An Information Theoretic Approach for Privacy Preservation in Distance-based Machine Learning

Submitted in partial fulfillment of the requirements for

the degree of

Doctor of Philosophy

in

Electrical & Computer Engineering

Abelino Enrique Jiménez Gajardo

B.S., Mathematical Engineering, Universidad de Chile
M.S., Electrical & Computer Engineering, Carnegie Mellon University

Carnegie Mellon University
Pittsburgh, PA

June 2019

Acknowledgements

To my family, Nicole and Agustín, for their love and infinite support in this adventure that takes us away from our work, home and friends.

To my parents Evangelina and Abelino, my sister Eliana, my in-laws Ezzio, Valentina, Pablo and Isabella, and our grandmom María, for their love and support from afar.

To my Pittsburgh family, Pía, Militza, Erika, Christian, Cristobal and Gabriel, for having been the incredible friends that they were, and ending up becoming another part of our family.

To my lifelong friends, Karina Díaz and Gonzalo Vidueria, who despite the distance, were always with their support during all these years.

To my advisor, professor Bhiksha Raj, for giving me the opportunity of working with him and his infinite patience and comprehension. Thanks for the tremendous freedom I had during the development of this work.

To the committee members, professors Manuel Blum, Aswin Sankaranarayanan and Bryan Parno, for being willing to participate in the revision of this work, providing a valuable feedback and give me comments that have always helped improve it.

To the Robust-MLSP research group, professors Rita Singh and Richard Stern, and my mates Benjamin Elizalde, Anurag Kumar, Anders Oland, Nia Peters, Nikolas Wolfe, Joana Correia, Raymond Xia, Tyler Vuong, Yang Gao, Yandong Wen, Wenbo Liu, Wenbo Zhao and Mahmoud Alismail. I learned a lot from each of you. I am fortunate to have you as colleagues and friends.

To Carnegie Mellon University, for being an incredible place to learn and grow.

To CONICYT, through the program Becas Chile, that funded this research.

Abstract

As cloud-based services become increasingly popular as platforms for storage and computation, privacy issues relating to their use have become increasingly important. Much of the data stored on cloud platforms are private, belonging to individuals or institutions who often desire to utilize the facilities provided by these platforms, but, at the same time, do not desire to expose their data to the platform itself.

Encrypting the data prior to storage on the cloud helps to protect private information. However, this causes problems if we need to perform computations on them, for instance, to train some machine learning algorithm. This requires the server to observe the content, so decryption is necessary. This gives rise to privacy concerns in different cloud computing settings. Several solutions based on cryptographic techniques have been proposed to address the issue. However, they have high computational cost and high bandwidth requirements, and in practice are difficult to scale.

In this work, we propose an alternative approach. In this work we introduce a privacy mechanism based on *limited leakage transformations* which have two key properties:

1. Individual transformed vectors are uninformative about their preimage; and

2. The comparison of transformed data points can provide information about the similarity of their preimages, but only if they are sufficiently close; the comparison provides no information about them otherwise.

We use tools from information theory to state theoretical properties and describe how to use this kind of scheme in practical scenarios.

We study the implications of using our proposed method in distance-based machine learning, which is the family of algorithms that depend directly on distance computations, with the objective of developing privacy mechanisms that enable the use of such methods without revealing private data. We discuss how to perform both training and inference phases under a private setting. Our goal is to show that fast and private computations on the cloud are feasible and useful for this class of techniques. We present our progress in this research and future directions to be addressed.

Contents

List of Tables

List of Figures

Part I

Introduction

Chapter 1

Thesis Overview

1.1 Motivation

Cloud-based services have become popular as platforms for large storage and expensive computation, meaning on-demand supply of software and hardware at any negotiable level. These features are particularly useful for *Machine Learning as a Service*, where we can distinguish two parties: a *client*, who has *data* with which he wants to work, and a *service provider*, who has a *model* or a *platform* to work with the client's data and return a desirable outcome. This is summarized as figure 1.1 shows,

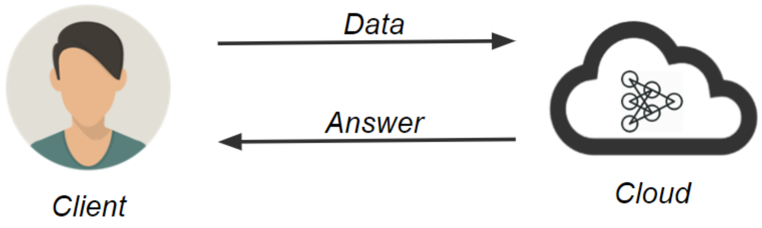


Figure 1.1: Client-Cloud setting. When we talk about *Machine Learning as a Service*, we consider two parties, a *client* and a *service provider*, who interact in order to obtain a desirable answer.

Under this setting, we can consider two Machine Learning tasks,

1. **Training models.**

   The growing success of Machine Learning is mainly due to having large volumes of data, more computational resources and better and more sophisticated algorithms. Hence, in different situations, it is much more convenient to store and process these large amounts of data on the cloud. For instance, having access to multiple computers to parallelize some procedures can be relevant for training models in a reasonable amount of time.

2. **Inference with trained models.**
   In many situations, machine learning applications such as face or speech recognition are deployed using a client-server model, where the client has the input data (image, speech, financial data, genetic code, etc.), and a remote server in the cloud has the corresponding machine learning model. There are several reasons for considering this model. First, in general the use of a machine learning model may require heavy computation and many parameters to control, making it more efficient to submit the client's data to a server than send a model to a client. The second reason is related to intellectual property constraints; the appropriate model and its parameters to perform a specific task is a key resource for the service provider. Thus, keeping the secrecy of the model is part of most business strategies.

However, despite the benefits that the cloud provides for addressing the challenges that appear in machine learning tasks, privacy issues relating to their use have become increasingly important. In fact, much of the data submitted and stored on cloud platforms are private, belonging to individuals or institutions who often desire to utilize the facilities provided by these platforms, but, at the same time, do not desire to expose their data to the platform itself.

For example, in situations that involve health data, the privacy issues that emerge are quite evident. We can consider the case where a patient wants to know the probability of having a specific type of cancer. For this, their doctors suggest using a machine learning model that uses the patient's genetic information as input. Although, this model can be very accurate in its estimation, it is clear that the patient is exposed to various problems that affect their privacy in case the model is deployed in the cloud. In effect, not only the genetic information of the patient will be exposed and potentially at risk, but also the outcome of the model to predict cancer. In this case, it is desirable to protect both the input and the output of the model.

Similar problems appear in the financial sector. In this case the input data may be associated with information regarding income, loans or debts of people or institutions. A model could take these data and generate some useful outcome for some organization. For example, a bank could use this type of data to predict the risk of a customer not paying a loan. If the bank makes use of the cloud to train a model to predict this risk, then the bank could put in risk information that is very sensitive for its customers as well as its business. Indeed, both the customers data and the trained model would be exposed to the cloud, being potentially compromised.

In general, if we need to upload data to the cloud to train a machine learning model, besides exposing the data, the final model will be totally exposed, being a risk if the platform is susceptible to be compromised. Similarly, in case we need to use an external machine learning service, similar concerns appear. Even though the service provider has a privacy policy, the client's data is usually stored in an external repository that may be compromised. In addition, the service provider may disclose the data along with the outcome of the model.

Therefore, it is very important to have a mechanism to protect the data from the risks we have discussed. An obvious solution is to encrypt the data prior to the submission and storage on the cloud. However, this causes problems in case we want to perform any computation on the data, and in particular, when we want to make any comparison between data instances.

This is particularly important given the fact that many machine learning algorithms are based on the comparison of examples of a data set, or the comparison between examples and an array of parameters. More specifically, many machine learning algorithms make extensive use of distance calculations between instances. We will call this type of algorithms simply *distance-based machine learning algorithms*, which will be the focus of this work. In consequence, there is a need to "*hide*" data, while permitting to compute distances between instances.

1.2 Thesis Statement

With the previous motivation, we proposed a privacy-preserving framework for distance-based machine learning algorithms that allows to process data without revealing it. We focus on the two tasks described above: training and inference. In the former, we assume that the client has data and wants to train a machine learning model using a cloud platform. The privacy constraints require that the cloud provider should not observe the data and not get any relevant information about the resulting model. In the second setting, we assume that the client has the data and the server has a trained machine learning model to be used. In this case, based on the privacy requirements, the server should not observe the data and the client should not observe the model.

The main objective of this thesis is to show that:

Fast and private computations on the cloud
are feasible and useful for distance-based machine learning tasks.

To support this statement, we introduce tools from Information Theory to propose a framework for privacy preservation, setting a tunable and user designed trade-off between privacy and utility. The presented work attempts to contribute to the field of Privacy-preserving Machine Learning, with a specific focus on distance-based methods, such as Clustering, Nearest Neighbors Classification, Support Vector Machines, among many others.

1.3 Summary of Contributions

To the best of our knowledge, this is the first work to make a connection between data comparison and privacy requirements using an information theoretical approach.

The technical contributions of this thesis are the following:

1. The introduction of a mathematical framework to study the *information leakage* in sample comparisons; a concept that is properly defined in this work.

2. The presentation of different theorems that allow understanding and controlling information leakage.

3. The relationship between transformations that satisfy our privacy requirements with distance-based machine learning methods.

1.4 Thesis Organization

This work is organized as follows,

- In chapter 2, we make an overview about the current privacy preserving technologies, showing their advantages and drawbacks.

- In chapter 3, we present different privacy leakage measurements that have been introduced, discussing different points of view.

