of four MLP outputs trained with Perceptive Linear Predictive (PLP) features (13 static + first derivative), PLP with log-RelAtive SpecTrAl (RASTA) speech processing features (13 static + first derivative),

ajuda	enviar	menu	saudação
alterar	fim	operador	seguinte
anterior	gravar	ouvir	stop
apagar	guardar	programar	telefonar
cancelar	ligar	rechamar	tocar
conferência	lista	repetir	transferir
continuar	marcar	sair	

Table 4.1: List of selected application words.

Modulation SpectroGram (MSG) features (28 static) and Advanced Font-End features from the European Telecommunications Standards Institute (ETSI) (13 static + first and second derivatives). The language-dependent MLP networks were trained using different amounts of annotated data [1]. For the pt-PT acoustic models, 57 hours of Broadcast News (BN) downsampled data and 58 hours of mixed fixed-telephone and mobile-telephone data were used. The pt-BR models were trained with around 13 hours of BN downsampled data. The es-ES networks used 36 hours of BN downsampled data and 21 hours of fixed-telephone data. The en-US system was trained with the HUB–4 96 [52] and HUB–4 97 [41] downsampled data sets, which contain around 142 hours of television and radio broadcast data. Each MLP network is characterized by the size of its input layer that depends on the particular parameterization and the frame context size (13 for PLP, PLP-RASTA and ETSI; 15 for MSG), the number of units of the two hidden layers (500), and the size of the output layer. In this case, only monophone units are modelled, resulting in MLP networks of 39 (38 phonemes + 1 silence) soft-max outputs in the case of pt-PT. Finally, low-energy frames detected with the alignment generated by a simple bi-Gaussian model of the log energy distribution computed for each speech segment is removed.

4.4.2 Experiments using Posteriors

We ran experiments on a sub-set of the Portuguese SpeechDat(II) corpus [56, 38]. More details on this corpus are presented in Section A.1. We selected 27 different application words, shown in Table 4.1, from the SpeechDat application words list. The evaluation set consists of 481 utterances from one of the data categories of the Portuguese SpeechDat(II)

	pt-PT posteriors
cosine	0.812
Euclidean	0.800

Table 4.2: Query-by-example speech search results, given in terms of MTWV, when using pt-PT posterior features.

corpus consisting of word spotting phrases using embedded application words. Most of the utterances contain only one application word (83.8%), but some utterances have zero (1.5%), two (12.0%) or three application words (2.7%). Special care was taken in selecting both the files containing the application words (training set) and the ones in the evaluation set, as to prevent any given speaker to be present in both sets. For each audio file, we extracted four sets of features: posteriors for the pt-PT, pt-BR, es-ES and en-US languages. The posteriors were extracted according to the procedure described in Section 4.4.1.

In our baseline experiments, without using SBE hashes, we evaluated the pt-PT posterior features, as they are the ones that best match the language present on the corpus. The results we obtained are presented in Table 4.2. Each row contains the results obtained when two different distance metrics are considered when performing the DTW algorithm. The cosine distance was used because it is the one that best adapts to the posterior features. The Euclidean distance was used because it is the one that best adapts to the SBE hashing scheme. We also performed experiments using the posteriors for pt-BR, es-ES and en-US, as well as a combination of all posteriors for different languages at the distance matrices level, as this is known to improve results in some situations [136]. However, none of these experiments lead to improvements on the results.

4.4.3 Experiments using SBE Hashes

For the experiments on PPQESS, we took the features used in the previous experiments and computed their corresponding SBE hashes using the method described in Section 2.2.2. The decision threshold for the value of the normalized Hamming distance under which we consider that leakage occurs was empirically set at 0.475. It was verified that for the values of bits per coefficient (*bpc*) considered in our experiments, if the Euclidean distance between the original

leakage	∼ 5%	∼ 25%	∼ 50%	∼ 75%	∼ 95%
bpc=4	–	0.731	0.798	0.790	0.787
bpc=8	–	0.784	0.744	0.794	0.784
bpc=16	–	**0.800**	0.799	0.798	0.797

Table 4.3: Query-by-example speech search results, given in terms of MTWV, when using SBE hashes of pt-PT posterior features.

features was above Δ, then the corresponding normalized Hamming distance between the hashes was above that threshold. As mentioned before, the amount of leakage is exclusively controlled by Δ. Since the best baseline results were obtained for the posteriors for pt-PT, we only analysed them in the following experiments. The results obtained are presented in Table 4.3. As expected, increasing the value of Δ (and therefore increasing the amount of hashes that reveal information regarding the Euclidean distance between the original feature vectors) leads to better keyword detection results, specially when lower values of *bpc* are considered. Surprisingly, increasing the value of *bpc* does not lead to better keyword detection results. A possible reason for this may be that there is a clear separation between most of the utterances with and without a target keyword, which would mean that they are only slightly affected by the noise introduced when computing the SBE hashes. An interesting result is that the observed degradation when the SBE hashes are considered comes almost exclusively from the fact the SBE hashes relate to the Euclidean distance between the original vectors, and therefore they are very promising for obtaining privacy-preserving techniques for other speech processing tasks. For the sake of completion, we present in Figure 4.2 the DET curves for the baselines using the cosine and Euclidean distances, as well as the best results using SBE hashes (*bpc*=16, 25% leakage).

4.5 Privacy Analysis and Other Practical Issues

We recall the requirements from Section 4.1: the system must not be able to recover either the speaker's speech or unauthorized information from the data received from the speaker. Sensitive information includes not only what was actually said by the speaker (either whole sentences or individual words) but also information about gender, age, nationality (through

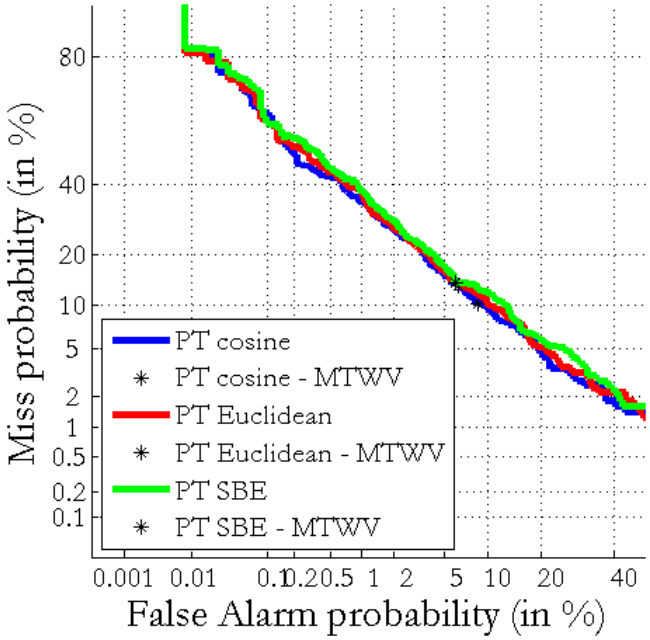


Figure 4.2: DET curves for the baseline and best SBE hashes results.

both language and accent) or even, in an extreme situation, speaker-specific voice fingerprints that can be used to impersonate him elsewhere. Here, however, the main concern is with keeping the actual speech content private, although some hints regarding other information could also be extracted from the features.

Due to their nature, SBE hashes provide a basic but strong form of security: a vector $\mathbf{x}$ cannot be recovered, even in part, from its corresponding SBE hash $\mathbf{q}(\mathbf{x})$, if the projection matrix $\mathbf{A}$ and dither vector $\mathbf{w}$ are unknown. If matrix $\mathbf{A}$ and vector $\mathbf{w}$ are adequately generated, then the hashes themselves provide information-theoretic security [18].

As mentioned in Section 4.3, in our setup for a PPQESS system, a user only sends SBE hashes computed from features extracted from his recordings to the remote storage facility. The user never needs to reveal matrix $\mathbf{A}$ and vector $\mathbf{w}$ to anyone, thus making it impossible for a malicious server to break the SBE hashing scheme. This means that no information regarding the user's voice or what he has said can be obtained by a malicious server.

If a malicious hacker is able to gain access to the user's device where the SBE private parameters are stored, then he might try and reverse the hashing process. However, even though

there is no formal proof regarding the non-invertibility of SBE hashes, the quantization function $Q(\cdot)$ makes it difficult to perform such a computation. The hacker might then be tempted to record words of interest, extract features from the audio, compute his own set of hashes and then try to match them to the hashes stored by the user, in order to find some close matches. This too seems intractable to perform, as he would need to replicate almost exactly the original feature generation process in order to obtain comparable feature values.

4.6 Summary

This chapter contained the details on how to build a PPQESS system. The proposed method allows performing spoken query search in an encrypted domain, by analysing ciphered information computed from the original recordings. Unlike other hashing techniques, SBE allows for the computation of the distance between vectors that are close enough, but are not perfect matches. We showed how these hashes can be combined with DTW based on posterior derived features to perform secure speech search. Experiments performed on a sub-set of the SpeechDat Portuguese corpus showed that the proposed privacy-preserving system obtains similar results to its non-private counterpart. An analysis of the level of data privacy achieved by our approach was also presented.

5 Privacy-Preserving Speaker Verification using SBE

This chapter covers the work performed on the Privacy-Preserving Speaker Verification (PPSV) task using Secure Binary Embedding (SBE) hashes. Section 5.2 contains an overview on different Speaker Verification (SV) techniques. Section 5.3 describes the details of the experimental setup, followed by Section 5.4, which presents some experimental results. Finally, Section 5.5 shows an analysis on the level of privacy achieved by our approach.

5.1 Introduction

Voice-based authentication systems, also often called SV systems, have significant privacy concerns. Current systems require access to recordings of a user's voice. A malicious system, or a hacker who has compromised the system, could edit the recordings to impersonate the user. Even if the user only transmits features extracted from the voice, such that a recording cannot be synthesized from them, other risks remain. Information about the user's identity, gender, nationality, etc., could be deduced even from parameterized signals, and potentially abused. Also, there is scope for direct privacy violation. In order to authenticate a user, the system must retain a model for the user, which it can compare to incoming recordings. These models may now be used to uncover other recordings by the user, such as on services like YouTube, where the user may have assumed anonymity.

Our objective is to enable SV in a manner that avoids risk to a user's privacy. To guarantee the privacy of the user, we require that the system should neither have access to the user's recordings, nor possess a model of the user's speech. These almost-paradoxical sounding requirements for a voice-based biometric system are not, in fact, unreasonable, and are assumed for other forms of secure biometrics as well [3]. The above requirements address our goals

for the system; in addition we also desire that the system in turn must not be vulnerable to imposters who may gain access to a client device such as a smartphone that the user employs to connect to it. We will assume that all communication between the user and the system is over an appropriately secured channel, and that we need not specifically consider either the man-in-the-middle or replay attacks.

Privacy concerns in voice biometric systems have been largely ignored until recently, and literature on the topic remains minimal. Previous work includes treating this task as an Secure Two-Party Computation (STPC) problem [98] or address it using an Locality-Sensitive Hashing (LSH) based approach [99]. The first approach employs Homomorphic Encryption (HE) methods to ensure that the system only sees encrypted data from the user, and only stores encrypted models that it cannot decrypt by itself, thereby satisfying privacy requirements. However, the computational overhead of repeated encryption and decryption makes this solution impractical. Moreover, this approach retains vulnerabilities to certain types of imposters [100]. The second approach converts voice recordings into password-like strings. Authentication is performed by matching the strings to stored templates. This method is fast and secure, satisfying both requirements mentioned above. However, it compromises accuracy by requiring an exact match between hashes derived from the user's speech and those in the models stored by the system.

5.2 Speaker Verification

In conventional text-independent SV systems, the user provides the system with voice samples during an enrolment phase and the system employs these samples to build a model for the user. Later, in the verification phase, new incoming speech signals are compared to this model to verify the user identity, usually generating a log-likelihood verification score.

In terms of speech parameterization, SV systems are typically built upon conventional short-term cepstral features, like Mel-Frequency Cepstral Coefficients (MFCC) or Perceptive Linear Predictive (PLP) features. As in other speech applications, feature vectors are usually aug-

mented with their derivatives. As for speaker modelling, the most popular approach is to obtain a Gaussian Mixture Model (GMM) for each target speaker based on Maximum A Posteriori (MAP) adaptation of the Gaussian means of a Universal Background Model (UBM), known as GMM-UBM technique [114]. The UBM is a GMM trained with several hours of speech representing the general distribution of speech characteristics from any speaker. We will refer to the speakers considered for training the UBM as imposters. In addition to reducing enrolment data requirements, MAP adaptation also ensures a one-to-one correspondence between the Gaussians in the UBM and those in the model for the speaker. Given a new recording purported to be from a target speaker, the log-likelihood assigned to it by the GMM for the target speaker is compared to that obtained from the UBM to determine if the speaker must be accepted or not [114].

More recently, variations of this GMM-UBM scheme have resulted in the proliferation of new successful vector-based methods, such as the GMM Supervector (GSV) [26] approach and the Joint Factor Analysis (JFA) [62] or Total Variability (TV) [33] based compensation methods. On one hand, the method generally known as GSV consists of mapping each speech utterance to a high-dimensional vector space. A Support Vector Machine (SVM) is then used for classification of speaker vectors within this space. The mapping to the high-dimensional space is achieved by stacking all parameters (usually the means) of the adapted speaker GMM (obtained in the same way as in the GMM-UBM scheme) in a single supervector. To verify that a given test recording was indeed spoken by the speaker, the supervector derived from a new recording is classified by the SVM. On the other hand, TV modelling [33] has rapidly emerged as one of the most powerful approaches for SV and has become the current de-facto standard. In this approach, closely related to the JFA [62], the speaker and the channel variabilities of the high-dimensional supervector are jointly modelled as a single low-rank TV space. The low-dimensionality TV factors extracted from a given speech segment form a vector, named i-vector, which represents the speech segment in a very compact and efficient way. Since the i-vector comprises both speaker and channel variabilities, in the i-vector framework for SV some sort of channel compensation or channel modelling technique usually follows the i-vector extraction process. Regarding channel compensation, Linear Discriminant

Analysis (LDA) or Within-Class Covariance Normalization (WCCN) are typically applied to compensate for channel nuisance in the i-vector space [32]. The verification score can then be obtained either based on a simple cosine similarity between the target speaker i-vector and the test utterance i-vector, or by evaluating the test utterance i-vector with a previously trained SVM. In the latter, of the two choices, cosine kernel is usually preferred. Recently, new channel modelling techniques for i-vectors, such as Probabilistic Linear Discriminant Analysis (PLDA) [22], have been reported to overcome problems of classical cosine-distance scoring of i-vectors with channel compensation.

5.3 Speaker Verification using SBE

In the following experiments we employ an approach based on SVM (trained on target and imposter features) for speaker modelling and scoring. We extract features according to the two methods described in Section 5.2. Regarding the i-vectors, we do not consider any channel compensation methods, since we are only focused in obtaining a baseline for our privacy-preserving methods, not state-of-the-art results. Nevertheless, it would be straightforward to incorporate (at least) LDA and WCCN compensation without significant alterations of the proposed scheme.