- In chapter 4, we introduce the main concepts of this thesis. We present the definition of Limited Leakage Transformation along with some theoretical results, discussing the scope of these as a mechanism to enhance privacy in the cloud. Some of these ideas have been published as [1], [2].

- In chapter 5, we discuss how to estimate distances between points after transforming the data using the mechanism shown in the previous chapter. We present theoretical results and we analyse practical implications. Part of this work has been published as, [1], [3].

- In chapter 6, we analyse how to train distance-based machine learning models after applying the privacy enhancing transformation we have presented. One of the most important results is one that relates the transformed data to Kernel methods. In particular we show that with this kind of transformation we can change a non-linear method to a linear one with minimal loss of performance. Part of this work has been published as, [4], [5]

- In chapter 7, we start discussing potential applications that the introduced technique may have. In this chapter we study the case of biometrics protections, with a focus on speech signatures, comparing our method with some current solutions and applications. Part of this work has been published as, [2], [6].

- In chapter 8, we analyze the potential application of this technique in private image retrieval. In this case, we need to retrieve images from providing an image query. We would like to obtain the most similar image in a dataset. Since the image may contain sensitive information, it is necessary to hide the content of the query and the images in the data set. We discuss how our method helps to solve this problem and some current implementations that have used our method.

- In chapter 9, we present an application on two party computation. Here, there are two parties, each has a real valued vector and they want to compute the distance between them without revealing their corresponding vectors to each other. We present how our method may help to solve this problem or reduce the complexity in case of dealing with high dimensional data. Part of this work has been published as [3]

- In chapter 10, we present our conclusions and in chapter 11 we comment on lines for future work.

Chapter 2

Privacy Preserving Technologies

In this chapter we present some solutions that have been proposed to address the privacy concerns that arise in the setting we presented in the previous chapter. In general, there are two kind of solutions; one group based on cryptographic methods, where the security relies on the unfeasibility of solving some particular problems, while the second group of solutions is based on information theoretic guarantees, where the analysis of the security is unrestricted by the computational abilities that an adversary may have.

2.1 Cryptographic Solutions

In this section we give an overview of some tools based on cryptographic methods. In general, these methods achieve a high level of security, but are difficult to scale.

2.1.1 Homomorphic Encryption

Homomorphic Encryption schemes are a special type of cryptosystems that allow for operations to be performed on ciphertexts without requiring knowledge of the corresponding plaintexts. Hence, the structure of the plaintext space is preserved in the ciphertext space for additions and/or multiplications of plaintext data under encryption. Thus, Homomorphic Encryption enables operations on encrypted data without any decryption.

More formally, if two data instances x and z are encrypted to $E(x)$ and $E(z)$ using a homomorphic encryption scheme, we can obtain the encryption of the result of an operation $\otimes$ performed on x and z by performing another, possibly the same, operation $\oplus$ directly on the encrypted versions of x and z,

$$E(x \otimes z) \quad = \quad E(x) \oplus E(z) \tag{2.1}$$

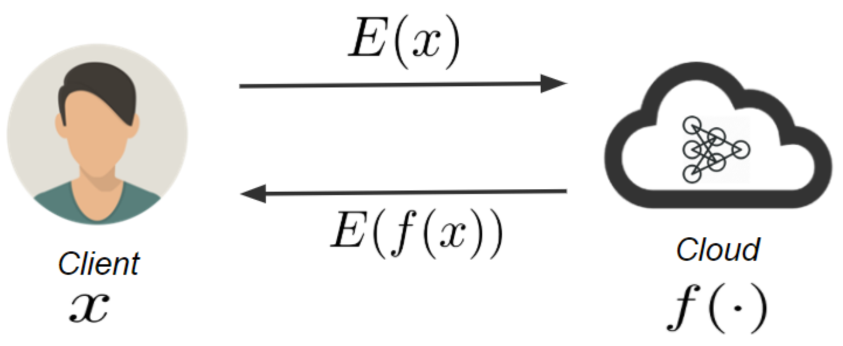


Figure 2.1: Homomorphic Encryption applied in the Client-Server setting.

Using HE, a data owner can encrypt their data with the public key, send it to a service provider that has a model and has no access to the secret key, and receive the required answer in encrypted form, which only the client can decrypt with its secret key.

We can distinguish three types of Homomorphic Encryption schemes:

1. Partially Homomorphic Encryption (PHE)
2. Somewhat Homomorphic Encryption (SHE)
3. Fully Homomorphic Encryption (FHE)

FHE allows unlimited additions and multiplications at the cost of an increased computational load [46], while SHE schemes have a fixed limit of multiplications to speed up their execution. PHE schemes support either additions or multiplications, hence, they are only partially homomorphic.

The problem of SHE schemes is that the resulting ciphertext cannot be decrypted when the limit of multiplications is exceeded. Furthermore, there are SHE schemes that cannot correctly decrypt when different operations are combined, i.e., only additions or multiplications are possible but combining both operations in the encrypted domain cannot be handled.

Some examples of Homomorphic Encryption schemes include Paillier [47], ElGamal [48], and NTRU [49] cryptosystems. In general, the drawback of these schemes is a relatively high computational overhead and the relatively large ciphertexts and keys. An extensive overview of different Homomophic Encryption Schemes can be found in [50].

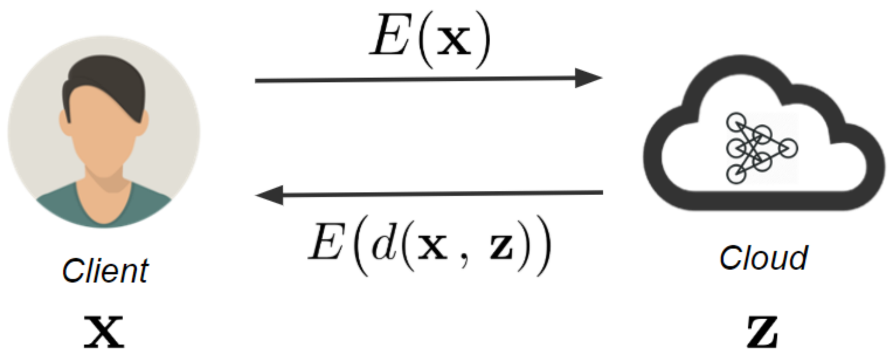


Figure 2.2: Homomorphic Encryption can be applied for private distance computation.

Distance Computation using Homomorphic Encryption

Given that the focus of this work is on distance-based methods, it is worth understanding how this type of technique is applied to calculate distances in the encrypted domain. To illustrate how this technique works, we will consider any Partial Additive Homomorphic Encryption, as Paillier, denoting the encryption function as E. These systems satisfy the following properties for any pair of integers x and z,

$$E(x) \cdot E(z) = E(x + z) \tag{2.2}$$

$$(E(x))^z = E(x \cdot z) \tag{2.3}$$

Hamming distance

We can suppose that $\mathbf{x}$ and $\mathbf{z}$ are vectors in $\{0, 1\}^N$. Let's assume that data owner has the vector $\mathbf{x}$ while the service provider has the vector $\mathbf{z}$. The data owner wants the value of the Hamming distance between these vectors without exposing the vector $\mathbf{x}$ to the service provider. The Hamming distance between binary vectors is just the number of positions at which the corresponding components are different.

On the other hand, the service provider is willing to expose the hamming distance, but not the entire vector $\mathbf{z}$. Note that if the Hamming distance is zero, then, the data owner will know that the service provider's vector is equal to its vector, so will have total knowledge about the service provider's data. So, assume that the service provider does not have any problem about this issue, this means, he or she is willing to expose just the information that the Hamming distance provides.

We can use PHE to solve this problem. Indeed, we can assume that both the data owner and the service provider have a public key of a partial homomorphic encryption scheme, but only the data owner has the secret key.

Then, first the data owner encrypts each of the components of its vector, obtaining $E(x_1),\ E(x_2), ..., E(x_N)$, and sends these values to the service provider.

To compute the Hamming distance with the encrypted data, the service provider performs the following computation

$$\prod_{i=1}^{N} E(x_i) \cdot E(z_i) \cdot E(x_i)^{-2z_i} = \prod_{i=1}^{N} E(x_i + z_i - 2x_i z_i) \tag{2.4}$$

$$= E\left(\sum_{i=1}^{N} x_i + z_i - 2x_i z_i\right) \tag{2.5}$$

This equality is obtained using the basic properties that appear in Partial Additive HE. But, since we are dealing with binary data, it is easy to realize that the expression $x_i+z_i-2x_iz_i$ is 0 if $x_i = z_i$, and 1 otherwise. Therefore, the expression computed by the service provider is in fact the encrypted version of the Hamming distance. Note also that the service provider cannot obtain any information about the client's data.

Thus, the service provider sends back the value obtained after computing 2.5 to the data owner, who can decrypt this message using its secret key.

Note that this method does not work when both vectors are encrypted. However, using a fully homomorphic encryption scheme, this task can be easily solved.

Euclidean distance

A very similar approach can be considered for the Euclidean distance case. We can consider the same setting as before, where the data owner has the vector $\mathbf{x}$, while the service provider has the vector $\mathbf{z}$, and the data owner wants to know the squared Euclidean distance between these vectors under the same privacy constraints.

In this case, the data owner has to encrypt the components of its vector along with the sum of the square of each component. Hence, the service provider receives $E(x_1)$, $E(x_2), ..., E(x_N)$ and $E\left(\sum_{i=1}^{N} x_i^2\right)$. Then, the service provider, using its vector and the public key, can compute the following expression,

$$E\left(\sum_{i=1}^{N} x_i^2\right) \cdot E\left(\sum_{i=1}^{N} z_i^2\right) \cdot \prod_{i=1}^{N} E(x_i)^{-2z_i} = E\left(\sum_{i=1}^{N} x_i^2 + z_i^2 - 2x_i z_i\right) \tag{2.6}$$

$$= E\left(\|\mathbf{x} - \mathbf{z}\|^2\right) \tag{2.7}$$

Then, the service provider sends these values to the client and she or he can decrypt the message using its secret key, obtaining the value of the squared Euclidean distance.

More example about how to apply HE to different distance computations can be found in [54]

$\hat{\wedge}$	$b = 0$	$b = 1$
$a = 0$	$E_{k_0^0}\left(E_{k_0^1}\left(k_0^2\right)\right)$	$E_{k_0^0}\left(E_{k_1^1}\left(k_0^2\right)\right)$
$a = 1$	$E_{k_1^0}\left(E_{k_0^1}\left(k_0^2\right)\right)$	$E_{k_1^0}\left(E_{k_1^1}\left(k_1^2\right)\right)$

Table 2.1: Truth table for garbled AND gate

2.1.2 Secure Two-Party Computation

Secure Multiparty Computation is a sub-field of cryptography that concerns with settings in which n parties are interested in jointly computing a function f using their input data while maintaining the privacy of their inputs. When $n = 2$, we refer as Secure Two-Party Computation (STPC).

Privacy in these kinds of protocols is established by the constraint that no party should learn anything apart from the output and the intermediate steps of the computation. Thus, in Secure Two-Party Computation we consider that each party has inputs x_1 and x_2 respectively, and they need to compute the output $f(x_1, x_2) = (y_1\ , y_2)$. Under this setting, the privacy constraints require that each party provides the value x_i and obtains the outcome y_i, as well as does not observe the input or output of any other party.

Here, the function f is usually represented as a circuit, this is, a directed acyclic graph where the edges represent intermediate wires, and the nodes show input and output wires, and gates that compute a basic function; consider, for example, Boolean functions where the gates consist in AND and XOR gates. For the secure evaluation of f, parties rely on securely computing every gate in the function's circuit.

The most well known example is given by the Yao's garbled circuits protocol [55], which has been subject to many optimisations and used in different applications. Garbled Circuits operate on binary inputs and compute a function by evaluating its garbled Boolean circuit representation.

In this protocol we consider that one party will garble the circuit and the other will evaluate it. The former is called the garbler whereas the latter is called the evaluator. First, for each wire w, the garbler creates random labels k_0^w and $k_1^w \in \{0,1\}^k$, where k is the security parameter, for example, $k = 128$. Then, for each gate g, the garbler generates a garbled gate $\hat{g}$ such that, given $\hat{g}$ and the labels corresponding to the values of the input wires of g, the evaluator can compute the label corresponding to the correct value of the output wire of g. This construction makes use of a symmetric encryption scheme. As example, we can consider an AND gate. If $E_k(x)$ denoted the encrypted version of x using the key k, table 2.1 shows the truth table of the garbled AND gate.

In this example k_0^0 and k_1^0 denotes the labels corresponding to the wire related to a for the values 0 and 1 respectively. Similarly, k_0^1 and k_1^1 are the labels corresponding to the wire related to b for the values 0 and 1 respectively, and k_0^1 and k_1^1 are the labels corresponding to output wire for the values 0 and 1 respectively.

After garbling each gate in the circuit, the garbler randomly permutes the entries of each garbled gate and sends the resulting garbled circuit to the evaluator. A final task is that, in order to enable the evaluation of the garbled circuit, the garbler has to transfer the labels corresponding to the input bits of both parties without revealing one party's input to the other. Since the labels are chosen uniformly at random, just sending the labels corresponding to the garbler's inputs to the evaluator reveals nothing about the actual input. However, sending the labels for the input of the evaluator is problematic, as the garbler must not know which labels to send but also cannot send both labels for each bit because then the evaluator could learn information about the garbler's input by evaluating the circuit on other inputs than its own. The solution to this problem is a cryptographic primitive called Oblivious Transfer (OT). Such an OT allows a sending party with the input $(x_0\, x_1)$ to send the value x_b to a receiving party with input bit $b \in \{0\,,\,1\}$ without learning anything about b and without revealing anything about x_{1-b} to the receiver. In order to send the evaluator its corresponding labels, both parties engage in an OT protocol for every input bit y_j of the evaluator's input y, with the garbler acting as sender with input $(k_0^j\,,\,k_1^j)$ and the evaluator acting as receiver with input y_j.

We can summarize the Yao's GC protocol in the following steps:

1. The garbler transforms f into a Boolean circuit consisting of XOR and AND gates. It generates labels for all wires and garbled gates for every gate in the circuit.

2. The garbler sends the permuted garbled gates and the labels corresponding to its own input bits to the evaluator. Then, both parties engage in OTs, where the evaluator obliviously receives the labels corresponding to its input bits.

3. The evaluator evaluates each gate in the garbled circuit using the labels obtained in step 2.

4. To reveal the output, the garbler can reveal the plaintext bits corresponding to the circuit output labels obtained in step 3.

Compared to Homomorphic Encryption, most of the Secure Two Party Computation protocols rely on relatively cheap symmetric cryptographic operations, for example AES, but STPC requires interaction per gate between the parties, while HE does this only for the inputs and outputs.

An overview of some frameworks that implement STPC protocols can be found in [75].

SMC in Machine Learning Systems

Many efforts have been done to preserve privacy in machine learning algorithms using Secure Multiparty protocols, e.g. in clustering [72], nearest neighbor classification, support vector machine, neural networks[74] and many other techniques [73].

However, most of these works focus on the inference task based on a client-server model and generally have problems with the scaling up the computation. For example, in [72] the communication cost increases linearly in the number of points involved and the dimensionality of the data.

2.2 Information Theoretic Privacy

As mentioned earlier, another class of privacy preserving methods, referred to as information theoretical privacy mechanisms, are based on information theory tools. The main advantage of these techniques is that they provide privacy against adversaries with unbounded computational power.

2.2.1 Differential Privacy

A data set $\mathcal{D}$ is a collection of elements. A randomised *query mechanism* $\mathcal{M}$ produces a response when performed on a given data set. Two data sets $\mathcal{D}$ and $\mathcal{D}'$ are said to be adjacent if they differ by at most one element. There are two proposed definitions for adjacent data sets. The stronger one is based on deletion: $\mathcal{D}'$ contains one entry less than $\mathcal{D}$. The weaker one is based on substitution: one entry of $\mathcal{D}'$ differs in value from that in $\mathcal{D}$.

The query mechanism $\mathcal{M}$ is said to satisfy *differential privacy* [51] if the probability of $\mathcal{M}$ resulting in a solution S when performed on a data set $\mathcal{D}$ is very close to the probability of $\mathcal{M}$ resulting in the same solution S when executed on an adjacent data set $\mathcal{D}'$. Formally, we say that a randomised function $\mathcal{M}$ satisfies $\varepsilon-$differential privacy if for all adjacent data sets $\mathcal{D}$ and $\mathcal{D}'$ and for any $S \in \text{range}(\mathcal{M})$,

$$\mathbb{P}\left[\mathcal{M}(\mathcal{D}) = S\right] \quad \leq \quad \exp(\varepsilon)\, \mathbb{P}\left[\mathcal{M}(\mathcal{D}') = S\right] \tag{2.8}$$

The value of the ε parameter is referred to as *leakage* and determines the degree of privacy. A smaller ε represents a stronger privacy. In practice, ε is set less than 1 (e.g., 0.1 or ln(2)).

As a consequence of this model, we may consider that if an individual chooses to contribute to the data set, there is a little or no increase in privacy risk for the individual as compared to not choosing to contribute to the data set. With the aim of designing mechanisms that satisfy differential privacy, two composition theorems are used:

Theorem 2.2.1 ***(Sequential Composition)****. If a set of mechanisms $\{\mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_n\}$ is sequentially performed on a data set, and each $\mathcal{M}_i$ provides ε_i-differential privacy, then the entire mechanism will provide $(\sum_{i=1}^{n} \varepsilon_i)$-differential privacy.*

Theorem 2.2.2 ***(Parallel Composition).*** *If $\{\mathcal{D}_1, ..., \mathcal{D}_n\}$ is a partition of the data set and $\mathcal{M}_i$ is a ε_i-differential privacy mechanism applied on $\mathcal{D}_i$, then the mechanism that applies the $\mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_n$ in the corresponding sets $\mathcal{D}_1, ..., \mathcal{D}_n$ is a $(\max\{\varepsilon_1, ..., \varepsilon_n\})$-differential privacy mechanism.*

These two theorems help to control the degradation of privacy when we need to compose several differentially private mechanisms. One way to achieve differential privacy is through the *exponential mechanism* [62] for releasing continues-valued functions. Given a function f to be evaluated over the data set $\mathcal{D}$, we need to add a perturbation to the value of $f(\mathcal{D})$ to prevent leakage. To do this, the mechanism $\mathcal{M}$ adds the appropriate perturbation η, such that $f(\mathcal{D}) + \eta$ satisfies differential privacy. The distribution of η is determined by the sensitivity of f on the data set $\mathcal{D}$. This is the maximum difference between $f(\mathcal{D})$ and $f(\mathcal{D}')$ where $\mathcal{D}'$ is an adjacent data set. Formally, the sensitivity S of f is given by:

$$S = \max_{\mathcal{D},\mathcal{D}'} \|f(\mathcal{D}) - f(\mathcal{D}')\|_1 \tag{2.9}$$

Hence, the sensitivity S of the function f indicates how much the function is likely to change after changing one instance from the data set. It is possible to prove that if η is sampled from Laplace(S/ε), then the mechanism $f(\mathcal{D}) + \eta$ satisfies ε-differential privacy [62]. Note that the perturbation η introduces an error with respect to the true value $f(\mathcal{D})$ which is inversely proportional to ε. This implies a trade-off between privacy and utility. Differential privacy has become an important research field; for a general overview and further details see [52].