Before we can compute the supervectors or the i-vectors, we need to perform MAP adaptation of the UBM to each feature vector from the target speakers and the imposters, some technical issues arise. In order to perform the MAP adaptation to the target speakers, either the server needs to gain access to their recordings or each target speaker needs to gain access to the UBM. Additionally, since for each target speaker the same SBE parameters $\mathbf{A}$ and $\mathbf{w}$ must be used for computing the hashes from all supervectors or i-vectors, these parameters need to be shared between that target user and the server. Because of the privacy concerns from both parties, an alternative solution must be found. The solution we propose consists in having the server providing a toolkit that is seen as a black box by the target speakers and performs the necessary operations without disclosing sensitive information. This black box performs two operations. First, given feature vectors provided by each target speaker, obtain

the MAP adaptation of the UBM and then compute the corresponding supervectors or i-vectors. Second, given the SBE parameters from a target speaker, compute the SBE hashes for the supervectors or i-vectors corresponding to the imposters. The user can then send all SBE hashes to the server.

Once again, we consider both parties to possess *honest-but-curious* behaviour. Additionally, we assume that the black box provided by the server has been certified by a trusted third-party. Although the server cannot request any information from the users (and therefore we do not need to assume an attack model for it), the users themselves are allowed to perform Chosen-Plaintext Attacks (CPA) on the black box in order to indirectly learn information about the UBM.

5.3.1 SVM based Classifier

An SVM [31] is a supervised learning model usually used for data analysis, namely classification, pattern recognition and regression. Given a set of training examples, each tagged with one of many categories, an SVM training algorithm builds a model that tries to assign as many examples with the same tag as possible to the same region in space. It also attempts to make the gaps between these regions as wide as possible. Testing examples are classified according to the region in which they fall. Optionally, the distance between the testing examples and the boundaries of the region they belong to can be used as a classification confidence measure.

In our experiments an SVM is then trained to distinguish between vectors corresponding to the target speakers and vectors corresponding to the imposters [25]. To verify that a given test recording was indeed spoken by the speaker, the supervector derived from the recording is classified by the SVM. Once again, a priori distributions may be assigned to the parameters through factor analysis – in this case the factor vectors may themselves be used to represent the recordings, and classification may be performed directly with them. We use as a baseline classifier a conventional Radial Basis Function (RBF) kernel based SVM, where the kernel is given by $k(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma \cdot d^2(\mathbf{x}_i, \mathbf{x}_j)}$. Here $d(\mathbf{x}_i, \mathbf{x}_j)$ refers to the Euclidean distance

between $\mathbf{x}_i$ and $\mathbf{x}_j$, and γ is a scaling factor. The RBF kernel has been observed to result in good classification accuracies for the SV task [5]. Moreover, it can be easily adapted to our privacy-preserving schemes.

5.4 Results

In this section we present the results obtained on the PPSV task when SBE hashes are the technique used for maintaining data privacy. We start by analysing two baselines scenarios where SVM classifiers are trained using feature supervectors and i-vectors, respectively. We produce equivalent classifiers for the SBE hashes extracted from these features and compare their performance both in terms of Equal Error Rate (EER) and Detection Error Tradeoff (DET) curves, both commonly used metrics for the SV task. EER gives equal importance to false positives and false negatives, therefore being task independent, while DET curves complement EER by showing the impact of the trade-off between false positives and false negatives.

5.4.1 Feature Extraction

In our experiments we considered two types of features: the GSV and the i-vectors. The common step for obtaining both of them is to compute 39 MFCC features (12 MFCC coefficients plus the log-energy, together with the corresponding temporal differences and double differences), extracted from frames of 25ms at the rate of 100 frames per second (fps).

For the supervectors, following the method described in Section 5.2, we trained the UBM using all the training data from the YOHO Speaker Verification corpus [23, 24]. The UBM was adapted to each recording to obtain a single GSV. The length of the supervectors depends on the number of Gaussians in the UBM: a UBM with N Gaussians results in supervectors with $L = 39N$ dimensions.

As for the i-vectors, a UBM composed by 1024 Gaussians was trained using the National Institute of Standards and Technology (NIST) Speaker Recognition Evaluation (SRE) 2004,

#Gaussians	4	8	16	32	64	128
EER	3.55	1.52	0.60	0.25	0.22	0.21

Table 5.1: Speaker verification results, given in terms of EER (%age), when using supervectors on the YOHO Speaker Verification corpus.

2005 and 2006 corpora [88, 89, 90]. The TV factor matrix **T** was estimated according to [63], also using the same three corpora. The dimension of the TV sub-space was fixed to 400. 10 Expectation-Maximization (EM) iterations were applied consisting of a maximum likelihood estimation followed with minimum divergence update. The covariance matrix was not updated in any of the EM iterations. The estimated **T** matrix is used for extraction of the TV factors of the processing speech segments as described in [63].

5.4.2 Experiments using Supervectors

In our baseline experiments with supervectors, we considered the YOHO Speaker Verification corpus [23, 24]. More details on the contents of this corpus are presented in Section A.2. We did not explicitly record imposters – instead, for each of the 138 speakers in the corpus, the remaining 137 were used as imposters. We evaluated UBMs of different sizes, with the number of Gaussian components ranging from 4 to 128 Gaussians, to find the optimal settings. All experiments were performed using the LIBSVM toolkit [28]. Table 5.1 shows the results, averaging all the speakers, in terms of EER. As expected, the performance improves as the number of Gaussians increases up to a limit related to the size of the dataset. We do not present results with larger amounts (values up to 2048 Gaussians are common in the literature [34]) because, for this particular corpus, they do not provide improvements. In fact, the results obtained with mixtures of 32 Gaussians are already very close to the ones obtained with 128. Hence, the experiments with SBE hashes will involve only 32 Gaussians.

5.4.3 Experiments using I-vectors

We ran experiments using i-vectors on the *8conv-short3* and *8conv-10sec* test sets of the NIST SRE 2008 corpus [92], following the experimental setup defined in the NIST SRE 2008 task.

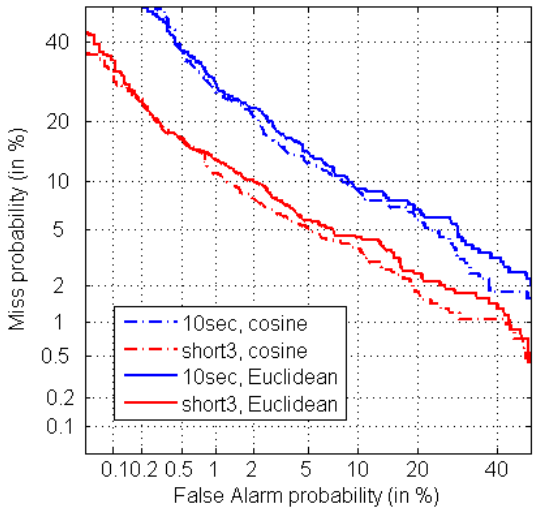


Figure 5.1: Speaker verification results, given in terms of DET curves, when using i-vectors on the NIST SRE 2008 corpus.

	8conv-short3	*8conv-10sec*
cosine kernel	5.1	9.4
Euclidean kernel	5.6	9.5

Table 5.2: Speaker verification results, given in terms of EER (%age), when using i-vectors on the NIST SRE 2008 corpus, and 1024 Gaussians are considered.

More details on the contents of this corpus are presented in Section A.3. Figure 5.1 and Table 5.2 show the results obtained using two different kernels. Since we are using an SVM for classifying the i-vectors, the choice of the selected kernels and scoring scheme must be addressed. Thus, we report results using SVMs with a cosine similarity based kernel, and an Euclidean distance based kernel. We need to consider both of them because although the first is known to be the most adequate for modelling i-vectors [33], the latter is the one to which the SBE hashes best relate. For obtaining the cosine based kernel, we first normalize the i-vectors, mapping them into the unit circle, and then use a linear kernel. For obtaining the Euclidean distance based kernel, we use an RBF kernel. Analysing the results, we can see that only minimal degradation is obtained when considering the Euclidean kernel instead of the cosine one, and therefore it is legitimate to consider the Euclidean distance as an adequate metric for comparing different i-vectors.

Note that these baseline results are slightly worse than the ones obtained in the NIST SRE

2008 task with state-of-the-art systems [92]. There are several reasons for this. First, in this work we are only interested in exploiting the TV modelling simply as a sort of factor analysis based front-end extractor for providing compact speaker representations of fixed length from which SBE hashes can be obtained. Thus, the problem of channel compensation is out of the scope of the present study and neither additional channel compensation techniques nor any kind of score normalization methods have been considered.

Although the effect of channel variability is partially minimized in our experimental framework, since we consider multi-session training conditions and only telephone-telephone trials, it is still responsible for a certain degradation in the results. Also, the limited amount of data used in our experimental framework for training the TV matrix further prevented us from achieving better baseline results. The decrease in performance is more visible in the *10sec* set, as the recordings are much shorter and therefore the lack of additional compensation techniques is more noticeable. This, however, does not invalidate any of our results using SBE in the following section, as we are primarily concerned with analysing how using privacy-preserving SBE hashes compares to a baseline SV system and we are not focused on improving the current state-of-the-art results for the SV task.

5.4.4 Experiments using SBE Hashes on Supervectors

The application of the SBE to SV systems using SVM is straightforward: if the classifier could be made to operate on SBE hashes instead of the original features themselves, SV may be performed without exposing speaker data. The RBF kernel must be modified to work with Hamming distances between SBE hashes:

$$k(\mathbf{x}, \mathbf{x}') = e^{-\gamma \cdot d_H^2(\mathbf{q}(\mathbf{x}), \mathbf{q}(\mathbf{x}'))} \tag{5.1}$$

Note that for a given $\mathbf{A}$ and $\mathbf{w}$, the modified kernel closely approximates the conventional RBF for small $d(\mathbf{x}, \mathbf{x}')$, but varies significantly from it at larger $d(\mathbf{x}, \mathbf{x}')$. While it does not satisfy Mercer's conditions [80] and cannot be considered a true kernel, in practice it is effective as

leakage	$\sim 5\%$	$\sim 25\%$	$\sim 50\%$
bpc=4	2.27	1.40	1.25
bpc=8	1.32	0.89	0.80
bpc=16	0.76	0.60	0.51

Table 5.3: Speaker verification results, given in terms of EER (%age), when using SBE hashes of supervectors on YOHO Speaker Verification corpus, and 32 Gaussians are considered.

we shall see in the experiments below.

As already mentioned in Section 2.2.2, the SBE has two parameters that can be varied: the quantization step size Δ and the number of bits M. The value of M by itself is not a useful number, as different values of L (dimensionality of the feature vectors) require different values of M; hence we report our results as a function of bits per coefficient (*bpc*), computed as M/L. The *bpc* allows us to govern the variance of the Universal Quantization (UQ). Recalling the definition presented in Section 2.2.2, leakage in this context refers to the fraction of recordings from any speakers whose SBE hashes have a normalized Hamming distance below the threshold at which Hamming distance d_H is proportional to the Euclidean distance d with respect to any recording from another speaker. This threshold was empirically set at 0.475. Once again, it was verified that for the values of *bpc* considered in our experiments, if the Euclidean distance between the original features was above Δ, then the corresponding normalized Hamming distance between the hashes was above that threshold. The amount of leakage is exclusively controlled by Δ, and its implications will be further discussed in Section 5.5.

The results obtained using SBE hashes on the supervectors extracted from the YOHO Speaker Verification corpus [23, 24] are presented in Table 5.3. As expected, both increasing Δ and *bpc* improves the classification performance. At *bpc*=16, the performance stabilizes rather quickly and thereafter is largely independent of Δ. We can easily check that our approach leads to an almost negligible increase in the classification error when compared to the non-secure version.

In order to further illustrate the performance of our approach in terms of EER, we compare our results with previous work on the PPSV task using supervectors on the YOHO Speaker Verification corpus [99]. The main difference in the two approaches is that while our approach

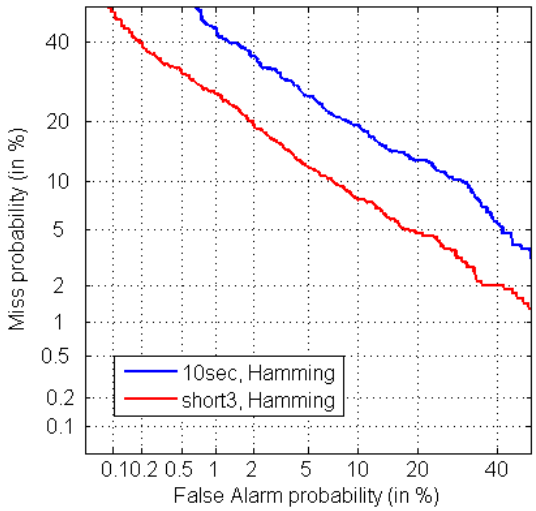


Figure 5.2: Speaker verification results, given in terms of DET curves, when using SBE hashes of i-vectors on the NIST SRE 2008 corpus.

	8conv-short3	*8conv-10sec*
Hamming kernel	8.8	14.7

Table 5.4: Speaker verification results, given in terms of EER (%age), when using SBE hashes of i-vectors on the NIST SRE 2008 corpus, and 1024 Gaussians are considered.

is based on SBE, which is a distance-preserving hashing scheme, the approach used by previous work is based on LSH, which, as mentioned in Section 2.2.1, only allows for the detection of nearest neighbours. The authors report an EER of 11.86%, a value far greater than the ones obtained by our approach.

5.4.5 Experiments using SBE Hashes on I-vectors

For the following set of experiments, we considered the same setup regarding the SVM kernel, the SBE configuration and the threshold value. The results using SBE hashes on the i-vectors extracted from the NIST SRE 2008 corpus [92] are presented in Figure 5.2 and Table 5.4. We considered *bpc*=16 and 50% leakage, since these were the values that provided the best results in the experiments with the YOHO Speaker verification corpus. We notice that there is a degradation in the SV performance achieved using our proposed privacy-preserving approach when compared with the non-private baseline. The reduced performance probably happens because the NIST SRE 2008 corpus [92] is much more challenging than the YOHO Speaker

Verification corpus [23, 24], as can be seen in the difference in terms of EER between both baseline results. Nevertheless, this is a small price to pay for keeping a user's identity secret. It must be also taken into consideration that the proposed system is built upon a baseline verification system that could be improved based on some of the strategies described in Section 5.2. Thus, a more robust non-private baseline could also contribute to improve the private counterpart. Overall, we consider these results promising and a new step forward towards the future development of robust and efficient privacy-preserving speaker authentication.

5.5 Privacy Analysis and Other Practical Issues

We recall the privacy requirements from Section 5.1: the system should not be able to learn anything about the speaker's identity, gender, nationality, etc. These data privacy requirements are a bit more relaxed than the ones considered in Section 4.5, as we do not have any concern about what was actually said by the user, as both the supervectors and the i-vectors ignore that information and only try to capture the uniqueness of the user's voice. We only need to keep the user's voice fingerprint private. Nevertheless, this still means that no information regarding the original features should be recovered.