Differential Privacy in Machine Learning

In machine learning systems, the query mechanism can be thought of as an algorithm learning the classification rule which is evaluated over the training data set. The output of an algorithm satisfying differential privacy is likely to be the same when the value of any single data set instance is modified, and therefore, no additional information can be obtained about any individual training instances with certainty by observing the output of the learning algorithm.

In addition, the ε-differential privacy model limits the information that an adversary can gain about a particular private value by observing an entire function learned from a data set containing that value, even if she or he knows every other value in the database. Naturally, if we want to apply this method to machine learning problem we need to be aware of the trade-off between privacy and learnability.

Most of the training methods for a recognition system are based on optimisation problems. In practice, there are two strategies for inserting differential privacy on the learning process of a recognition system. The first type adds noise to the execution process of the corresponding optimisation algorithm. The second type makes a perturbation directly to the objective function, typically adding a differentially private noise before the learning procedure. Both strategies have been explored in different models, which we will now briefly mention.

1. *Large Margin Classifiers.* In [63] is presented an algorithm for learning a discriminatively trained multiclass Gaussian mixture model-based classifier that preserves differential privacy using a large margin loss function. The solution involves adding a perturbation term to the objective function. The authors show that differential privacy is satisfied and they establish a bound on the excess risk of the classifier learned which is directly proportional to the number of classes and inversely proportional to the privacy parameter reflecting a trade-off between privacy and utility.

2. *Multi-Party Classifiers with Differential Privacy.* In many cases it is common to have the problem of learning a classifier from a multi-party collection of private data. The goal is to learn a classifier from the union of all the data. In [64], the imposed conditions are that: (a) None of the parties are willing to share the data with one another or with any third party (e.g., a curator). (b) The computed classifier cannot be reverse engineered to learn about any individual data instance possessed by any contributing party

3. *Differentially Private Deep Learning.* Deep learning has been one of the most successful techniques in machine learning and signal processing. The basic idea is to apply a multiple-layer structure to extract complex features from high-dimensional data and use them to build models. Each layer has a set of parameters that must be learned from training samples, minimizing a a loss function, which depends on these parameters. In practice, stochastic gradient descent (SGD) methods are used to achieve this objective. As previously mentioned, the knowledge of the model or its use on particular samples can leak information about the training data, which results in privacy risks. To deal with this issue, deep learning models can be adapted to reach differential privacy. In [65] a differentially private SGD algorithm is designed by introducing a sparse vector technique. Similarly, in [66] a differentially private SGD algorithm is designed relying on a Gaussian Mechanism [52]. On the other hand, in [67] a perturbation on the objective function of a deep auto-encoder is considered to achieve differential privacy.

Note that differential privacy solves a different privacy problem from those that Homomorphic Encryption and Secure Multiparty Computation solve. While Homomorphic Encryption and Secure Two-Party Computation solve the problem of private inference, these techniques do not deal with the problem of potential leakage of information using the system responses. In fact, an attacker may try to use the responses of the system to infer model parameters and eventually learn information about training samples. In this context, differential privacy can be added to prevent those types of attack.

In addition, differential privacy provides the relative guarantee that the release of information will be just as likely whether or not the data about an individual is present in the database. As a consequence, if an individual chooses to contribute to the database there is little or no increase in the privacy risk of the individual as opposed to not contribute to the database.

Local Differential Privacy

The general idea of differential privacy works under the notion of privacy in databases. Thus, all these constructions make sense at the moment of releasing a global statistic or a model from the data set.

However, there are some situations where data belonging to particular individuals must be released to untrusted parties. To address this problem, people have proposed the notion of local differential privacy.

Definition 2.2.1 *Let ε be a positive real number and $\mathcal{A}$ be a randomized algorithm that takes a user's private data as input. The algorithm $\mathcal{A}$ is said to provide ε-**local differential privacy** if, for all pairs of user's possible private data x and x' and all subsets S of the range of $\mathcal{A}$ we have,*

$$\mathbb{P}\Big(\mathcal{A}(x) \in S\Big) \quad \leq \quad e^{\varepsilon} \cdot \mathbb{P}\Big(\mathcal{A}(x') \in S\Big) \tag{2.10}$$

where the probability is taken over the randomness used by the algorithm.

The main difference between this definition and the standard definition of Differential Privacy is that in differential privacy the probabilities are of the outputs of an algorithm that takes all users' data, while an ε-local differential privacy algorithm takes as input a single user's data.

This approach is much convenient for practical purposes, in case a data set comes from different parties, for example, smart phone devices, the randomization can be done locally.

The main disadvantage of this method is the fact that the randomization is strictly related to the algorithm we want to compute. In addition, the privacy guarantees are stated for the final outcome and not for the user's data itself. In fact, most of the local differential privacy methods are based on adding a particular type of noise to the user's data. Thus, this randomized observation may give personal information.

Part II

An Information Theoretical Approach to Limit Leakage in Comparisons

Chapter 3

Privacy Leakage Measurements

In this chapter, we will discuss different ways to quantify the privacy leakage after processing data. For this, we will use information theory tools, which will allow us to formalize our problem and analyze the privacy requirements regardless of the computational capacities that an adversary may have. Indeed, information theory helps to understand fundamental limits of a *hiding mechanism*, guiding the development of techniques to address new challenges related to privacy concerns in cloud computing settings.

3.1 Preliminaries

In general, we will understand that a privacy protection mechanism takes a piece of data X, and transforms it into a new random variable Y. The variable X can refer to a data instance or a complete set of data. Thus, we are interested in analyzing the statistical relationships between X and Y, in particular we are interested in studying the information that we can know about X by observing Y. This motivates the following definition.

Definition 3.1.1 *Let X and Y be two random variables. The* ***Information Density*** *at the point (x, y) is given by*

$$i(x\,;\,y) \quad = \quad \log \frac{\mathbb{P}(X = x|Y = y)}{\mathbb{P}(X = x)} \tag{3.1}$$

Thus, the information density in the point (x, y) compares the prior probability of obtaining x as an instance of X, and the probability of obtaining x for X under the knowledge that $Y = y$. In other words, it is a direct comparison between a prior and a posterior. If this number is very close to 0, it means that the knowledge of $Y = y$ contributes little information about whether $X = x$. On the other hand, if the information density is far from 0, then it means that the knowledge that $Y = y$ reveals relevant information about whether $X = x$ or not.

Note that, by Bayes's rule, we can obtain easily the symmetry property, that is $i(x,y) = i(y,x)$. Therefore, the same measure works to quantify how much information the knowledge of $X = x$ reveals about whether $Y = y$ or not.

In order to obtain a global measure about the leakage that Y produces on X one can try to combine the information density at different values of (x,y). Within the different ways to do it, the simplest is to take the average weighted by the probability of obtaining (x,y). As a result, we get a measure that is commonly used in Information Theory.

Definition 3.1.2 *The* ***Mutual Information*** *[56] between two random variables X and Y is*

$$\begin{aligned} I(X\,;\,Y) &= \sum_{x,y} \mathbb{P}(X=x\,,\,Y=y)\log\frac{\mathbb{P}(X=x\,|\,Y=y)}{\mathbb{P}(X=x)} && (3.2)\\ &= \sum_{x,y} \mathbb{P}(X=x\,,\,Y=y)\,i(x,y) && (3.3)\end{aligned}$$

The mutual information is just the average of the Information Density through all possible values of X and Y. Some of the well known properties of the mutual information are the following,

1. *Non-negativity.* For any pair of discrete random variables X and Y, we always have

$$I(X\,;\,Y) \geq 0 \tag{3.4}$$

2. *Symmetry.* The mutual information between X and Y is equal to the mutual information between Y and X.

$$I(X\,;\,Y) = I(Y\,;\,X) \tag{3.5}$$

3. *Relation to the entropy.* The mutual information between two random variables can be written as a difference between two entropies; the entropy of X and the conditional entropy of X given Y. In other words, the mutual information can be seen as a measure of reduction of entropy of one random variable under the knowledge of the other one.

$$I(X\,;\,Y) = H(X) - H(X|Y) \tag{3.6}$$

4. *Measure of independence.* X and Y are independent if and only if $I(X\,;\,Y) = 0$. Thus, if the mutual information is zero, any knowledge about one random variable is irrelevant to knowledge about the other random variable.

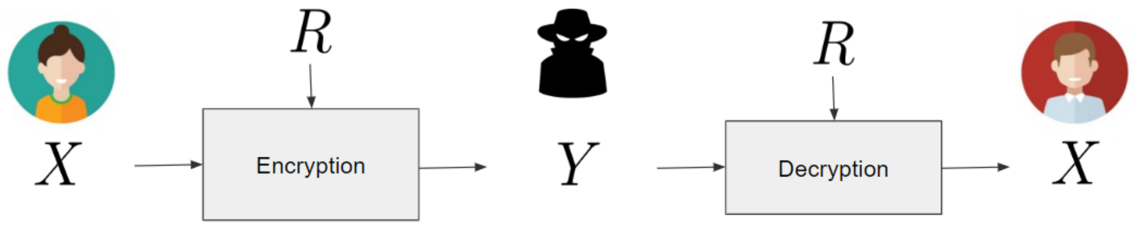


Figure 3.1: Common setting in encryption schemes. Two parties share a key that they use to communicate securely.

3.2 Information Secrecy

The mutual information has been used to analyze different problems both in signal processing and communication theory. In this section, we will see its connection to preserving secrecy in cryptographic settings. Here, we assume the well established approach, where there are two parties, Alice and Bob, who want to communicate each other over a untrusted channel. We assume that Alice has a message X to be shared with Bob. In order to prevent that an eavesdropper may observe the message, Alice decides to *encrypt* her message. To do so, Alice and Bob have already shared a random secret key R, so Alice uses it, combined with X, to produce the message Y. To ensure the secrecy of the message, an eavesdropper should not gain any information about X from observing Y without the knowledge of R. Hence, a cryptosystem can be described by the triplet (X, R, Y), where the random variable X represents the messages that Alice can send (assuming that a message comes from a particular distribution), R is the random variable that describes the key generation, and Y the possible encrypted messages. Figure 3.1 illustrates this setting.

The security of this setting can be formalized in information theoretical terms, motivating the following definition.

Definition 3.2.1 *A cryptosytem (X, R, Y) is an* ***Error-free perfect secrecy*** *[57] system if*

$$I(X\,;\,Y) = 0 \tag{3.7}$$

$$H(X\,|\,R, Y) = 0 \tag{3.8}$$

$$I(X\,;\,R) = 0 \tag{3.9}$$

The first equation of this definition states that X and Y must not be related, ensuring perfect secrecy. The second equation says that, under the knowledge of R and Y, there is no entropy corresponding to X, which is a condition for decryption. The last equation states that the message X and the secret key R are not related, which is just a guarantee for practical purposes.

Hence, in a system with perfect secrecy, an adversary who does not have the key cannot find any statistical relationship between a plaintext X and its ciphertext Y. Nevertheless, if the adversary has the cipher text and the key, there is no uncertainty about the plaintext.

However, in order to achieve perfect secrecy, it is possible to prove [109] that the support of the random variable X must contain fewer elements than the support of R. In other words, the key space must be larger than the plaintext space. This, of course, has important consequences for practical purposes, since the possible messages that Alice can send depends on how large the key space is. This fact has been extensively studied both in Information Theory and Cryptography. To guarantee a level of security in more practical conditions, different relaxations to perfect privacy have been proposed.

Definition 3.2.2 *A cryptosystem (X, R, Y) satisfies **Strong Secrecy** [58] [59] with leakage ξ if*

$$I(X\,;\,Y) \quad \leq \quad \xi \tag{3.10}$$

Of course, it is desirable that the value of ξ is as small as possible to obtain a high level of security. Note that due to the constrains that appear in cryptosystems with perfect secrecy, we can claim that most of the current encryption schemes satisfy strong secrecy with some value of ξ. We can also say that strong secrecy is a necessary condition to obtain a secure cryptosystem.

The idea of controlling the mutual information between X and Y has been also considered for designing privacy mechanisms. One example is known as the rate-distortion approach, where attempts to constrain or optmize a utility function such as an empirical loss $\hat{R}_X(Y) = \frac{1}{n}\sum_{i=1}^{n} \mathrm{loss}_{X_i}(Y)$. The goal is to minimize the mutual information between Y and X while ensuring some amount of utility, i.e.

$$\min_{\mathbb{P}(Y|X)} I(X\,;\,Y) \quad \text{s.t.} \quad \mathbb{E}\left[\hat{R}_X(Y)\right] \leq \gamma \tag{3.11}$$

To find the rate-distortion function $\mathbb{P}(\,\cdot\,|X)$, the Blahut-Arimoto algorithm [60] [61] is used which starting from some initial $\mathbb{P}(Y)$ iteratively updates

$$\mathbb{P}(Y|X) \quad \propto \quad \mathbb{P}(Y)e^{-\beta \hat{R}_X(Y)} \tag{3.12}$$

3.3 Comparison of Transformed Messages

One of the aspects that is not addressed in the traditional framework is what happens when we have multiple encrypted messages. An adversary may not be interested in learning the relation between the plaintext and the ciphertext. It may just want to find statistical dependencies between different ciphertexts, because the relationship between ciphertexts may tell them something about the relation between the corresponding plaintext.

To limit the adversary's learning capabilities, we should control the leakage between comparisons, that is, for any $i, j, i \neq j$, we would like to make $I(Y_i\,;\,Y_j)$ as small as possible.

One important observation is that this is not a necessary condition for the other properties. Indeed, even under a strong property as Perfect Secrecy we may not have a guarantee to protect data under relationship learning. Just to illustrate this, we can take the example of *One-time Pad with reused key.*

Example 3.3.1 *Let X_1 and X_2 be elements in $\{0,1\}$. We can select as secret key an element $R \in \{0,1\}$ uniformly random. With this, we can encrypt these values as $Y_1 = X_1 \oplus R$ and $Y_2 = X_2 \oplus R$. In this case, we have Perfect Secrecy in terms of (X_i, Y_i); in fact, $I(X_i\,;\,Y_i) = 0$. However, since $Y_1 \oplus Y_2 = X_1 \oplus X_2$, we can conclude that, if $X_1 \oplus X_2 = 1$, then $I(Y_1\,;\,Y_2) = 1$. In the same way, if $X_1 \oplus X_2 = 0$, then $I(Y_1\,;\,Y_2) = 1$.*

Our goal is to allow learning relationships, but only if the plaintext satisfies some specific conditions. In particular, the data owner wants to use the cloud to perform some particular computation, but she or he is not interested in recovering the original data once it is uploaded to the cloud. We also assume that she or he is willing to leak some information in exchange for making computation simpler.

Thus, we have two research questions to explore:

1. How to control the information leakage?
2. What kind of computations can be done?

We need a flexible framework that allows us to control the trade-off between privacy and utility. In the next chapter we try to address these questions.

Chapter 4

Limited Leakage Transformations

In this chapter, we will introduce the concept around which our work revolves; *Limited Leakage Transformations.* The objective in our case is to transform data points so that when comparing transformed values, they can reveal information if and only if the original points are close enough; if the data points are far enough, negligible information is leaked. To do this, we will give a general definition of what we understand by a Limited Leakage Transformation, and then analyze particular constructions of these kinds of functions, analyzing their practical implications.

To facilitate the reading, it was decided to exclude from this chapter the proofs of all the theorems and propositions, and to present them in the appendix of this work.

4.1 General Assumptions

Throughout this chapter, we will assume that there is an individual, who we will refer to as *Data owner*, that has a finite collection of data points. We assume these are elements in a metric space $\mathcal{M}$, provided with a metric d, which satisfies the standard properties. The data owner wants to perform some computation on its data using the cloud.

We will also assume there is an adversary who can observe the data located in cloud but he or she cannot make any modification to it; thus, we will not consider that the integrity of the data is an issue. The adversary's objective is essentially trying to get as much information as possible observing the data located in the cloud.

Since the data owner can consider that its data is sensitive, he or she decides to transform it in order to limit the leakage of information, but in some way, to continue allowing some comparison between her or his data instances. Since we are working with the metric space structure, it is natural to think that the type of calculations that the data owner wishes to make depends on the value of the metric between two points. Indeed, as we mentioned before, there are many algorithms of machine learning that depend on some measure of similarity between data instances, and ultimately, some measure of distance between different examples. Most of these algorithms make extensive use of distance whenever it is small enough, making cases where the distance is large, that is, when the similarity is small, have an insignificant effect, or even cases that the data owner wants to avoid.

In this way, the information useful for a distance-based machine learning algorithm is associated to those comparisons where the distance is small. This does not mean that, in general, the information revealed between data instances whose distance is large is not relevant for other tasks, but for purposes of the algorithm that the data owner wants to compute this information that is not useful, so it is desirable to hide it from a potential adversary. In other words, the data owner is willing to reveal to the cloud, or to a potential adversary, the information that is strictly necessary for the algorithm that he wishes to calculate. Thus, if the data owner wants to use the cloud to train an algorithm that depends on the distance between points, but only if it is small, then it can transform its data points in order to control the information leakage that is not associated with this condition.

Next, we will try to formalize these ideas, delivering a precise definition of transformations that allow to control the information leakage by comparison, and to establish different mathematical statements.

4.2 Controlling information leakage in data comparison

As mentioned earlier, we consider we are working with points in a metric space $\mathcal{M}$, and we are measuring the closeness between points with a metric d, which satisfies the standard properties of a metric, which are:

1. $d(\mathbf{x}_0\,,\,\mathbf{x}_1) = 0$ if and only if $\mathbf{x}_0 = \mathbf{x}_1$
2. For every $\mathbf{x}_0\,,\,\mathbf{x}_1 \in \mathcal{M}$ we have that $d(\mathbf{x}_0\,,\,\mathbf{x}_1) = d(\mathbf{x}_1\,,\,\mathbf{x}_0)$
3. For every $\mathbf{x}_0\,,\,\mathbf{x}_1\,,\,\mathbf{x}_2 \in \mathcal{M}$, we have that $d(\mathbf{x}_0\,,\,\mathbf{x}_1) \leq d(\mathbf{x}_0\,,\,\mathbf{x}_2) + d(\mathbf{x}_2\,,\,\mathbf{x}_1)$

It is worth saying that beyond the assumption of metric space, we are not considering any other restriction on the set $\mathcal{M}$, in particular, this could be an infinite set like $\mathbb{R}^N$ or the set of finite binary sequences, or a finite set such as $\{0,1\}^N$, to give some examples.