An additional issue arises from the GMM-UBM technique we considered for the SV task. The problem is that both the user and the system have sensitive data that they wish to keep private. On the one hand the user has his own voice fingerprint, represented by the supervectors or the i-vectors, and on the other hand the system has its UBM trained using the audio of several speakers. Since the same SBE, random matrix $\mathbf{A}$ and dither vector $\mathbf{w}$, must be used to hide the private data from both the user and the system, two scenarios are possible: either the user possesses $\mathbf{A}$ and $\mathbf{w}$ as private keys or the system provides different $\mathbf{A}$ and $\mathbf{w}$ for each user.

In the first scenario, since the system only obtains the hashes and does not know $\mathbf{A}$ or $\mathbf{w}$, it provably has no means of recovering vectors from their hashes. Moreover, it also has no way of computing hashes given a data vector. As a result, the process is totally secure from

abuse by the system. Moreover, this also protects the user from imposters, as the user's private key is tied to his client device(s), and if these are secured, *e.g.* by a password, the system cannot be broken into by an imposter who poses as the user. The problem here is that since the system has no means of computing hashes, the user is in charge of downloading imposter data, creating a UBM, computing the corresponding hashes and sending them to the system, which is a computationally demanding set of tasks. One alternative is to use a single-class classifier, which only requires positive instances from the user. However the performance of such classifiers tends to be poor [99]. A more practical solution is for the user to generate his set of negative samples from the positive samples. This can be achieved by using the method proposed in [139], in which the user iteratively estimates SVM classifiers and generates negative instances that lie on the wrong side of the decision boundary. The authors of this technique state that after a few iterations the classification results are comparable to the ones obtained using sets of genuine positive and negative samples; however whether this is true for supervectors or i-vectors extracted from speech recordings remains to be seen. Finally, the black box solution described in Section 5.3 also solves the problem regarding the UBM without any loss in terms of performance.

In the second scenario, we have to rely on the non-invertibility assumption of the SBE hashes. Under this assumption, the system generates $\mathbf{A}$ or $\mathbf{w}$ and sends them to the user. The system is now in charge of generating imposter hashes. Users only send the SBE hashes of their enrolment data to the system. However, even if the SBE hashes are not invertible, this process is not secure. Since the system can generate hashes on its own, it does not prevent the system from searching public sources such as YouTube for instances of the user's voice. This may not be an acceptable compromise in many situations.

5.6 Summary

This chapter described how to implement a PPSV system using SBE hashes. Conventional remote SV systems rely on the system to have access to the user's recordings, or features derived from them, and also a model of the user's voice. In the proposed approaches, the

system has access to none of them. The features extracted from the user's recordings are transformed to bit strings in a way that allows the computation of approximate distances, instead of exact ones. Afterwards, an SVM classifier with a modified kernel operates on the hashes. This allows SV to be performed without exposing speaker data. Experiments showed that the secure system yielded similar results as its non-private counterpart. The approach presented introduces negligible computational overhead, specially if compared to Secure Multi-Party Computation (SMPC) methods, and may be extended to other types of biometric authentication. An analysis of the level of data privacy achieved by our approach was also presented.

6 Privacy-Preserving Speaker Verification using GC

This chapter covers the work performed on the Privacy-Preserving Speaker Verification (PPSV) task using Garbled Circuit (GC). Section 6.2 presents an overview on different Speaker Verification (SV) techniques using a Gaussian Mixture Model (GMM). Sections 6.3, 6.4 and 6.5 describe how to implement all the functions required for evaluating a GMM using GC. Section 6.6 contains some details on the experimental setup, followed by Section 6.7, which presents some experimental results. Finally, Section 6.8 shows an analysis on the level of privacy achieved by our approach.

6.1 Introduction

Current SV systems have significant privacy concerns, as they require access to recordings of a user's voice and/or a model representing it. A malicious hacker could gain access to this information and use it to impersonate the user. We recall our privacy premises for a SV system from Section 5.1: we require that the system should neither have access to the user's recordings nor possess a model of the user's speech. Once again, these almost-paradoxical sounding requirements for a voice-based biometric system are not, in fact, unreasonable.

6.2 Speaker Verification using GMM

Conventional SV systems require users to input voice samples into the system during an enrolment phase. The system then uses these samples to build a model for the corresponding user. Finally, the user provides fresh samples to be compared with his model during the verification phase. These SV systems normally perform a likelihood ratio test to confirm the

user's identity.

In the enrolment phase, each input sample $\mathbf{x}$ is first parameterized into a sequence of feature vectors $x_1, \ldots, x_T$, usually Mel-Frequency Cepstral Coefficients (MFCC) vectors. A large collection of recordings from non-target speakers is then used to train a Universal Background Model (UBM), λ_U, which is a GMM representing the global distribution of speech. The UBM is adapted to each user's enrolment recording in order to obtain a user-specific GMM, λ_S, through Maximum A Posteriori (MAP) adaptation [114]. Performing a MAP adaptation ensures a one-to-one correspondence between the Gaussians in the UBM and those in the user models.

Later, in the verification phase, the user provides a new input sample $\mathbf{x}$. The likelihood of each x_t given by either a UBM or a user-adapted GMM composed of M Gaussians is computed by:

$$P(x_t|\lambda_{S/U}) = \sum_{i=1}^{M} w_i^{S/U} \mathcal{N}\left(x_t, \mu_i^{S/U}, \Sigma_i^{S/U}\right), \tag{6.1}$$

where $\mathcal{N}\left(x_t, \mu_i^{S/U}, \Sigma_i^{S/U}\right)$ is the i^{th} multivariate Gaussian with mean $\mu_i^{S/U}$ and covariance $\Sigma_i^{S/U}$ computed at the feature vector x_t and $w_i^{S/U}$ is the relative weight of that Gaussian in the mixture. This process is repeated for all feature vectors in $\mathbf{x}$. Finally, the likelihood ratio is computed and a verification decision is made using:

$$\frac{P(\mathbf{x}|\lambda_S)}{P(\mathbf{x}|\lambda_U)} \begin{cases} \geq \theta & \text{accept user,} \\ < \theta & \text{reject user,} \end{cases} \tag{6.2}$$

where

$$P(\mathbf{x}|\lambda_{S/U}) = \prod_{t=1}^{T} P(x_t|\lambda_{S/U}) \tag{6.3}$$

and θ is a predefined threshold parameter. Since typical values for $P(x_t|\lambda_{S/U})$ are usually prone to underflow issues, their logarithms are normally considered instead. Therefore, Equations 6.1, 6.2 and 6.3 are equivalent, respectively, to:

$$\log P(x_t|\lambda_{S/U}) = \text{Logsum}_i \left(\log P(x_t|i)\right)) = \text{Logsum}_i \left(\log \left(w_i \, \mathcal{N}\left(x_t, \mu_i, \Sigma_i\right)\right)\right) \tag{6.4}$$

$$\log P(\mathbf{x}|\lambda_S) - \log P(\mathbf{x}|\lambda_U) \begin{cases} \geq \theta & \text{accept user,} \\ < \theta & \text{reject user,} \end{cases} \tag{6.5}$$

$$\log P(\mathbf{x}|\lambda_{S/U}) = \sum_{t=1}^{T} \log P(x_t|\lambda_{S/U}) \tag{6.6}$$

in the log notation.

6.2.1 Reformulation of the GMM based Classifier

We use a GMM-based approach for speaker modelling and we evaluate it against a reference UBM using a log-likelihood ratio. However, due to the nature of GC, first there is a need to reformulate most of the basic equations for properly evaluating a GMM, namely the logsum operation. This is needed due to the restriction of fixed-point representation and because we wish to minimize the number of operations to be performed in the encrypted domain.

For the details on our GMM based classifier, we rely on previous work on how to rewrite Equation 6.4 so that it is easier to evaluate in a privacy-preserving setting [100]. In short, in log notation each multivariate weighted Gaussian is represented as an $(N+1) \times (N+1)$ matrix $\overline{G_i}$:

$$\overline{G_i} = \left[\begin{array}{c:c} -\frac{1}{2}\Sigma_i^{-1} & \Sigma_i^{-1}\mu_i \\ \hdashline 0 & \overline{g_i} - \frac{1}{2}\mu_i^T\Sigma_i^{-1}\mu_i \end{array}\right], \tag{6.7}$$

$$\overline{g_i} = \log w_i - (N/2)\log(2\pi) - (1/2)\log|\Sigma_i|, \tag{6.8}$$

where N is the size of the feature vector x_t, leading to $\log\left(w_i\,\mathcal{N}\left(x_t, \mu_i, \Sigma_i\right)\right) = \overline{x_t}^T \overline{G_i} \overline{x_t}$, with $\overline{x_t} = [x_t|1]^T$. The expression for obtaining the log-likelihood between each Gaussian and the feature vector is further simplified into computing a single scalar product:

$$\log P(x_t|i) = \log\left(w_i\,\mathcal{N}\left(x_t, \mu_i, \Sigma_i\right)\right) = \hat{x_t}^T \hat{G}_i, \tag{6.9}$$

where $\hat{x_t}$ is an extended feature vector composed by pairwise products $\overline{x_t}^k \overline{x_t}^l$ and $\hat{G}_i$ is a

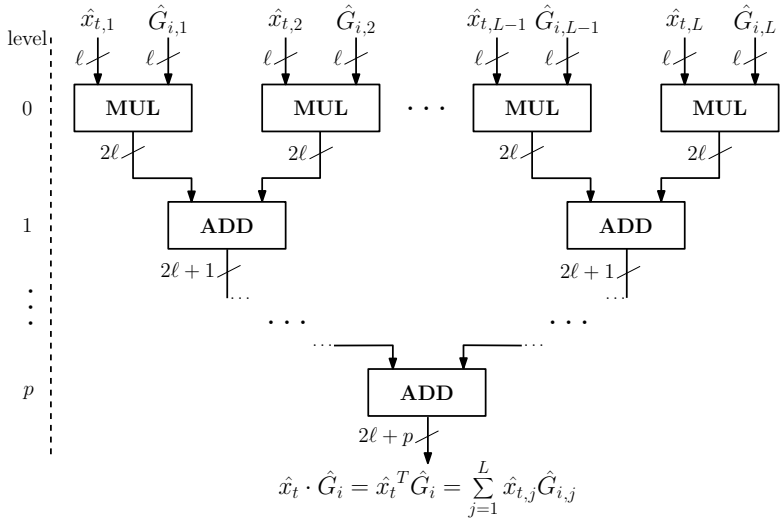


Figure 6.1: Scalar product using GC.

linearization of $\overline{G_i}$, both with size $L = (N+1)^2$. This formulation is going to be the one considered in all our experiments.

6.3 Basic Operations using GC

Given that evaluating a GMM requires many different complex operations and GC work in a modular way and at the logic gate level of abstraction, it is first necessary to start with some basic operations and build up from them. Such operations include addition (ADD), subtraction (SUB), multiplication (MUL), comparison (CMP), multiplexing (MUX), etc. From these, it is possible to obtain more complex operations, such as scalar product, linear filtering, distances, etc. A more complete list of operations, as well as circuit design details for all of them is available in the literature [68]. Notice that once all the operations required to evaluate a GMM are represented in terms of these basic operations and cast as Boolean circuits, it is straightforward to convert them to garbled versions of these circuits using the method described in Section 2.1.5.

The scalar product, a linear operation consisting of multiplications and additions, is of special interest for evaluating a GMM, as it is clear from Equation 6.9. A diagram illustrating the scalar product between two arrays $\hat{x}_t$ and $\hat{G}_i$ is presented in Figure 6.1. Notice that if each

element of each input array is represented using ℓ bits, the output value ${\hat{x}_t}^T \hat{G}_i$ needs in theory at most $2\ell + p$, $p = \lceil \log L \rceil$, bits to be represented. However, in many real situations this is often not the case, and we will address this problem in Section 6.7.

6.4 Logsum Operation using GC

The logsum operation is not only a core component for our GMM-based approach to SV but also very useful for many signal processing and pattern classification tasks, as it is a workaround to underflow/overflow problems. Such tasks include sparse signal recovery [127], matrix recovery [35], and in the evaluation of a large variety of statistical models such as latent Dirichlet allocation [14] and, of course, GMM [113]. Thus, the logsum operation may in fact be considered one of the key building blocks in statistical and signal processing computations.

Consider that it is necessary to compute the occurrence probability of an event X. The probability models employed are commonly mixture models of the form $P(X) = \sum_{i=1}^{N} P(X, Y_i)$. Rather than computing $P(X)$, one works with $\log P(X)$, which is in turn expressed in terms of the logarithm of the component probabilities as

$$z(m) = \log \left(\sum_{i=1}^{N} \exp(m_i) \right), \tag{6.10}$$

where $z(m) = \log\left(P(X)\right)$, and $m_i = \log\left(P(X, Y_i)\right)$. The equation above is a *logsum* operation. In order to further avoid overflow/underflow problems that arise when all the m_i terms become too large in magnitude, Equation 6.10 is usually further recast as

$$z(m) = m_X + \log \left(\sum_{i=1}^{N} \exp(m_i - m_X) \right), \tag{6.11}$$

with $m_X = \max_i m_i$.

$m : t$	$f(m) : n_1 \cdot m + n_2$	
$m \leq t_1$	$n_{1,1}$	$n_{2,1}$
$t_1 < m \leq t_2$	$n_{1,2}$	$n_{2,2}$
$\ldots$	$\ldots$	$\ldots$
$t_{k-2} < m \leq t_{k-1}$	$n_{1,k-1}$	$n_{2,k-1}$
$t_{k-1} < m$	$n_{1,k}$	$n_{2,k}$

Table 6.1: Look-up table with k entries for the PLA.

6.4.1 Piecewise Linear Approximation

Many algorithms rely on evaluating non-linear functions in order to compute an output. One such operation is the logsum, as it is clear from Equation 6.11. These non-linear functions are normally replaced by the corresponding Taylor series, whose terms are computed until the error between iterations falls below a pre-determined threshold. However, when limited computational resources are available, the non-linear function is often be replaced by a Piecewise Linear Approximation (PLA). A method for computing a PLA using GC has already been proposed [104], but it only focuses on a 1^{st}-order approach (Ramp Quantization (RQ)), discarding a 0^{th}-order approach (Step Quantization (SQ)). It is nevertheless instructive to evaluate both options in the context of GC, where computation is at a premium. Although it is expected that the error obtained with the SQ approach should be larger than the one obtained with the RQ approach, the computational benefits of the lower-order approximation can outweigh the increased error which is caused by loss of resolution.

A standard way to implement a PLA is to split the domain of the input function $f(\cdot)$ into k intervals, and then compute linear approximations $f(\cdot)$ for each interval, adjusting all the interval limits t_i and linearization parameters $n_{1,i}$ and $n_{2,i}$ in order to obtain the minimum error possible regarding $f(\cdot)$. These parameters are then placed in the form of a look-up table such as the one presented in Table 6.1. In order to obtain the correct parameter pair ($n_{1,i}$,$n_{2,i}$) for each value of m, it is necessary to compare m with all the t in order to find an index j such that $t_j < m < t_{j+1}$, allowing for computing $f_{\text{PLA}}(m) = n_{1,j+1} \cdot m + n_{2,j+1} \approx f(m)$, an approximation of the desired value. In the situations where $m < t_1$ or $m > t_{k-1}$, the approximation of the desired value is obtained by computing $f_{\text{PLA}}(m) = n_{1,1} \cdot m + n_{2,1}$

or $f_{\text{PLA}}(m) = n_{1,k} \cdot m + n_{2,k}$, respectively. This last scenario is not unlikely, since many functions have an unbounded domain, meaning that it is necessary to force artificial bounds when optimizing the locations of t. Therefore, special care is required while computing $n_{*,1}$ and $n_{*,k}$, so that they do not provide increasingly erroneous outputs as the difference between m and t_1 or t_{k-1} increases.

6.4.2 Logsum of $N = 2$ values

If only $N = 2$ inputs for the logsum operation are considered, then Equation 6.11 can be simplified to

$$z(m) = m_X + \log\left(1 + \exp(m_N - m_X)\right), \tag{6.12}$$

with $m_X = \max(m_1, m_2)$, $m_N = \min(m_1, m_2)$. After performing PLA, the approximation for the logsum operation is given by

$$z(m) \approx m_X + (n_1 \cdot (m_N - m_X) + n_2) \tag{6.13}$$

If $N = 2$ inputs and $k = 4$ look-up table entries are considered, a circuit diagram showing the GC diagram for the logsum operation is presented in Figure 6.2. There are three major blocks in this diagram, separated from each other by horizontal dotted lines. In each block, the individual boxes correspond to a simple operation. The first block contains the computation of all the necessary elements for obtaining $z(m)$, namely m_X, m_N and $m_{diff} = m_N - m_X$, by means of computing a minimum (MIN), a maximum (MAX) and a subtraction (SUB), respectively. The second block contains the circuit for obtaining the parameters for the PLA of the logsum operation. It starts by performing comparisons (CMP) between m_{diff} and the look-up table entries t, then the corresponding decisions i are used as control inputs to the hierarchical multiplexer (4-MUX), and finally the unknown parameters $n_{1,?}$ and $n_{2,?}$ are outputted. The third block performs the linear approximation of the logsum operation, outputting the final result. It performs a multiplication (MUL) and an addition (ADD) between m_{diff} and $n_{1,?}$ and $n_{2,?}$, respectively. The final operation is an addition between m_X

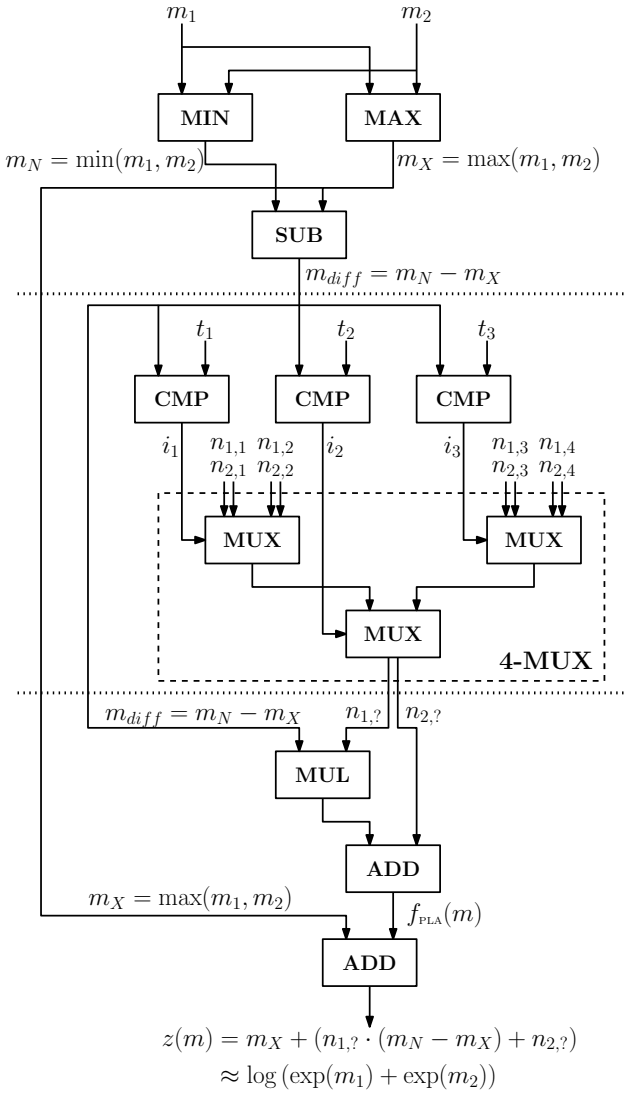


Figure 6.2: Circuit diagram for the logsum operation with $N = 2$ inputs and $k = 4$ look-up table entries.

and $f_{\text{PLA}}(m)$, thus completing the computation of the logsum.

This diagram raises several implementation concerns. One of the most relevant is the number of bits (ℓ) – how it increases with each computation and what can be done to keep it as small as possible, since more bits mean more computational costs and consequently longer execution times. The operations where the number of bits of the output is larger than the number of bits of the inputs are ADD (1 extra bit), SUB (1 extra bit) and MUL (ℓ extra bits), so the other operations will be ignored in this analysis. Regarding the SUB operation, since the output will always be a negative number, the sign bit is uninformative, and therefore can be discarded. As for the MUL and the first ADD operations, the ℓ Least Significant Bits (LSB) of $f_{\text{PLA}}(m)$ need to be truncated, as otherwise one may end up with a scaling factor different from the one of m_X (recall that all numbers are being represented with fixed-point instead of floating-point). We also remove the Most Significant Bits (MSB), as preliminary experiments suggest that it never contains any useful information, *i.e.*, it is always 0. On the last ADD we do not remove any bits, thus increasing the total number of bits by 1 every time the logsum operation is computed.

A second concern is the choice between the SQ and RQ approaches to the PLA. Both of them were implemented following the diagram in Figure 6.2, but whereas the RQ approach requires all the blocks, for the SQ approach the MUL and the first ADD block can be ignored, since in this situation $n_{1,?} = 0$. This also means that for the SQ approach there is no need to reduce the number of bits of the output.

6.4.3 Logsum of any N values

The just described implements the logsum operation for the simplest case of $N = 2$ input values. Now we present the generalization of this method to any given N. Let us start by recalling the original formulation of the logsum in Equation 6.11, which for $N = 4$ can be

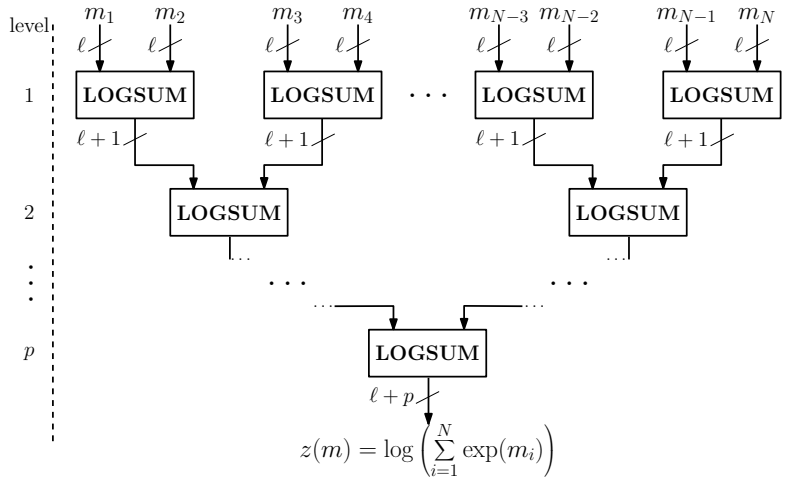


Figure 6.3: Circuit diagram for the logsum operation with any N inputs.

written as

$$\begin{aligned} z(m) &= \log\left(\exp(m_1) + \exp(m_2) + \exp(m_3) + \exp(m_4)\right) \\ &= \log\left(e^{\log(\exp(m_1)+\exp(m_2))} + e^{\log(\exp(m_3)+\exp(m_4))}\right) \\ &= \log\left(e^{z(m')} + e^{z(m'')}\right), \end{aligned} \tag{6.14}$$

with $m' = (m_1, m_2)$ and $m'' = (m_3, m_4)$. This means that the logsum for $N = 4$ input values can be obtained by computing two logsum operations, each of two input values. This process can be repeated as many times as needed, leading to a hierarchical tree structure for computing the logsum operation, as illustrated in Figure 6.3. With this structure, the difference in the number of bits between the output and the inputs is $p = \lceil \log_2 N \rceil$.

6.5 GMM Evaluation using GC

Given the algorithms for computing the scalar product and the logsum with GC, performing a GMM evaluation becomes a problem trivial to solve. The diagram showing it can be done when $M = 4$ Gaussians are considered is presented in Figure 6.4. The 4-LOGSUM block illustrates how the logsum of M elements can be computed as the expense of several logsums of two elements each.

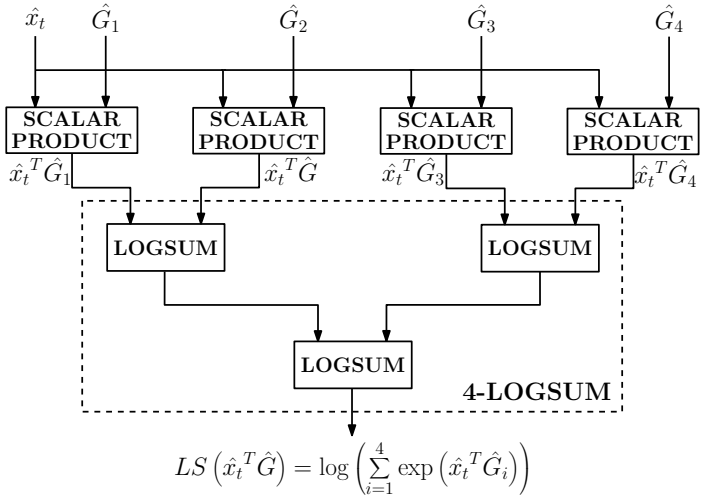


Figure 6.4: Evaluation of a Garbled GMM for $M = 4$.

The last step is to perform the ratio between the log-likelihoods of sample $\mathbf{x}$ with the UBM, λ_U, and the user-adapted GMM, λ_S. Since both of them are represented as logarithms, for obtaining $\log(P(\mathbf{x}|\lambda_{S/U}))$ one must sum all the values $\log(P(x_t|\lambda_{S/U}))$ (instead of multiplying them) and the ratio must be computed as a subtraction (instead of a division).

6.6 Speaker Verification using GC

For obtaining a PPSV using GC we need to perform the operations described in Section 6.2 using the building blocks presented in Sections 6.3, 6.4 and 6.5. The MAP adaptation required for obtaining the different GMM for each user can be performed using a black box in a manner similar to the one described in Section 5.3.

We consider both parties to behave as *honest-but-curious* attackers. Due to the nature of the protocols involved in sharing and executing GC, neither party can perform Chosen-Ciphertext Attacks (CCA1) or Adaptive Chosen-Ciphertext Attacks (CCA2), so we will only assume that they are capable of performing Chosen-Plaintext Attacks (CPA).

#Gaussians	16	32	64	128	256
non-private approach	5.18	4.04	2.99	2.03	1.43

Table 6.2: Speaker verification results, given in terms of EER (%age), when a baseline GMM technique is considered.

6.7 Results

In this section we present the results obtained on the PPSV task when GCs are the technique used for maintaining data privacy. We analyse a reference GMM classifier and use it as a baseline for our privacy-preserving classifier implemented using GC, comparing their performance in terms of Equal Error Rate (EER), for consistency with the results presented in Section 5.4. Additionally, due to the large number of encryptions and decryptions required when using GC, we also provide results in terms of execution times for different configurations.

6.7.1 Feature Extraction

We parameterized the audio signals from the YOHO Speaker Verification corpus [23, 24] into MFCC features extracted from 25ms frames at the rate of 100 frames per second (fps). For each audio frame, we extracted 13 MFCC coefficients and computed their corresponding temporal Δ and $\Delta\Delta$, resulting in 39 features per frame. The UBM was trained using the whole enrolment set, and each speaker-specific model was obtained by adapting the UBM to the enrolment data from each speaker.

6.7.2 Experiments using a baseline GMM

In our initial experiments we aimed to achieve reference results for our approach using GC. The results obtained in terms of EER on our SV approach are presented in Table 6.2. It is straightforward to verify that increasing the total number of Gaussians leads to reduced error rates, and that for 256 Gaussians we achieve a good EER value for the baseline.

#Gaussians	16	32	64	128	256
GC approach ($\ell = 16$)	6.00	4.73	3.06	2.09	1.60

Table 6.3: Speaker verification results, given in terms of EER (%age), when a Garbled GMM using GC technique is considered.

6.7.3 Experiments using GC

We now analyse our privacy-preserving approach using GC. We considered two toolkits on our experiments: the one developed by the Visual Information Processing and Protection (VIPP) research group [103] and the JustGarble toolkit [11]. The first one is easier to use and includes all the basic operations we needed for implementing the functions required for evaluating a Garbled GMM. The second is designed to minimize execution time by means of optimized code and tightly controlled memory allocation and access. Both toolkits perform offline Oblivious Transfer (OT) [9] using short ciphers [87], use the point-and-permute technique [73] for decryption at the gate level and evaluate XOR gates for "free" [66]. The circuits generated by both toolkits are relatively small in size (a few megabytes), which means that there are no significant communication costs associated with offline OT protocols, even if several different GC need to be stored remotely. We decided not to use any toolkit that allows for additional privacy and security techniques, such as reusing the same GC several times [51] or protection against malicious adversaries [71], as we were interested in obtaining the smallest possible execution times and these techniques lead to further computational costs. The results we obtained are presented in Table 6.3. We can see that increasing the number of Gaussians leads to smaller gaps between our approach and the baseline. Since in both approaches the computations performed are exactly the same, these gaps can only be justified by the number of bits ℓ considered, the use of a PLA to compute the logsum operation and/or the additional bits that are ignored after the scalar products and the logsum operations. As mentioned in Sections 6.3 and 6.4, each of these operations introduces many additional bits to the output. Regarding the scalar product, for the corpus and feature extraction process that was considered, preliminary experiments revealed that all additional bits could be discarded, as many MSB had the value '0' and ignoring some LSB did not significantly change the output

#Gaussians	16	32	64	128	256
t_{mean}	0.089	0.179	0.360	0.723	1.436
t_{max}	0.092	0.185	0.376	0.777	1.476

Table 6.4: Speaker verification results, given in terms of execution time (sec), when a Garbled GMM using GC technique is considered.

values. As for the logsum, both using a PLA and ignoring all but one of the additional bits does not seem to visibly affect the results [112]. Therefore, it is reasonable to assume that the difference between the results is due to $\ell = 16$ bit numbers being considered instead of a full 32-bit floating-point representation.