To prevent an adversary gaining information about our points in this space, we are going to transform the points to a space $\mathcal{E}$. In principle, we will not impose any metric structure on $\mathcal{E}$, but in the following chapters we will see how to take advantage of this to do certain operations in this space that reveal information about the corresponding points in the original space $\mathcal{M}$. For now, the only relevant assumption about $\mathcal{E}$ is that it is a finite set.

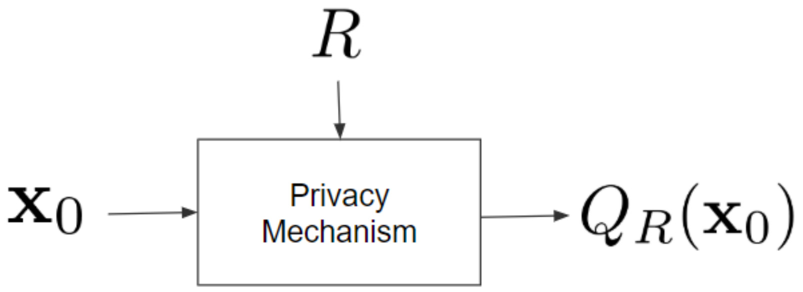


Figure 4.1: We assume that a privacy mechanism transform data instances $\mathbf{x}$ using a source of randomness R into an object $Q_R(\mathbf{x})$

In principle we would like to be able to design this transformation and make its design public. Since we can not control the randomness of the elements in $\mathcal{M}$, and as we said, we are assuming that they do not have any random components, we must introduce some source of randomness in order to avoid dictionary attacks by the adversary. Thus, similar to what is an encryption scheme, we will generate a secret key R, which for purposes of this model is a random variable that does not depend on any point of $\mathcal{M}$. Then, using this secret key R, we can compose a transformation Q_R that allows us to convert points in $\mathcal{M}$ into points in $\mathcal{E}$. Therefore, for any $\mathbf{x} \in \mathcal{M}$, R composes the random variable $Q_R(\mathbf{x})$ with support in $\mathcal{E}$; all the randomness of $Q_R(\mathbf{x})$ comes from R and $\mathbf{x}$ is considered as a fixed parameter.

This idea seems to be very similar to a standard encryption scheme, however, we have two key differences. First, since we want to enable operations in the cloud, we must allow some leakage that is associated with the proximity of the points to be hidden. This cannot be achieved with classic encryption models. Indeed, although the security of the models can be controlled, the potential leakage is uniform for all the points, and consequently, all the possible relationships that can be established. In addition, by design, standard encryption models do not allow inference of relationships with encrypted data. While we could do that using Homomorphic Encryption, the computational cost increases by several orders of magnitude.

The second important difference is the fact that in this setting we are not interested in recovering the original point after transforming it. Our only interest is to be able to perform some operations in the cloud. Part of this work is to show that we can make estimates of the closeness between points without necessarily preserving all information about them. What remains to be determined is what type of properties the random transformation Q_R must have in order to satisfy the requirements described above.

The first property that must be satisfied is the fact that the value of the random variable $Q_R(\mathbf{x})$ does not reveal any information about $\mathbf{x}$. This is formalized in the following definition.

Definition 4.2.1 *We say that the random transformation Q_R satisfies **indistinguishability** if for any $\mathbf{x}_0$,* *and* $\mathbf{x}_1 \in \mathcal{M}$*, the corresponding random variables* $Q_R(\mathbf{x}_0)$ *and* $Q_R(\mathbf{x}_1)$ *have the same probability distribution.*

In other words, the distribution of $Q_R(\mathbf{x})$ over the possible values in $\mathcal{E}$ does not depend on $\mathbf{x}$, and therefore, only observing realizations of $Q_R(\mathbf{x})$ we can not infer anything about $\mathbf{x}$. It should be noted that this does not mean that when observing realizations on other points you can not gain information about $\mathbf{x}$. Indeed. if we go back to the example of *One-time pad*, we can notice that if we consider $\mathcal{M} = \{0, 1\}$, and $R \sim \mathit{Bernoulli}(0.5)$, then One-time pad can be described as $Q_R(\mathbf{x}) = \mathbf{x} \oplus R$. In this case, Q_R satisfies indistinguishability, but we already know that by applying the same transformation on another point in $\mathcal{M}$ an adversary can infer information about $\mathbf{x}$.

Thus, the indistinguishability property seems to be a necessary condition, but it is far from being sufficient for the requirements we seek. In order to be able to limit the information leakage between instance comparisons, it is useful to consider the leakage measures discussed in the previous chapter, to limit the possible inferences that an adversary can make. In particular, we will use the mutual information between transformed points, making it negligible when the distance between the original points is large. Formally, we introduce the following definition.

Definition 4.2.2 *Consider that the random variable* R *composes a transformation* $Q_R : \mathcal{M} \to \mathcal{E}$*. We say that* Q_R *satisfies the* ***Limited Leakage Property*** *(LLP) with respect the metric* d*, if there exist positive constants* C *and* γ *such that for every* $\mathbf{x}_0$, $\mathbf{x}_1 \in \mathcal{M}$ *we obtain,*

$$I\big(Q_R(\mathbf{x}_0)\,;\,Q_R(\mathbf{x}_1)\big) \quad \leq \quad C\,\exp\Big(-\gamma\, d(\mathbf{x}_0, \mathbf{x}_1)\Big) \tag{4.1}$$

Therefore, LLP states that if two points are far enough, then their corresponding transformation should look independent, this means, the adversary can not learn any useful statistical relationship analysing these transformed points.

The values of the parameters C and γ determine the information leakage that occurs when comparing transformed points. The parameter C is associated with the entropy of a transformed point. In effect, if $\mathbf{x}_0 = \mathbf{x}_1$, we have that if Q_R satisfies LLP, then

$$\begin{aligned} I\big(Q_R(\mathbf{x}_0)\,;\,Q_R(\mathbf{x}_1)\big) &= H\big(Q_R(\mathbf{x}_0)\big) && (4.2)\\ &\leq C && (4.3) \end{aligned}$$

Consequently, if Q_R satisfies LLP, then the parameter C must be at least an upper bound for the entropy of the transformation $H(Q_R(\mathbf{x}_0))$ at each point. Note that, if Q_R also satisfies the indistinguishability property, then the value of entropy $H(Q_R(\mathbf{x}_0))$ does not depend on $\mathbf{x}_0$.

On the other hand, the parameter γ refers to the leakage control. If γ grows, the speed with which the information leakage decays as function of the distance between points will be higher. It is important to note that in either case, LLP considers that the relationship between distance and information leakage has an exponential decay. This ensures that for sufficiently large distances, the information leakage is negligible, and therefore, for practical purposes, an adversary can not distinguish between comparing two transformed points, or two independent realizations of the same random variable.

However, the pairwise comparison between transformed points seems to not be a strong enough guarantee. In fact, limiting leakage from pairwise comparison is not enough to bound the overall information leakage. The following example illustrates this fact.

Example 4.2.1 *Consider Y_1 and Y_2 two independent random variables, both distributed as a Bernoulli(0.5), We can define the random variable Y_3 as $Y_1 \oplus Y_2$, where $\oplus$ stands for the XOR operation. Then the joint probability function is given as the following table shows,*

Y_1	Y_2	$Y_3 = Y_1 \oplus Y_2$	$\mathbb{P}(Y_1, Y_2, Y_3)$
0	0	0	0.25
0	1	1	0.25
1	0	1	0.25
1	1	0	0.25

Table 4.1: Joint probability function that illustrates that pairwise independence does not imply joint independence.

It is straightforward to check out that in this case, we get $I(Y_1\,;Y_2) = I(Y_1\,;Y_3) = I(Y_2\,;Y_3) = 0$, but $I(Y_1, Y_2\,;Y_3) = 1$. Indeed, knowing the values of Y_1 and Y_2 we can determine the value of Y_3 with no error.

In general, the pairwise mutual information just gives us a lower bound of a more generic information leakage measurement, as is shown bellow,

$$I\Big(Q(\mathbf{x}_0)\ ;\ Q(\mathbf{x}_1), Q(\mathbf{x}_2), ..., Q(\mathbf{x}_n)\Big) \quad \geq \quad \max_i I\big(Q(\mathbf{x}_0)\,; Q(\mathbf{x}_i)\big) \tag{4.4}$$

This inequality comes directly from the *Information Processing Inequality* considering the projection mapping. Hence, the information leakage from pairwise comparison is just a lower bound of the information leakage produced by more complex comparisons.

Note that this consideration is relevant in practical scenarios. Even though an adversary cannot learn relationships between a transformed point $Q(\mathbf{x}_0)$ and $Q(\mathbf{x}_i)$ because $I(Q(\mathbf{x}_0), Q(\mathbf{x}_i))$ is limited, eventually, the adversary can learn other relationships if $I\left(Q(\mathbf{x}_0)\,; Q(\mathbf{x}_1), Q(\mathbf{x}_2), ..., Q(\mathbf{x}_n)\right)$ is large.

That said, it is worth asking whether there are conditions in which the Limited Leakage Property can guarantee control of information leakage over more complex comparisons. For this, it is necessary to introduce the following definition.