Since we also wanted our approach to be as practical as possible to implement in real-life situations, we also analysed it in terms of execution time. The results we obtained are presented in Table 6.4. The results presented correspond to the mean and the maximum time taken, in seconds, to evaluate a Garbled GMM, *i.e.*, to compute a single $\hat{x_t}^T \hat{G}$, when $\ell = 16$ is considered. They were obtained by running experiments using the JustGarble toolkit [11] on an Intel Core i7-3630QM CPU @ 2.40GHz. We notice that in all experiments the execution times are extremely fast, given that a privacy-preserving approach is considered. Also, the execution time scales linearly with the number of Gaussians. We only present the execution times for evaluating individual frames since all the required evaluations are independent from each other and may be computed in parallel. However, even if we consider that only a single computer core is available, the overall execution time would be approximately 5 minutes (evaluating $\hat{x_t}^T \hat{G}^U$ and $\hat{x_t}^T \hat{G}^S$ for all the frames extracted from a 4-second audio file sampled at 100 fps) in the situation where 64 Gaussians are considered.

In order to further illustrate the efficiency of our approach in terms of execution time, we compare our results with previous work on the PPSV task using GMM on the YOHO Speaker Verification corpus [100]. The main difference between the two approaches is that we replaced the Secure Two-Party Computation (STPC) protocols using Homomorphic Encryption (HE) with the one using GC. The authors report execution times of approximately 2 minutes for 256-bit cyphers and approximately 30 minutes for 1024-bit cyphers per 1-second audio file. However, given that we analyse 4-second audio files and that the security level provided by the

GC implementations we considered is comparable to the security provided by homomorphic cryptosystems using 1024-bit ciphers[1], it is fair to conclude that our approach was able to achieve almost an order of magnitude of improvement in terms of execution time.

6.8 Privacy Analysis and Other Practical Issues

The privacy guarantees provided by GC depend essentially on two factors: the key length of the cryptographic technique used for hiding the original information and the way these ciphers are transferred between the system and the user. The first factor relates to the amount of time that it would take for a hacker to break the cryptographic technique. Choosing larger values for the key length obviously leads to greater security. A compilation of minimum values for the key strength and other security parameters for various techniques is available and kept up to date on a regular basis [48]. The second factor is easily addressed by having both parties use OT for transferring information between them, as it has been demonstrated to keep data private. Also, as mentioned at the end of Section 2.1.5, several other security guarantees regarding GC have been proved in the literature.

Both toolkits used in our experiments consider adequate key lengths for encrypting the logic gates and the correctly implement the OT protocol, meaning that neither party can learn any sensitive information from the other, rendering our approach to PPSV using GC secure, given that we assume that no GC is ever reused. Additional security guarantees regarding GC, like the ones mentioned at the end of Section 2.1.5, were not considered but could be implemented at the expense of additional computational cost.

[1]The security of GC is proportional to the bit length of the symmetric keys used for encrypting the bits inside the logic tables. The security of homomorphic cryptosystems such as Paillier is proportional to the bit length of the cyphers themselves. Both GC toolkits we considered use a symmetric key bit length of 80, which is equivalent to bit length of 1024 for the cyphers produced by the Paillier cryptosystem [48].

6.9 Summary

This chapter described how to implement a PPSV system using GC. We started by proposing a technique for computing the logsum operation using GC. Our technique relies on replacing the logsum operation with an equivalent PLA, taking advantage of recent advances in efficient methods for both designing and implementing GC. We elaborate on how all the required blocks should be assembled in order to obtain small errors regarding the original logsum operation and very fast execution times. Then, by applying it to the UBM-GMM technique, we were able to obtain an implementation that simultaneously is secure, has high accuracy and is efficient.

7 Conclusions and Future Work

7.1 Conclusions

The main contribution of this thesis is having shown that the two speech processing tasks that were chosen as examples can be rendered secure, at the cost of almost negligible degradation, thus fulfilling the objectives highlighted in Section 1.4. To the best of our knowledge, the systems we designed, implemented and evaluated constitute therefore the first efficient way of doing privacy-preserving speech mining.

Concerning the Query-by-Example Speech Search (QESS), the proposed method enables the search to be performed in an encrypted domain, by analysing ciphered information computed from the original recordings. Unlike other hashing techniques, Secure Binary Embedding (SBE) makes it possible to compute the distance between vectors that are close enough, but are not perfect matches. This thesis showed how these hashes may be combined with Dynamic Time Warping (DTW) based on posterior derived features to perform secure speech search. Experiments showed that the proposed privacy-preserving system obtains similar results to its non-private counterpart.

Concerning Speaker Verification (SV) systems, conventional approaches rely on the system to have access to the user's recordings, or features derived from them, and also a model of the user's voice. In the approaches proposed in this project, the system does not have access to any of them. The features extracted from the user's recordings are transformed into bit strings in a way that enables the computation of approximate distances, instead of exact ones. Afterwards, an Support Vector Machine (SVM) classifier with a modified kernel operates on the hashes. This allows speaker verification to be performed without exposing speaker data. Again, experiments showed that the secure system yielded similar results to its non-private

counterpart.

In the final stage of this thesis, SBE was also successfully applied to a totally different class of problems in the area of natural language processing, namely Important Passage Retrieval (IPR) and multi-document summarization. This work was not directly related to speech processing, but was strongly motivated by the very encouraging results achieved for the two speech processing tasks.

7.2 Future Work

There are a few directions for future work that are worth considering. These include not only in improving the performance and/or security guarantees of the solutions presented in this thesis, but also applying privacy-preserving solutions to complex speech processing tasks.

Regarding the SBE hashing technique, the main problem relates to the absence of a formal proof of non-invertability of the hashes, although the hashing scheme provides information-theoretical security. This restricts how protocols involving a user and a remote system are designed, as well as limiting or even preventing distribution of the embedding parameters to other parties. Two approaches to addressing this issue are possible: the first consists in proving the non-invertibility of the SBE hashes, the second consists in analysing mechanisms in order to develop more secure embeddings. The second approach seems more interesting, as a new embedding scheme developed from scratch may not only have inherently non-invertibility properties from design, but also be less "noisy", leading to better performance when used for more complex speech processing tasks. In particular, we are already on our way to designing such as embedding technique (known as Secure Modular Hashing (SMH) [61]). This technique is currently under development by Abelino Jiménez, a PhD student under the supervision of Prof. Bhiksha Raj at Carnegie Mellon University, whose cooperation we would like to acknowledge.

Regarding the use of Secure Multi-Party Computation (SMPC) techniques, although in this thesis a privacy-preserving solution using Garbled Circuit (GC) was provided, it is of interest

to consider the use of Fully Homomorphic Encryption (FHE). Although such techniques are probably still a few years away from being efficient and practical to use, they provide clean protocols, strong security guarantees and theoretically no increase of error rates.

Finally, one speech processing task that was not considered in this thesis due to its complexity was Large Vocabulary Continuous Speech Recognition (LVCSR). It would be interesting to apply privacy-preserving techniques to address it, either similar to the ones considered here or different ones, as it is one the most important and unsolved problems in the speech processing community.

A Corpora

This appendix contains the details on the three main corpora used throughout this thesis. Section A.1 describes the Portuguese SpeechDat(II) corpus, considered in Chapter 4. Section A.2 describes the YOHO Speaker Verification corpus, considered in Chapters 5 and 6. Finally, Section A.3 describes the 2008 National Institute of Standards and Technology (NIST) Speaker Recognition Evaluation (SRE) corpus, considered in Chapter 5.

A.1 Portuguese SpeechDat(II) Corpus

The Portuguese SpeechDat(II) [56, 38] comprises 4027 Portuguese speakers (1861 males, 2166 females) recorded over the Portuguese fixed telephone network. This database is partitioned into 11 CDs. The speech databases built within the SpeechDat(II) project were validated by SPEX, the Netherlands, to assess their compliance with the SpeechDat format and content specifications. Speech samples are stored as sequences of 8-bit 8 kHz A-law. Each prompted utterance is stored in a separate file.

Each speech file has an American Standard Code for Information Interchange (ASCII) SAM label file with information about calling session, recording conditions, speaker sex, age and accent, signal file, recording date and time, assessment codes and the label file body itself. This includes the prompting script and the orthographic transcription. A pronunciation lexicon with citation phonemic transcriptions in Speech Assessment Methods Phonetic Alphabet (SAMPA) for each word is also included. Each speaker uttered the following items:

- 1 isolated single digit
- 1 sequence of 10 isolated digits

- 4 numbers: 1 sheet number (5+ digits), 1 telephone number (9-11 digits), 1 credit card number (14-16 digits), 1 PIN code (6 digits)
- 1 currency money amount
- 1 natural number
- 3 dates: 1 spontaneous (date or year of birth), 1 prompted date, 1 relative or general date expression
- 2 time phrases: 1 time of day (spontaneous), 1 time phrase (word style)
- 3 spelled words: 1 spontaneous (own forename), 1 city name, 1 real word for coverage
- 5 directory assistance utterances: 1 spontaneous, own forename, 1 city of birth / growing up (spontaneous), 1 frequent city name, 1 frequent company name, 1 common forename and surname
- 2 yes/no questions: 1 predominantly 'yes' question, 1 predominantly 'no' question
- 3 application words
- 1 keyword phrase using an embedded application word
- 4 phonetically rich words (chosen from a set of 4000)
- 9 phonetically rich sentences (chosen from a set of 3600)

The following age distribution has been obtained: 241 speakers are below 16 years old, 1404 speakers are between 16 and 30, 1532 speakers are between 31 and 45, 711 speakers are between 46 and 60, and 139 speakers are over 60.

A.2 YOHO Speaker Verification Corpus

The YOHO database [23, 24] contains a large scale, high-quality speech corpus to support text-dependent speaker authentication research, such as is used in secure access technology.

The data was collected in 1989 by International Telephone & Telegraph (ITT) under a US Government contract, but was not available for public use before 1994. Note that certain changes have been made to the corpus, mainly to insure the privacy of the speakers and some data has been withheld by the government for future use in testing.

YOHO contains:

- Combination lock phrases (*e.g.* 36-24-36)
- Collected over three-month period in a real-world office environment
- Four enrolment sessions per subject with 24 phrases per session
- Ten test sessions per subject with four phrases per session
- 8kHz sampling with 3.8 kHz analog bandwidth
- 1.5 gigabytes of data

The number of trials is thus sufficient to permit evaluation testing at high confidence levels. In each session, a speaker was prompted with a series of phrases to be read aloud each phrase was a sequence of three two-digit numbers (*e.g.* 35-72-41, pronounced thirty-five seventy-two forty-one). The first four sessions for a given speaker were enrolment sessions of 24 phrases and all additional sessions were verification trials of four phrases each. In all there are 552 enrolment sessions and 1380 trial sessions, with a nominal time interval of three days between sessions.

A.3 2008 NIST Speaker Recognition Evaluation Corpus

The 2008 NIST SRE corpus [92] consists of two sets (or conditions), one for training and one for testing, used for the NIST Year 2008 Speaker Recognition Evaluation Plan. Both conditions will consist of continuous conversational excerpts with no prior removal of intervals of silence. Also, except for summed channel telephone conversations and long interview

segments as described below, two separate conversation channels will be provided (to aid systems in echo cancellation, dialogue analysis, etc.). For all such two-channel segments, the primary channel containing the target speaker to be recognized will be identified.

A.3.1 Training Conditions

The six training conditions to be included involve target speakers defined by the following training data:

- *10-sec*: A two-channel excerpt from a telephone conversation estimated to contain approximately 10 seconds of speech of the target on its designated side. (An energy-based automatic speech detector will be used to estimate the duration of actual speech in the chosen excerpts.)
- *short2*: One two-channel telephone conversational excerpt, of approximately five minutes total duration, with the target speaker channel designated or a microphone recorded conversational segment of approximately three minutes total duration involving the target speaker and an interviewer. For the interview segments most of the speech will generally be spoken by the target speaker, and for consistency across the condition, a second zeroed out channel will be included.
- *3conv*: Three two-channel telephone conversational excerpts involving the target speaker on their designated sides.
- *8conv*: Eight two-channel telephone conversation excerpts involving the target speaker on their designated sides.
- *long*: A single channel microphone recorded conversational segment of eight minutes or longer duration involving the target speaker and an interviewer. Most of the speech will generally be spoken by the target speaker.
- *3summed*: Three summed-channel telephone conversational excerpts, formed by sample-by-sample summing of their two sides. Each of these conversations will include both the

target speaker and another speaker. These three non-target speakers will all be distinct.

A.3.2 Test Segment Conditions

The four test segment conditions to be included are the following:

- *10-sec*: A two-channel excerpt from a telephone conversation estimated to contain approximately 10 seconds of speech of the putative target speaker on its designated side (An energy-based automatic speech detector will be used to estimate the duration of actual speech in the chosen excerpts.)

- *short3*: A two-channel telephone conversational excerpt, of approximately five minutes total duration, with the putative target speaker channel designated or a similar such telephone conversation but with the putative target channel being a (simultaneously recorded) microphone channel or a microphone recorded conversational segment of approximately three minutes total duration involving the putative target speaker and an interviewer. For the interview segments, most of the speech will generally be spoken by the target speaker, and for consistency across the condition, a second zeroed out channel will be included.

- *long*: A single channel microphone recorded conversational segment of eight minutes or longer duration involving the putative target speaker and an interviewer. Most of the speech will generally be spoken by the target speaker.

- *summed*: A summed-channel telephone conversation formed by sample-by-sample summing of its two sides.

B Privacy-Preserving Extractive Document Summarization

This appendix covers the work performed on the Privacy-Preserving Extractive Document Summarization (PPEDS) task using Secure Binary Embedding (SBE) hashes. Since it possible to perform document summarization without using speech processing techniques, this is a clear example of how the same privacy-preserving techniques used for speech processing also apply to other research areas with promising results. Section B.2 contains a very brief overview on some extractive document summarization techniques. Section B.3 illustrates the details on the experimental setup. Section B.4 presents some experimental results. Finally, Section B.5 shows an analysis on the level of data privacy achieved by our approach.

B.1 Introduction

Document summarization is the problem of extracting the most important sentences in a document or set of documents. By "important", we mean those passages that capture most of the key information these documents are attempting to convey. A limitation to the usage of such methods is their assumption that the input texts are of public domain. However, problems arise when these documents cannot be made public. Consider the scenario where a company has millions of classified documents organized into several topics. The company may need to obtain a summary from each topic, but it lacks the computational power or know-how to do so. At the same time, they can not share those documents with a third party with such capabilities, as they may contain sensitive information. As a result, the company must obfuscate their own data before sending it to the third party, a requirement that is seemingly at odds with the objective of extracting summaries from it.

B.2 Extractive Document Summarization

Assessing content relevance is the first step performed by automatic summarization systems. There are two main approaches described in the literature. On the one hand, extractive summarization consists of determining the most relevant segments, usually sentences, of one or more information sources. On the other hand, automatic abstractive summarizers also need to identify the most relevant content that will then be submitted to transformation and generation stages of this kind of system. Since abstractive summarizers are more complex to implement and may produce syntactically incoherent sentences, most of the current work in automatic summarization focuses on extractive summarization.

B.2.1 Single-Document Summarization

Text and speech information sources influence the complexity of the document summarization approaches differently. For textual-based summarization, it is common to use complex information, such as syntactic, semantic and discourse information, either to assess relevance or to reduce the length of the output. However, speech-based summarization has an extra layer of complexity, caused by speech-related issues like recognition errors or disfluencies. As a result, it is useful to use speech-specific information (*e.g.* acoustic/prosodic features, recognition confidence scores) or to improve both the assessment of relevance and the intelligibility of automatic speech recognizer transcriptions. These problems not only increase the difficulty in determining the salient information, but also constrain the applicability of summarization techniques. Nevertheless, shallow text summarization approaches such as Latent Semantic Analysis (LSA) and Maximal Marginal Relevance (MMR) seem to achieve performances comparable to the ones using specific speech-related features [102].

The approach we considered for single-document summarization is textual Important Passage Retrieval (IPR) using the KP-Centrality method [116]. We chose this model for its adaptability to different types of information sources (*e.g.* text, audio and video) and state-of-the-art performance. It is based on the notion of combining key phrases with support sets. A support

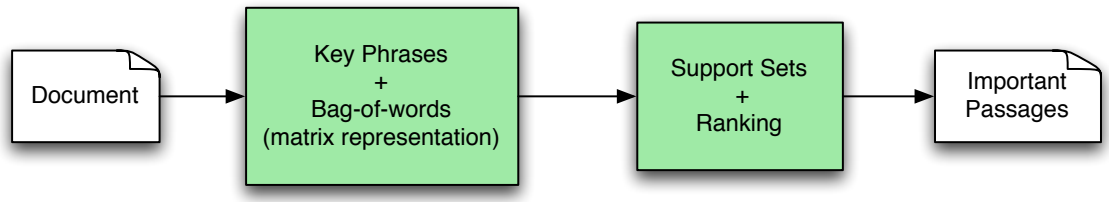


Figure B.1: Flowchart of the IPR method.

set is a group of the most semantically related passages. These semantic passages are selected using heuristics based on the passage order method. This type of heuristic explores the structure of the input source to partition the candidate passages to be included in the support set in two subsets: the ones closer to the passage associated with the support set under construction and the ones further apart. These heuristics use a permutation, $d_1^i, d_2^i, \cdots, d_{N-1}^i$, of the distances of the passages s_k to the passage p_i, related to the support set under construction, with $d_k^i = dist(s_k, p_i)$, $1 \leq k \leq N-1$, where N is the number of passages, corresponding to the order of occurrence of passages s_k in the input source. The evaluation metric that is normally used is the cosine distance.

The KP-Centrality method consists of two steps, as illustrated in Figure B.1. It extracts a set of key phrases using a supervised approach and combines them with a bag-of-words model in a compact matrix representation, given by:

$$\begin{bmatrix} w(t_1,p_1) & \dots & w(t_1,p_N) & w(t_1,k_1) & \dots & w(t_1,k_M) \\ \vdots & & & & & \vdots \\ w(t_T,p_1) & \dots & w(t_T,p_N) & w(t_T,k_1) & \dots & w(t_T,k_M) \end{bmatrix}, \tag{B.1}$$

where w is a function of the number of occurrences of each term t in every passage p or key phrase k, T is the number of terms, N is the number of sentences and M is the number of key phrases. Then, using $I \cup K \triangleq p_1, \dots, p_N, k_1, \dots, k_M = q_1, \dots, q_{N+M}$, a support set S_i is computed for each passage p_i using:

$$S_i \triangleq \{s \in I \cup K : sim(s, q_i) > \varepsilon_i \wedge s \neq q_i\}, \tag{B.2}$$

for $i = 1, \ldots, N + M$. Passages are ranked excluding the set of key phrases according to:

$$\underset{s \in (\cup_{i=1}^{N} S_i) - K}{\arg\max} \left| \{S_i : s \in S_i\} \right|. \tag{B.3}$$

B.2.2 Multi-Document Summarization

Popular baselines for multi-document summarization fall into one of the following general models: centrality-based, MMR and coverage-based methods. Additionally, methods such as KP-Centrality, which is centrality and coverage-based, follow more than one paradigm. In general, centrality-based models are used to produce generic summaries, while the MMR family generates query-oriented ones.

To determine the most representative sentences of a set of documents, we used a multi-document approach based on KP-Centrality known as Waterfall KP-Centrality [78]. Waterfall KP-Centrality iteratively combines the summaries of each document that was generated using KP-Centrality following a cascade process: it starts by merging the intermediate summaries of the first two documents, according to their chronological order. This merged intermediate summary is then summarized and merged with the summary of following document. We iterate this process through all documents until the most recent one. The summarization method uses as input a set of key phrases that we extract from each input document, joins the extracted sets, and ranks the key phrases using their frequency. To generate each intermediate summary, we use the top key phrases, excluding the ones that do not occur in the input document.

B.3 Experimental Setup

Our approach for PPEDS systems closely follows the formulation presented in Section B.2. However, the party owning all the documents, Alice, now needs to rely on a third party, Bob, to perform all the computations regarding the summarization process.

B.3.1 Document Summarizers

Typically, in a non-private scenario, Alice, the only party involved, performs key phrase extraction, combines them with the bag-of-words model in a compact matrix representation, computes the support sets for each document and finally uses them to retrieve the summaries. In our scenario, she must outsource the summarization process to Bob. However, Alice must first obfuscate the information contained in the compact matrix representation. If Bob receives this information as is, he could use the term frequencies to infer on the contents of the original documents and gain access to private or classified information which Alice does not wish to disclose. Alice computes SBE hashes of her compact matrix representation using the method described in Section 2.2.2, keeping the randomization parameters $\mathbf{A}$ and $\mathbf{w}$ to herself. She sends these hashes to Bob, who computes the support sets and extracts the important passages. Because Bob receives binary hashes instead of the original matrix representation, he must use the normalized Hamming distance instead of the cosine distance in this step, since it is the metric the SBE hashes best relate to. Finally, he returns the hashes corresponding to the important passages to Alice, who then uses them to get the information she desires.

B.3.2 Feature Extraction

We performed the key phrase extraction process using the Multi-purpose Automatic Topic Indexing (MAUI) toolkit [79] expanded with shallow semantic features, such as number of named entities, part-of-speech tags and four n-gram domain model probabilities. This expanded feature set leads to improvements regarding the quality of the key phrases.

B.4 Results

In this section we present the results obtained on the PPEDS task when SBE hashes are used for maintaining data privacy. We perform experiments both in the single-document and multi-document summarization scenarios.

	Concisus	PT-BN
cosine distance	0.575	0.612
Euclidean distance	0.507	0.590

Table B.1: Single-document summarization results, given in terms of ROUGE–1, when a baseline KP-Centrality technique is considered.

B.4.1 Single-Document Summarization Experiments

In order to evaluate our approach to single-document summarization, we performed experiments on the English version of the Concisus dataset [120] and the Portuguese Broadcast News (PT-BN) dataset [115]. The Concisus dataset is composed by seventy eight event reports and respective summaries, distributed across three different types of events: aviation accidents, earthquakes, and train accidents. The PT-BN dataset consists of automatic transcriptions of eighteen Broadcast News (BN) stories. News stories cover several generic topics such as society, politics and sports, among others. For each news story, there is a human-produced abstract, used as reference.

We present some baseline experiments in order to obtain reference values for our approach. We generated three passages summaries for Concisus Dataset, which are commonly found in online news web sites like Google News. In the experiments using the PT-BN dataset, the summary size was determined by the size of the reference human summaries, which consisted in about 10% of the input news stories. For both experiments, we used the cosine and Euclidean distances as evaluation metrics, since the first is the usual metric for computing textual similarity, but the second is the one that relates to the SBE technique. All results are presented in terms of Recall-Oriented Understudy for Gisting Evaluation (ROUGE) [69], in particular ROUGE–1, which is the most widely used evaluation measure for this scenario. The results we obtained for the Concisus and the PT-BN datasets are presented in Table B.1.

We considered forty key phrases in our experiments since it is the usual choice when news articles are considered. As expected, we noticed some slight degradation when the Euclidean distance is considered, but we still achieve better results than other state-of-the-art methods such as default centrality and LexRank. Reported results in the literature include

leakage	∼ 5%	∼ 25%	∼ 50%	∼ 75%	∼ 95%
bpc=4	0.365	0.437	0.465	0.486	0.495
bpc=8	0.424	0.384	0.436	0.452	0.500
bpc=16	0.384	0.416	0.450	0.463	0.517

Table B.2: Single-document summarization results, given in terms of ROUGE–1, when using SBE hashes on the Concisus dataset.

leakage	∼ 5%	∼ 25%	∼ 50%	∼ 75%	∼ 95%
bpc=4	0.314	0.340	0.470	0.478	0.562
bpc=8	0.327	0.324	0.486	0.507	0.527
bpc=16	0.338	0.336	0.520	0.473	0.550

Table B.3: Single-document summarization results, given in terms of ROUGE–1, when using SBE hashes on the PT-BN dataset.

ROUGE–1 = 0.443 and 0.531 using default centrality and ROUGE–1 = 0.428 and 0.471 using LexRank for the Concisus and PT-BN datasets, respectively [116]. This means that the forced change of metric due to the intrinsic properties of SBE does not affect the validity of our approach in any way.

For our privacy-preserving approach, we performed experiments using different values for the SBE parameters. The results we obtained in terms of ROUGE for the Concisus and the PT-BN datasets are presented in Tables B.2 and B.3, respectively.

Once again, leakage refers to the fraction of SBE hashes for which the normalized Hamming distance is proportional to the Euclidean distance between the original data vectors. The amount of leakage is exclusively controlled by Δ. bits per coefficient (*bpc*) is the ratio between the number of measurements M and the dimensionality of the original data vectors L, *i.e.*, $bpc = M/L$. As expected, increasing the amount of leakage (*i.e.*, increasing Δ) leads to improvements in the retrieval results. Surprisingly, changing the values of *bpc* does not lead to improved performance. The reason for this results might be due to the KP-Centrality method using support sets that consider multiple partial representations of the documents.

	TAC 2009	DUC 2007
cosine distance	0.514	0.370
Euclidean distance	0.489	0.364

Table B.4: Multi-document summarization results, given in terms of ROUGE–1, when a baseline Waterfall KP-Centrality technique is considered.

B.4.2 Multi-Document Summarization Experiments

For our approach to multi-document summarization, we performed experiments on the DUC 2007 [91] and TAC 2009 [93] datasets. DUC 2007 includes 45 clusters of 25 newswire documents and 4 human-created 250-word reference summaries. TAC 2009 has 44 topic clusters, each topic containing 2 sets of 10 news documents. There are 4 human-created 100-word reference summaries for each set. The reference summaries for the first set are query-oriented and for the second set are update summaries. In this work, we used the first set of reference summaries. We evaluated the different models by generating summaries with 250 words.

We present some baseline experiments in order to obtain reference values for our approach. We generated 250 words summaries for both TAC 2009 and DUC 2007 datasets. As in the single-document scenario, we used the cosine and the Euclidean distance as evaluation metrics. The results we obtained for the TAC 2009 and the DUC 2007 datasets are presented in Table B.4.

We considered forty key phrases in our experiments since it is the usual choice when news articles are considered. Once again, we notice some slight degradation when the Euclidean distance is considered, but we still achieve better results than other state-of-the-art methods. Reported results in the literature include ROUGE–1 = 0.328 and 0.415 using MEAD, ROUGE–1 = 0.327 and 0.392 using MMR, ROUGE–1 = 0.321 and 0.387 using Expect n-call@k, for the DUC 2007 and TAC 2009 datasets, respectively [78]. This means that the forced change of metric due to the intrinsic properties of SBE and the multiple application of SBE does not affect the validity of our approach in any way.

For our privacy-preserving approach, we performed experiments using different values for the SBE parameters. The results we obtained in terms of ROUGE for the DUC 2007 and the

leakage	∼ 5%	∼ 25%	∼ 50%	∼ 75%	∼ 95%
bpc=4	0.331	0.343	0.338	0.347	0.347
bpc=8	0.339	0.341	0.341	0.352	0.356
bpc=16	0.336	0.348	0.337	0.350	0.351

Table B.5: Multi-document summarization results, given in terms of ROUGE–1, when using SBE hashes on the DUC 2007 dataset.

leakage	∼ 5%	∼ 25%	∼ 50%	∼ 75%	∼ 95%
bpc=4	0.475	0.472	0.458	0.478	0.487
bpc=8	0.462	0.472	0.469	0.473	0.486
bpc=16	0.448	0.467	0.462	0.484	0.491

Table B.6: Multi-document summarization results, given in terms of ROUGE–1, when using SBE hashes on the TAC 2009.

TAC 2009 datasets are presented in Tables B.5 and B.6, respectively.

Unsurprisingly, increasing the amount of leakage (*i.e.*, increasing Δ) leads to improvements in the summarization results. However, changing *bpc* does not lead to improved performance. The reason for this might be due to the Waterfall KP-Centrality method using support sets that consider multiple partial representations of all documents. Even so, the most significant results is that for 95% leakage there is an almost negligible loss of performance.

B.5 Privacy Analysis and Other Practical Issues

The nature of the SBE hashes themselves provide information theoretical security if the configuration parameters $\mathbf{A}$ and $\mathbf{w}$ are not revealed. In our scenario, given that the hashes are computed over features extracted from the original texts and not on specific sentences of the texts themselves, even if somehow an attacker is able to steal the SBE configuration parameters and somehow break the hashes (which is unlikely), this still does not violate our privacy requisites in any way. This means that no useful information about the original data can be gained from their corresponding hashes.

B.6 Summary

This appendix described how a PPEDS system using SBE can be implemented. Our approach allows for third parties to retrieve important passages from documents without learning anything regarding their content. We used a hashing scheme to convert a key phrase and bag-of-words representation to bit strings in a way that allows the computation of approximate distances, instead of exact ones. Experiments show that our secure system yields similar results to its non-private counterpart on both single-document and multi-document scenarios. An analysis of the level of data privacy achieved by our approach was also presented.

Bibliography

[1] A. Abad, J. Luque, and I. Trancoso. Parallel Transformation Network Features for Speaker Recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 5300–5303, Prague, Czech Republic, May 22–27 2011.

[2] A. Abad, R. Astudillo, and I. Trancoso. The L2F Spoken Web Search System for MediaEval 2012. In *Working Notes of the MediaEval 2012 Workshop*, Pisa, Italy, October 4–5 2012.