Definition 4.2.3 *We say that a transformation Q_R is $\varepsilon-$**persistent** if for any point $\mathbf{x} \in \mathcal{M}$*

$$\mathbb{P}\Big(Q_R(\mathbf{x}) = Q_R(\mathbf{z}) \ , \ \forall \mathbf{z} \in B(\mathbf{x}, \varepsilon)\Big) \quad \geq \quad \frac{1}{2} \tag{4.5}$$

where $B(\mathbf{x}, \varepsilon)$ is the ball of radius ε with center $\mathbf{x}$ using the metric d. Formally,

$$B(\mathbf{x}, \varepsilon) \quad = \quad \{\mathbf{z} \in \mathcal{M} \ : \ d(\mathbf{x} \, , \, \mathbf{z}) \leq \varepsilon\} \tag{4.6}$$

This definition captures a requirement that seems reasonable for purposes of making comparisons when two points are close enough. Thus, a random transformation Q_R is ε-persistent if for any point $\mathbf{x}$ in $\mathcal{M}$, the probability that all points that are distant less than ε have the same transformation value $Q_R(\mathbf{x})$ is greater than 1/2. Since this probability, in principle, can vary depending on the value of $\mathbf{x}$, it is convenient to introduce the following quantity,

Definition 4.2.4 *Let ε be a positive number and Q_R a random transformation, we can define the **degree of ε-persistence** as,*

$$\rho(\varepsilon \, , \, Q_R) \quad = \quad \inf_{\mathbf{x} \in \mathcal{M}} \mathbb{P}\Big(Q_R(\mathbf{x}) = Q_R(\mathbf{z}) \ , \ \forall \mathbf{z} \in B(\mathbf{x}, \varepsilon)\Big) \tag{4.7}$$

From this definition, following statements are easily derived.

Proposition 4.2.1 *If R is a random variable that composes the transformation Q_R, then the following statements are true,*

1. *Q_R is ε-persistent if and only if $\rho(\varepsilon, Q_R) \geq \frac{1}{2}$*
2. *$\rho(0, Q_R) = 1$*

The notion of persistence allows us to understand the information leakage when the random transformation is applied on collections of points that are nearby. Thus, we can understand how to quantify the mutual information between the transformations of two groups whose points are in a conglomerate, as is illustrated in the figure 4.2.

The following theorem connects the Limited Leakage Property with the concept of ε-persistence, allowing to find an upper bound for the information leakage produced by complex comparisons of transformed data points.

Theorem 4.2.1 *Consider that $\{\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_n\} \subseteq B(\mathbf{x}_0, \varepsilon_1)$ and $\{\mathbf{z}_0, \mathbf{z}_1, ..., \mathbf{z}_m\} \subseteq B(\mathbf{z}_0, \varepsilon_2)$. If Q_R is a random transformation that satisfies LLP with constants C and γ, and Q_R is $\varepsilon-$persistent with $\varepsilon > \max\{\varepsilon_1, \varepsilon_2\}$,*

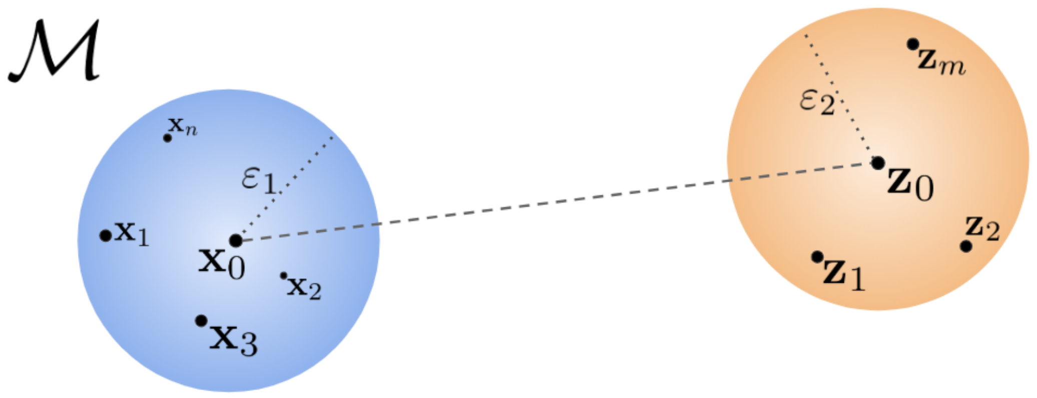


Figure 4.2: It is interesting to analyze the information leakage produced by comparing the transformed points from one group of points against the transformed points from other group, and to study the effect of the geometry in the leakage. In particular, we would like the study the effect on the leakage depending on how spread the points are.

then

$$
\begin{aligned}
I\Big(Q_R(\mathbf{x}_0)\,,\,Q_R(\mathbf{x}_1)\,,...,\,Q_R(\mathbf{x}_n)\;\;;\;\;Q_R(\mathbf{z}_0)\,,\,Q_R(\mathbf{z}_1)\,,...,\,Q_R(\mathbf{z}_m)\Big) \;\;\leq\;\; & C\,\exp\Big(-\gamma\,d(\mathbf{x}_0\,,\,\mathbf{z}_0)\Big)\\
& +H(p_1)+H(p_2)\\
& +(1-p_1)\log(|\mathcal{E}|^n-1)\\
& +(1-p_2)\log(|\mathcal{E}|^m-1) \quad (4.8)
\end{aligned}
$$

where $p_i = \rho(\varepsilon_i, Q_R)$*, and* $H(p)$ *is the entropy of a random variable distributed as a Bernoulli*(p)*, this is* $H(p) = -p\log p - (1-p)\log(1-p)$*.*

This result can be considered as a kind of generalization to LLP. Indeed, if we consider that the degrees of persistence are zero, then the upper bound of the theorem will be just the term $C\,\exp\big(-\gamma\,d(\mathbf{x}_0\,,\,\mathbf{z}_0)\big)$, and this condition is achievable if $\varepsilon_1 = \varepsilon_2 = 0$, getting the Limited Leakage Property.

A disadvantage of this bound is the fact that it depends linearly on the number of points on the ball around the points $\mathbf{x}_0$ or $\mathbf{z}_0$. Thus, given that this is only an upper bound for the information leakage, it is not clear from this result if it actually grows in the same way with the number of points observed. However, this result allows us to quantify the information leakage through a non trivial upper bound for it.

4.3 Designing a Limited Leakage Transformation

So far we have only defined what we understand by transformations that satisfy the Limited Leakage property and a theorem that analyzes the leakage of more complex comparisons under the assumption of ε -persistence. However, we have not shown any example of a transformation that achieves these conditions.

In this section we will introduce a concrete example that complies with the definitions and properties that we have established. We will consider in this case that $\mathcal{M}$ corresponds to the N-dimensional Euclidean space with the Euclidean distance $d(\mathbf{x}, \mathbf{z}) = \|\mathbf{x} - \mathbf{z}\| = \sqrt{\sum_{i=1}^{N}(x_i - z_i)^2}$. In addition, we will consider that $\mathcal{E}$ is equal to the finite set $\mathbb{Z}_k^M$, where $\mathbb{Z}_k = \{0, 1, ..., k-1\}$. The description of the random transformation is established by the following definition,

Definition 4.3.1 ***Gaussian Modular Hashing***. *Let k be an integer number greater than 1. Let A be an $M \times N$ matrix such that its components are independent and randomly generated according to a Gaussian distribution $\mathcal{N}(0, \sigma^2)$, and U be a vector with M random independent components distributed uniformly between 0 and k. We define the transformation $Q_{k,A,U} : \mathbb{R}^N \to \mathbb{Z}_k^M$ as*

$$Q_{k,A,U}(\mathbf{x}) \quad = \quad \lfloor A\mathbf{x} + U \rfloor \ (mod\ k) \tag{4.9}$$

where $(mod\ k)$ and $\lfloor \cdot \rfloor$ are component-wise, and $\lfloor \cdot \rfloor$ stands for the floor function.

Thus, in this case, the secret key R corresponds to (k, A, U). Note that k is not a random variable, but is a constant integer number. However, since k defines the distribution of U, just for completeness, we decided to include it in R.

To illustrate the effect of a Gaussian Modular Hash over the points of the Euclidean space, figure 4.3 shows the case where $k = 3$, transforming points belonging to $\mathbb{R}^2$. We can observe how this transformation is defined as a partition on the Euclidean space given by different stripes in different orientations and width produced in a random way.

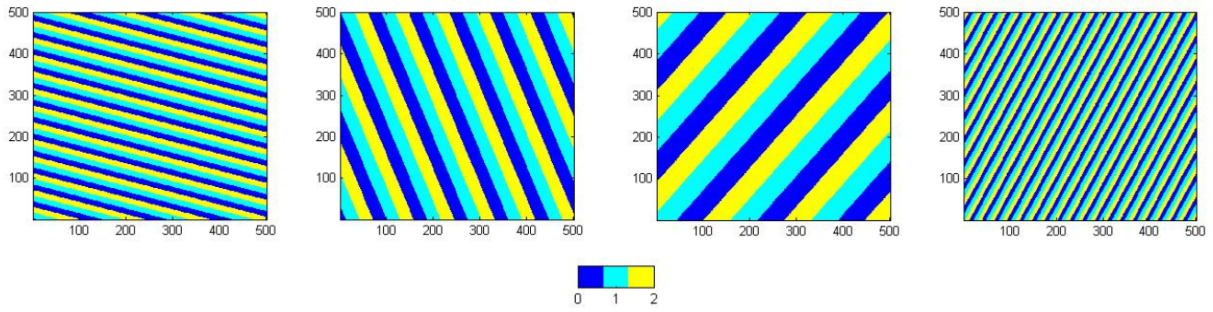


Figure 4.3: Example of realization of a Gaussian Modular Hash function from $\mathbb{R}^2$ to $\mathbb{Z}_3^4$. Each plot illustrates the value a one component of $Q_{3,A,U}(\mathbf{x})$ for different values of $\mathbf{x}$.