[3] A. Adler. Biometric System Security. In *Handbook of Biometrics*, pages 381–402. Springer, 2008.

[4] B. Aiello, Y. Ishai, and O. Reingold. Priced Oblivious Transfer: How to Sell Digital Goods. In *Advances in Cryptology – EUROCRYPT 2001*, pages 119–135, Innsbruck, Austria, May 6–10 2001.

[5] X. Anguera. Minivectors: An Improved GMM-SVM Approach for Speaker Verification. In *10th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pages 2351–2354, Brighton, United Kingdom, September 6–10 2009.

[6] X. Anguera, L. Rodríguez-Fuentes, I. Szöke, A. Buzo, F. Metze, and M. Peñagarikano. Query-by-Example Spoken Term Detection Evaluation on Low-Resource Languages. In *International Workshop on Spoken Language Technologies for Under-Resourced Languages (SLTU)*, pages 24–31, St. Petersburg, Russia, May 14–16 2014.

[7] G. Asharov, Y. Lindell, T. Schneider, and M. Zohner. More Efficient Oblivious Transfer and Extensions for Faster Secure Computation. In *20th ACM Conference on Computer and Communications Security (CCS)*, pages 535–548, Berlin, Germany, November 4–8 2013.

[8] M. Atallah, F. Kerschbaum, and W. Du. Secure and Private Sequence Comparisons. In *ACM Workshop on Privacy in the Electronic Society (WPES)*, pages 39–44, Washington, DC, USA, October 30 2003.

[9] D. Beaver. Precomputing Oblivious Transfer. In *Advances in Cryptology – CRYPTO'95*, pages 97–109, Santa Barbara, CA, USA, August 27–31 1995.

[10] M. Bellare, V. Hoang, and P. Rogaway. Foundations of Garbled Circuits. In *19th ACM Conference on Computer and Communications Security (CCS)*, pages 784–796, Raleigh, NC, USA, October 16–18 2012.

[11] M. Bellare, V. Hoang, S. Keelveedhi, and P. Rogaway. Efficient Garbling from a Fixed-Key Blockcipher. In *IEEE Symposium on Security and Privacy*, pages 478–492, San Francisco, CA, USA, May 19–22 2013. URL http://cseweb.ucsd.edu/groups/justgarble/.

[12] T. Bianchi, A. Piva, and M. Barni. Implementing the Discrete Fourier Transform in the Encrypted Domain. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 1757–1760, Las Vegas, NV, USA, March 30 – April 4 2008.

[13] I. Blake and V. Kolesnikov. Strong Conditional Oblivious Transfer and Computing on Intervals. In *Advances in Cryptology – ASIACRYPT 2004*, pages 515–529, Jeju Island, South Korea, December 5–9 2004.

[14] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[15] L. Blum, M. Blum, and M. Shub. A Simple Unpredictable Pseudo-Random Number Generator. *SIAM Journal on Computing*, 15(2):364–383, 1986.

[16] D. Boneh, A. Sahai, and B. Waters. Functional Encryption: Definitions and Challenges. In *Theory of Cryptography – 8th Theory of Cryptography Conference (TCC)*, pages 253–273, Providence, RI, USA, March 28–30 2011.

[17] P. Boufounos. Universal Rate-Efficient Scalar Quantization. *IEEE Transactions on Information Theory*, 58(3):1861–1872, 2012.

[18] P. Boufounos and S. Rane. Secure Binary Embeddings for Privacy Preserving Nearest Neighbors. In *IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 216–221, Iguaçu Falls, Brazil, November 29 Nov – December 02 2011.

[19] S. Boyd, S.-J. Kim, D. Patil, and M. Horowitz. Digital Circuit Optimization via Geometric Programming. *Operations Research*, 53(6):899–932, 2005.

[20] L. Brandão. Secure Two-Party Computation with Reusable Bit-Commitments, via a Cut-and-Choose with Forge-and-Lose Technique (Extended Abstract). In *Advances in Cryptology – ASIACRYPT 2013*, pages 441–463, Bengaluru, India, December 1–5 2013.

[21] G. Brassard, C. Crépeau, and J.-M. Robert. All-or-Nothing Disclosure of Secrets. In *Advances in Cryptology – CRYPTO'86*, pages 234–238, Santa Barbara, CA, USA, August 11–15 1986.

[22] L. Burget, O. Plchot, S. Cumani, O. Glembek, P. Matejka, and N. Brümmer. Discriminatively trained Probabilistic Linear Discriminant Analysis for Speaker Verification. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 4832–4835, Prague, Czech Republic, May 22–27 2011.

[23] J. Campbell. Testing with the YOHO CD-ROM Voice Verification Corpus. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages 341–344, Detroit, MI, USA, May 9–12 1995.

[24] J. Campbell and A. Higgins. YOHO Speaker Verification LDC94S16, 1994. URL https://catalog.ldc.upenn.edu/LDC94S16.

[25] W. Campbell, J. Campbell, D. Reynolds, E. Singer, and P. Torres-Carrasquillo. Support Vector Machines for Speaker and Language Recognition. *Computer Speech & Language*, 20(2):210–229, 2006.

[26] W. Campbell, D. Sturim, and D. Reynolds. Support Vector Machines using GMM Supervectors for Speaker Verification. *IEEE Signal Processing Letters*, 13(5):308–311, 2006.

[27] R. Canetti, D. Dachman-Soled, V. Vaikuntanathan, and H. Wee. Efficient Password Authenticated Key Exchange via Oblivious Transfer. In *15th International Conference on Practice and Theory in Public-Key Cryptography (PKC)*, pages 449–466, Darmstadt, Germany, May 21–23 2012.

[28] C.-C. Chang and C.-J. Lin. LIBSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27:1–27:27, 2001.

[29] K. Chaudhuri, A. Sarwate, and K. Sinha. Near-Optimal Differentially Private Principal Components. In *26th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 998–1006, Lake Tahoe, NV, USA, December 3–6 2012.

[30] J.-S. Coron, T. Lepoint, and M. Tibouchi. Scale-Invariant Fully Homomorphic Encryption over the Integers. In *17th International Conference on Practice and Theory in Public-Key Cryptography (PKC)*, pages 311–328, Buenos Aires, Argentina, March 26–28 2014.

[31] C. Cortes and V. Vapnik. Support-Vector Networks. *Machine Learning*, 20(3):273–297, 1995.

[32] N. Dehak, R. Dehak, P. Kenny, N. Brümmer, P. Ouellet, and P. Dumouchel. Support Vector Machines versus Fast Scoring in the Low-Dimensional Total Variability Space for Speaker Verification. In *10th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pages 1559–1562, Brighton, United Kingdom, September 6–10 2009.

[33] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet. Front-End Factor Analysis for Speaker Verification. *IEEE Transactions on Audio, Speech & Language Processing (TASLP)*, 19(4):788–798, 2011.

[34] N. Dehak, O. Plchot, M. Bahari, Hugo Van hamme L. Burget and, and R. Dehak. GMM Weights Adaptation Based on Subspace Approaches for Speaker Verification. In *Odyssey 2014: The Speaker and Language Recognition Workshop*, pages 48–53, Joensuu, Finland, June 16–19 2014.

[35] Y. Deng, Q. Dai, R. Liu, Z. Zhang, and S. Hu. Low-Rank Structure Learning via Nonconvex Heuristic Recovery. *IEEE Transactions on Neural Networks and Learning Systems*, 24(3):383–396, 2013.

[36] W. Diffie and M. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[37] D. Ellis, C. Cotton, and M. Mandel. Cross-Correlation of Beat-Synchronous Representations for Music Similarity. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 57–60, Las Vegas, NV, USA, March 30 – April 4 2008.

[38] ELRA. Portuguese SpeechDat(II), 1997. URL http://catalog.elra.info/product_info.php?products_id=578.

[39] S. Even, O. Goldreich, and A. Lempel. A Randomized Protocol for Signing Contracts. *Communications of the ACM*, 28(6):637–647, 1985.

[40] F. Eyben, M. Wöllmer, and B. Schuller. openSMILE: The Munich Versatile and Fast Open-Source Audio Feature Extractor. In *18th ACM International Conference on Multimedia*, pages 1459–1462, Firenze, Italy, October 25–29 2010. URL http://www.audeering.com/research/opensmile.

[41] J. Fiscus, J. Garofolo, M. Przybocki, W. Fisher, and D. Pallett. 1997 English Broadcast News Speech (HUB4), 1998. URL https://catalog.ldc.upenn.edu/LDC98S71.

[42] R. Fisher and F. Yates. *Statistical Tables for Biological, Agricultural and Medical Research*. Oliver & Boyd, 1938.

[43] C. Fontaine and F. Galand. A Survey of Homomorphic Encryption for Nonspecialists. *EURASIP Journal on Information Security*, 2007.

[44] M. Freedman, K. Nissim, and B. Pinkas. Efficient Private Matching and Set Intersection. In *Advances in Cryptology – EUROCRYPT 2004*, pages 1–19, Interlaken, Switzerland, May 2–6 2004.

[45] C. Gentry and S. Halevi. Implementing Gentry's Fully-Homomorphic Encryption Scheme. In *Advances in Cryptology – EUROCRYPT 2011*, pages 129–148, Tallinn, Estonia, May 15–19 2011.

[46] C. Gentry, S. Halevi, and N. Smart. Fully Homomorphic Encryption with Polylog Overhead. In *Advances in Cryptology – EUROCRYPT 2012*, pages 465–482, Cambridge, United Kingdom, April 15–19 2012.

[47] Craig Gentry. Fully Homomorphic Encryption using Ideal Lattices. In *41st ACM Symposium on the Theory of Computing (STOC)*, pages 169–178, Bethesda, MD, USA, May 31 – June 2 2009.

[48] D. Giry and J.-J. Quisquater. Cryptographic Key Length Recommendation, 2009. URL http://keylength.com.

[49] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikäinen. On Private Scalar Product Computation for Privacy-Preserving Data Mining. In *7th International Conference on Information Security and Cryptology (ICISC)*, pages 104–120, Seoul, South Korea, December 2–3 2004.

[50] O. Goldreich. *The Foundations of Cryptography – Volume 1, Basic Techniques*. Cambridge University Press, 2001.

[51] S. Goldwasser, Y. Kalai, R. Popa, V. Vaikuntanathan, and N. Zeldovich. Reusable Garbled Circuits and Succinct Functional Encryption. In *45th ACM Symposium on the Theory of Computing (STOC)*, pages 555–564, Palo Alto, CA, USA, June 1–4 2013.

[52] D. Graff, J. Garofolo, J. Fiscus, W. Fisher, and D. Pallett. 1996 English Broadcast News Speech (HUB4), 1997. URL https://catalog.ldc.upenn.edu/LDC97S44.

[53] S. Han and W. Ng. Privacy-Preserving Linear Fisher Discriminant Analysis. In *12th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, pages 136–147, Osaka, Japan, May 20–23 2008.

[54] S. Han, W. Ng, and P. Yu. Privacy-Preserving Singular Value Decomposition. In *25th International Conference on Data Engineering (ICDE)*, pages 1267–1270, Shanghai, China, March 29 – April 2 2009.

[55] T. Hazen, W. Shen, and C. White. Query-by-Example Spoken Term Detection using Phonetic Posteriorgram Templates. In *IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU)*, pages 421–426, Merano/Meran, Italy, December 13–17 2009.

[56] H. Höge, H. Tropf, R. Winski, H. van den Heuvel, R. Haeb-Umbach, and K. Choukri. European Speech Databases for Telephone Applications. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 1771–1774, Munich, Germany, April 21–24 1997.

[57] Y. Huang, D. Evans, J. Katz, and L. Malka. Faster Secure Two-Party Computation Using Garbled Circuits. In *20th USENIX Security Symposium*, pages 539–554, San Francisco, CA, USA, August 8–12 2011. URL http://mightbeevil.com/framework.

[58] I. Ioannidis, A. Grama, and M. Atallah. A Secure Protocol for Computing Dot-Products in Clustered and Distributed Environments. In *International Conference Parallel Processing (ICPP)*, pages 379–384, Vancouver, British Columbia, Canada, August 18–21 2002.

[59] J. Garofolo J. Fiscus, J. Ajot and G. Doddington. Results of the 2006 Spoken Term Detection Evaluation. In *ACM SIGIR Workshop 'Searching Spontaneous Conversational Speech'*, pages 51–57, Amsterdam, The Netherlands, July 23–27 2007.

[60] K. Järvinen, V. Kolesnikov, A.-R. Sadeghi, and T. Schneider. Garbled Circuits for Leakage-Resilience: Hardware Implementation and Evaluation of One-Time Programs. In *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, pages 383–397, Santa Barbara, CA, USA, August 17–20 2010.

[61] A. Jiménez, B. Raj, J. Portêlo, and I. Trancoso. Secure Modular Hashing. In *IEEE International Workshop on Information Forensics and Security (WIFS)*, Rome, Italy, November 16–19 2015.

[62] P. Kenny, G. Boulianne, P. Ouellet, and P. Dumouchel. Joint Factor Analysis Versus Eigenchannels in Speaker Recognition. *IEEE Transactions on Audio, Speech & Language Processing (TASLP)*, 15(4):1435–1447, 2007.

[63] P. Kenny, P. Ouellet, N. Dehak, V. Gupta, and P. Dumouchel. A Study of Interspeaker Variability in Speaker Verification. *IEEE Transactions on Audio, Speech & Language Processing (TASLP)*, 16(5):980–988, 2008.

[64] D. Knuth. *The Art of Computer Programming, Volume II: Seminumerical Algorithms*. Addison-Wesley Professional, 1981.

[65] N. Koblitz. Elliptic Curve Cryptosystems. *Mathematics of Computation*, 48(177):203–209, 1987.

[66] V. Kolesnikov and T. Schneider. Improved Garbled Circuit: Free XOR Gates and Applications. In *International Colloquium on Automata, Languages and Programming (ICALP)*, pages 486–498, Reykjavik, Iceland, July 7–11 2008.

[67] V. Kolesnikov, A.-R. Sadeghi, and T. Schneider. Improved Garbled Circuit Building Blocks and Applications to Auctions and Computing Minima. In *International Conference on Cryptology and Network Security (CANS)*, pages 1–20, Kanazawa, Japan, December 12–14 2009.

[68] R. Lazzeretti. *Privacy Preserving Processing of Biomedical Signals with Application to Remote Healthcare Systems*. PhD thesis, Information Engineering Department, University of Siena, November 14 2012.

[69] C.-Y. Lin. ROUGE: A Package for Automatic Evaluation of Summaries. In *42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 74–81, Barcelona, Spain, July 25–26 2004.

[70] H.-Y. Lin and W.-G. Tzeng. An Efficient Solution to the Millionaires' Problem based on Homomorphic Encryption. In *3rd Applied Cryptography and Network Security Conference (ACNS)*, pages 456–466, New York City, NY, USA, June 7–10 2005.

[71] Y. Lindell and B. Pinkas. An Efficient Protocol for Secure Two-Party Computation in the Presence of Malicious Adversaries. In *Advances in Cryptology – EUROCRYPT 2007*, pages 52–78, Barcelona, Spain, May 20–24 2007.

[72] Y. Lindell and B. Pinkas. Secure Two-Party Computation via Cut-and-Choose Oblivious Transfer. *Journal of Cryptology*, 25(4):680–722, 2012.

[73] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. Fairplay – Secure Two-Party Computation System. In *13th USENIX Security Symposium*, pages 287–302, Atlanta, GA, USA, August 9–13 2004.

[74] J. Mamou, B. Ramabhadran, and O. Siohan. Vocabulary Independent Spoken Term Detection. In *30th Annual International ACM SIGIR Conference*, pages 615–622, Amsterdam, The Netherlands, July 23–27 2007.

[75] G. Marsaglia. Diehard Battery of Tests of Randomness, 1995. URL http://www.stat.fsu.edu/pub/diehard/.

[76] L. Marujo, J. Portêlo, D. Matos, J. Neto, A. Gershman, J. Carbonell, I. Trancoso, and B. Raj. Privacy-Preserving Important Passage Retrieval. In *ACM SIGIR Workshop*

'Privacy-Preserving Information Retrieval', pages 7–12, Gold Coast, QLD, Australia, July 11 2014.

[77] L. Marujo, J. Portêlo, W. Ling, D. Matos, J. Neto, A. Gershman, J. Carbonell, I. Trancoso, and B. Raj. Privacy-Preserving Multi-Document Summarization. In *ACM SIGIR Workshop 'Privacy-Preserving Information Retrieval'*, pages 1–4, Santiago, Chile, August 13 2015.

[78] L. Marujo, R. Ribeiro, D. Matos, J. Neto, A. Gershman, and J. Carbonell. Extending a Single-Document Summarizer to Multi-Document: A Hierarchical Approach. In *4th Joint Conference on Lexical and Computational Semantics (*SEM)*, pages 176–181, Denver, CO, USA, June 4–5 2015.

[79] O. Medelyan, V. Perrone, and I. Witten. Subject Metadata Support powered by MAUI. In *Joint International Conference on Digital Libraries (JCDL)*, pages 407–408, Gold Coast, QLD, Australia, June 21–25 2010.

[80] James Mercer. Functions of Positive and Negative Type, and their Connection with the Theory of Integral Equations. *Philosophical Transactions of the Royal Society of London*, 209(A):441–458, 1909.

[81] F. Metze, X. Anguera, E. Barnard, M. Davel, and G. Gravier. The Spoken Web Search Task at MediaEval 2012. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 8121–8125, Vancouver, British Columbia, Canada, May 26–31 2013.

[82] D. Miller, M. Kleber, C.-L. Kao, O. Kimball, T. Colthurst, S. Lowe, R. Schwartz, and H. Gish. Rapid and Accurate Spoken Term Detection. In *8th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pages 314–317, Antwerp, Belgium, August 27–31 2007.

[83] P. Mohassel and B. Riva. Garbled Circuits Checking Garbled Circuits: More Efficient and Secure Two-Party Computation. In *Advances in Cryptology – CRYPTO 2013*, pages 36–53, Santa Barbara, CA, USA, August 18–22 2013.

[84] M. Müller. Dynamic Time Warping. In *Information Retrieval for Music and Motion*, pages 69–84. Springer, 2007.

[85] M. Naor and B. Pinkas. Oblivious Transfer and Polynomial Evaluation. In *31st ACM Symposium on the Theory of Computing (STOC)*, pages 245–254, Atlanta, GA, USA, May 1–4 1999.

[86] M. Naor and B. Pinkas. Oblivious Transfer with Adaptive Queries. In *Advances in Cryptology – CRYPTO'99*, pages 573–590, Santa Barbara, CA, USA, August 15-19 1999.

[87] M. Naor and B. Pinkas. Efficient Oblivious Transfer Protocols. In *12th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 448–457, Washington, DC, USA, January 7–9 2001.

[88] NIST. The NIST Year 2004 Speaker Recognition Evaluation Plan, 2004. URL http://www.itl.nist.gov/iad/mig/tests/spk/2004/SRE-04_evalplan-v1a.pdf.

[89] NIST. The NIST Year 2005 Speaker Recognition Evaluation Plan, 2005. URL http://www.itl.nist.gov/iad/mig/tests/spk/2005/sre-05_evalplasn-v6.

[90] NIST. The NIST Year 2006 Speaker Recognition Evaluation Plan, 2006. URL http://www.itl.nist.gov/iad/mig/tests/spk/2006/sre-06_evalplan-v9.

[91] NIST. Document Understanding Conference (DUC), 2007. URL http://www-nlpir.nist.gov/projects/duc/duc2007/tasks.html.

[92] NIST. The NIST Year 2008 Speaker Recognition Evaluation Plan, 2008. URL http://www.itl.nist.gov/iad/mig/tests/sre/2008/sre08_evalplan_release4.pdf.

[93] NIST. Text Analysis Conference (TAC), 2009. URL http://www.nist.gov/tac/2009/Summarization/.

[94] NIST. OpenKWS13 Keyword Search Evaluation Plan, 2013. URL http://www.nist.gov/itl/iad/mig/upload/OpenKWS13-EvalPlan.pdf.

[95] NIST. Draft KWS14 Keyword Search Evaluation Plan, 2014. URL http://nist.gov/itl/iad/mig/upload/KWS14-evalplan-v11.pdf.

[96] P. Paillier. Public-Key Cryptosystems based on Composite Degree Residuosity Classes. In *Advances in Cryptology – EUROCRYPT'99*, pages 223–238, Prague, Czech Republic, May 2–6 1999.

[97] P. Paillier and D. Pointcheval. Efficient Public-Key Cryptosystems Provably Secure Against Active Adversaries. In *Advances in Cryptology – ASIACRYPT'99*, pages 165–179, Singapore, November 14–18 1999.

[98] M. Pathak and B. Raj. Privacy Preserving Speaker Verification Using Adapted GMMs. In *12th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pages 2405–2408, Florence, Italy, August 27–31 2011.

[99] M. Pathak and B. Raj. Privacy-Preserving Speaker Verification as Password Matching. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1849–1852, Kyoto, Japan, March 25–30 2012.

[100] M. Pathak and B. Raj. Privacy-Preserving Speaker Verification and Identification Using Gaussian Mixture Models. *IEEE Transactions on Audio, Speech & Language Processing (TASLP)*, 21(2):397–406, 2013.

[101] M. Pathak, J. Portêlo, B. Raj, and I. Trancoso. Privacy-Preserving Speaker Authentication. In *Information Security Conference (ISC)*, pages 1–22, Passau, Germany, September 19–21 2012.

[102] G. Penn and X. Zhu. A Critical Reassessment of Evaluation Baselines for Speech Summarization. In *46th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 470–478, Columbus, OH, USA, June 15–20 2008.

[103] T. Pignata. Garbled Circuit Designer and Executer from the Visual Information Processing and Protection (VIPP) Research Group, 2012. URL http://clem.dii.unisi.it/~vipp/index.php/software/135-garbledcircuit.

[104] T. Pignata, R. Lazzeretti, and M. Barni. General Function Evaluation in a STPC Setting via Piecewise Linear Approximation. In *IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 55–60, Costa Adeje, Tenerife, Spain, December 2–5 2012.

[105] J. Portêlo, B. Raj, A. Abad, and I. Trancoso. On the Implementation of a Secure Musical Database Matching. In *19th European Signal Processing Conference (EUSIPCO)*, pages 1949–1953, Barcelona, Spain, August 29 – September 2 2011.

[106] J. Portêlo, B. Raj, and I. Trancoso. Attacking a Privacy-Preserving Music Matching Algorithm. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1821–1824, Kyoto, Japan, March 25–30 2012.

[107] J. Portêlo, A. Abad, B. Raj, and I. Trancoso. Secure Binary Embeddings of Front-End Factor Analysis for Privacy Preserving Speaker Verification. In *14th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pages 2494–2498, Lyon, France, August 25–29 2013.

[108] J. Portêlo, B. Raj, P. Boufounos, I. Trancoso, and A. Abad. Speaker Verification using Secure Binary Embeddings. In *21st European Signal Processing Conference (EUSIPCO)*, Marrakech, Morocco, September 9–13 2013.

[109] J. Portêlo, B. Raj, A. Abad, and I. Trancoso. Privacy-Preserving Speaker Verification using Secure Binary Embeddings. In *37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1268–1272, Opatija, Croatia, May 26–30 2014.

[110] J. Portêlo, B. Raj, A. Abad, and I. Trancoso. Privacy-Preserving Speaker Verification using Garbled GMMs. In *22nd European Signal Processing Conference (EUSIPCO)*, pages 2070–2074, Lisbon, Portugal, September 1–5 2014.

[111] J. Portêlo, B. Raj A. Abad, and I. Trancoso. Privacy-Preserving Query-by-Example Speech Search. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1797–1801, Brisbane, QLD, Australia, April 19–24 2015.

[112] J. Portêlo, B. Raj, and I. Trancoso. Logsum using Garbled Circuits. *PLoS ONE*, March 26 2015.

[113] D. Reynolds and R. Rose. Robust Text-Independent Speaker Identification using Gaussian Mixture Speaker Models. *IEEE Transactions on Speech and Audio Processing*, 3 (1):72–83, 1995.

[114] D. Reynolds, T. Quatieri, and R. Dunn. Speaker Verification Using Adapted Gaussian Mixture Models. *Digital Signal Processing*, 10(1):19–41, 2000.

[115] R. Ribeiro and D. Matos. Improving Speech-to-Text Summarization by using Additional Information Sources. In *Multi-source Multilingual Information Extraction and Summarization*, pages 277–298. Springer, 2013.

[116] R. Ribeiro, L. Marujo, D. Matos, J. Neto, A. Gershman, and J. Carbonell. Self Reinforcement for Important Passage Retrieval. In *36th Annual International ACM SIGIR Conference*, pages 845–848, Dublin, Ireland, July 28 – August 1 2013.

[117] R. Rivest, L. Adleman, and M. Dertouzos. On Data Banks and Privacy Homomorphisms. *Foundations of Secure Computation*, 4(11):169–180, 1978.

[118] R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[119] L. Rodríguez-Fuentes, A. Varona, M. Peñagarikano, G. Bordel, and M. Díez. High-Performance Query-by-Example Spoken Term Detection on the SWS 2013 Evaluation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7819–7823, Florence, Italy, May 4–9 2014.

[120] H. Saggion and S. Szasz. The CONCISUS Corpus of Event Summaries. In *8th International Conference on Language Resources and Evaluation (LREC)*, pages 2031–2037, Istanbul, Turkey, May 23–25 2012.

[121] A. Sahai and B. Waters. Fuzzy Identity-based Encryption. In *Advances in Cryptology – EUROCRYPT 2005*, pages 457–473, Aarhus, Denmark, May 22–26 2005.

[122] J. Sakuma and R. Wright. Privacy-Preserving Evaluation of Generalization Error and Its Application to Model and Attribute Selection. In *1st Asian Conference on Machine Learning (ACML)*, Nanjing, China, November 2-4 2009.

[123] B. Schneier. *Applied Cryptography – Protocols, Algorithms, and Source Code in C*. John Wiley & Sons Inc., 1996.

[124] P. Scholl and N. Smart. Improved Key Generation for Gentry's Fully Homomorphic Encryption Scheme. In *IMA International Conference on Cryptography and Coding (IMACC)*, pages 10–22, Oxford, United Kingdom, December 12–15 2011.

[125] M. Shashanka and P. Smaragdis. Secure Sound Classification: Gaussian Mixture Models. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 1088–1091, Toulouse, France, May 14–19 2006.

[126] M. Shashanka and P. Smaragdis. Privacy-Preserving Musical Database Matching. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 319–322, New Paltz, NY, USA, October 21–24 2007.

[127] Y. Shen, J. Fang, and H. Li. Exact Reconstruction Analysis of Log-Sum Minimization for Compressed Sensing. *IEEE Signal Processing Letters*, 20(12):1223–1226, 2013.

[128] P. Smaragdis and M. Shashanka. A Framework for Secure Speech Recognition. *IEEE Transactions on Audio, Speech & Language Processing (TASLP)*, 15(4):1404–1413, 2007.

[129] E. Suthampan and S. Maneewongvatana. Privacy Preserving Decision Tree in Multi Party Environment. In *2nd Asia Information Retrieval Symposium (AIRS)*, pages 727–732, Jeju Island, South Korea, October 13–15 2005.

[130] I. Szöke, J. Tejedor, M. Fapso, and J. Colás. BUT-HCTLab Approaches for Spoken Web Search – MediaEval 2011. In *Working Notes of the MediaEval 2011 Workshop*, Pisa, Italy, September 1–2 2011.

[131] I. Trancoso, J. Portêlo, B. Raj, G. Chollet, N. Cannings, D. Petrovska-Delacrétaz, A. Badii, and J.-J. Quisquater. Privacy Preserving Speech Processing. In *Poster presentation at the CHIST-ERA Conference*, Lisbon, Portugal, June 16–18 2015.

[132] A. Uitdenbogerd and J. Zobel. Melodic Matching Techniques for Large Music Databases. In *7th ACM International Conference on Multimedia*, pages 57–66, Orlando, FL, USA, October 30 – November 5 1999.

[133] J. Vaidya and C. Clifton. Privacy-Preserving k-means Clustering over Vertically Partitioned Data. In *9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 206–215, Washington, DC, USA, August 24–27 2003.

[134] A. Wang. An Industrial Strength Audio Search Algorithm. In *4th International Conference on Music Information Retrieval (ISMIR)*, pages 7–13, Baltimore, MD, USA, October 27–30 2003.

[135] A. Wang. The Shazam Music Recognition Service. *Communications of the ACM*, 49 (8):44–48, 2006.

[136] H. Wang, T. Lee, C.-C. Leung, B. Ma, and H. Li. Using Parallel Tokenizers with DTW Matrix Combination for Low-Resource Spoken Term Detection. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 8545–8549, Vancouver, British Columbia, Canada, May 26–31 2013.

[137] S. Warner. Randomized Response: A Survey Technique for Eliminating Evasive Answer Bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.

[138] A. C.-C. Yao. Protocols for Secure Computations (Extended Abstract). In *23rd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 160–164, Chicago, IL, USA, November 3–5 1982.

[139] H. Yu, J. Han, and K. Chang. PEBL: Positive Example based Learning for Web Page Classification using SVM. In *8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 239–248, Edmonton, Alberta, Canada, July 23–26 2002.

[140] S. Zahur, M. Rosulek, and D. Evans. Two Halves Make a Whole: Reducing Data Transfer in Garbled Circuits using Half Gates. In *Advances in Cryptology – EUROCRYPT 2015*, pages 220–250, Sofia, Bulgaria, April 26–30 2015.

[141] J. Zhan and S. Matwin. Privacy-Preserving Support Vector Machine Classification. *International Journal of Intelligent Information and Database Systems*, 1(3–4):356–385, 2007.