This construction seems to be similar to that used in the *Learning with Errors* (LWE) problem [76] [77]. Indeed, the fact that both technique use an affine linear transformation over the data to be protected. However, there are several difference that make significant distinctions on the kind of properties we can obtain. The first difference is related to the input data; in our construction we are dealing with real value vectors, but in the LWE problem, the input data is considered a vector in $\mathbb{Z}_q$, with q prime. Even though we could consider in practical scenarios a correspondence between real value vector and integer vector, for analysis of our scheme is useful to keep our assumption, especially because we are interested on study the effect on distance between points.

The second difference is in the distribution used to generate the parameters of the transformation. In both cases the operation takes the form $A\mathbf{x}+U$. As we already said, in our construction the coefficients of A are generated using a zero-mean Gaussian distribution. However, the in the LWE problem the distribution for the coefficients of the matrix A is a uniform distribution over $\mathbb{Z}_q$. Regarding the distribution of U, in LWE a custom distribution χ is used, unlike our method, that uses a uniform distribution.

Finally, other major distinction is on the kind of operations we use. In LWE both the additions and multiplication are done in modulo q. In our case, additions and multiplications are done as floating number, and the modulo appears just after converting the real value vector into an integer value vector through the floor function. In other world, we use the modulo operation primarily as a quantization function. Overall, with these difference stated, we should expect different properties.

In fact, We can show that a Gaussian Modular Hash transformation satisfies the properties we described in the previous section. First, we can analyze the distribution obtained after applying the transformation over a single point. The next result shows that indistinguishability is achievable.

Theorem 4.3.1 ***Indistinguishability***. *If (k, A, U) satisfies the conditions to compose a Gaussian Modular Hash function $Q_{k,A,U}$, then, for all $\mathbf{x} \in \mathbb{R}^N$ and for all $\mathbf{i} \in \mathbb{Z}_k^M$ we have*

$$\mathbb{P}\big(Q_{k,A,U}(\mathbf{x})\big) \quad = \quad \frac{1}{k^M} \tag{4.10}$$

Hence, the distribution of $Q_{k,A,U}(\mathbf{x})$ does not depend on the value of $\mathbf{x}$, so observing realizations from this random variable, we cannot gain any information about the original vector $\mathbf{x}$.

The second important result about Gaussian Modular Hashing is the fact that satisfies LLP using the Euclidean distance as a metric, as the following theorem states,

Theorem 4.3.2 ***Limited Leakage Property***. *If (k, A, U) satisfies the conditions to compose a Gaussian Modular Hash function $Q_{k,A,U}$, then, there exist positive constants C and γ such that for any pair of points $\mathbf{x}_1$, $\mathbf{x}_2 \in \mathbb{R}^N$ the mutual information between $Q_{k,A,U}(\mathbf{x}_1)$ and $Q_{k,A,U}(\mathbf{x}_2)$ is bounded as follows,*

$$I\big(Q_{k,A,U}(\mathbf{x}_1)\,;\,Q_{k,A,U}(\mathbf{x}_2)\big) \quad \leq \quad C\exp\left(-\gamma\sigma\|\mathbf{x}_1-\mathbf{x}_2\|\right) \tag{4.11}$$

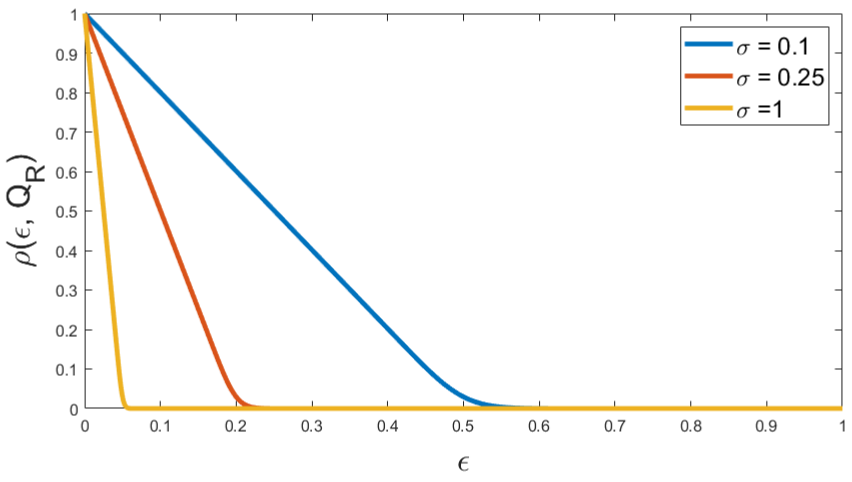


Figure 4.4: Probability that all elements in an ε-ball have the same hash value, as a function of ε for different values of σ, considering $N = 100$ and $M = 1$.

Here, we can see that the Gaussian Modular Hash satisfies LLP, but one of its main characteristics is the fact that the speed of decay depends on σ, which is the standard deviation of the Gaussian random variable used to generate A. Thus, since the parameter σ can be selected independently from the data points, this parameter helps to control the information leakage by design, that is, if we want that the mutual information between hashed data is smaller than κ when the distance between the points is larger than δ, then we can find a good value of σ to achieve this condition.

Finally, it remains to determine under what values of ε this transformation is ε-persistent. The next result clarifies this,

Theorem 4.3.3 *If (k, A, U) satisfies the conditions to compose a Gaussian Modular Hash function $Q_{k,A,U}$, where σ^2 is the variance of a_{ij}, then, the degree of persistence is given by*

$$\rho(\varepsilon, Q_{k,A,U}) \quad = \quad \left(\frac{\gamma\left(\frac{N}{2}, \frac{1}{4\varepsilon^2\sigma^2}\right)}{\Gamma\left(\frac{N}{2}\right)} - 2\sqrt{2}\sigma\varepsilon \frac{\gamma\left(\frac{N+1}{2}, \frac{1}{4\varepsilon^2\sigma^2}\right)}{\Gamma\left(\frac{N}{2}\right)} \right)^M \tag{4.12}$$

where $\gamma(s,x) = \int_0^x t^{s-1}e^{-t}dt$ is the lower incomplete gamma function, and $\Gamma(z) = \int_0^\infty x^{z-1}e^{-x}dx$ is the well known gamma function.

With this result, we can compute and plot the value of the persistence as a function of ε for different values of σ, which according to the previous theorem, controls the information leakage. Figure 4.4 shows this for $M = 1$.

We can see that as the value of σ increases, the range of ε values for which $\rho(\varepsilon, Q_R)$ is larger than $\frac{1}{2}$ decreases. We also can observe that this probability decreases almost linearly until a threshold point. With this, we can conclude that there are values of σ that allows us to reach ε-persistence, and eventually to control the information leakage under complex comparisons.

4.3.1 The case $k = 2$

In addition to the general results, we can further exploit the structure of the Gaussian Modular Hash. The following properties appear when we consider $k = 2$, that is, the result of the transformation are binary vectors.

The following result shows that information leakage can be controlled when comparing two transformed data against one.

Theorem 4.3.4 *If we consider the $Q_{2,A,U}$ is Gaussian Modular Hash, then the mutual information between $Q_{2,A,U}(\mathbf{x}_0)$ and $(Q_{2,A,U}(\mathbf{x}_1), Q_{2,A,U}(\mathbf{x}_2))$ goes exponentially fast to 0 when the minimum distance from $\mathbf{x}_0$ to $\mathbf{x}_1$ and $\mathbf{x}_2$ goes to infinity.*

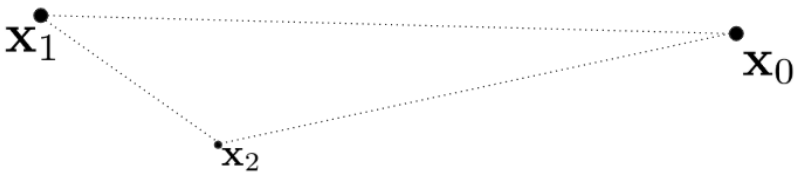


Figure 4.5: Geometric condition to control information leakage. When we compare two transformed points against a third transformed point, the information leakage is negligible if this third point is far enough.

Thus, this result shows, unlike the general result established in the previous section, that the information leakage goes to zero exponentially fast, regardless having an additional point.

The last observation is related to the generalization of this result. The following theorem states a different bound for the mutual information between one transformed point and a set of transformed points that belong to an ε ball in the original space, as figure 4.6 illustrates.

Theorem 4.3.5 *Let $\mathbf{x}_0$ be a point in $\mathbb{R}^N$. Consider the set of points $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\} \subseteq B(\mathbf{x}_1, \varepsilon)$. If we consider that Q_R is a Gaussian Modular Hash with $R = (2, A, U)$, then*

$$\begin{aligned} I\Big(Q_R(\mathbf{x}_0)\,;\,Q_R(\mathbf{x}_1), ..., Q_R(\mathbf{x}_n)\Big) \quad &\leq \quad I(Q_R(\mathbf{x}_0)\,;\,Q_R(\mathbf{x}_1)) \\ &\quad +1 - \rho(\varepsilon, Q_R) + \mathbb{P}\Big(Q_R(\mathbf{x}_1) = Q_R(\mathbf{x}^*)\Big) \log \frac{1}{\rho(\varepsilon, Q_R)} \end{aligned}$$

where $\mathbf{x}^*$ is the farthest point to $\mathbf{x}_1$ in the set $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$.

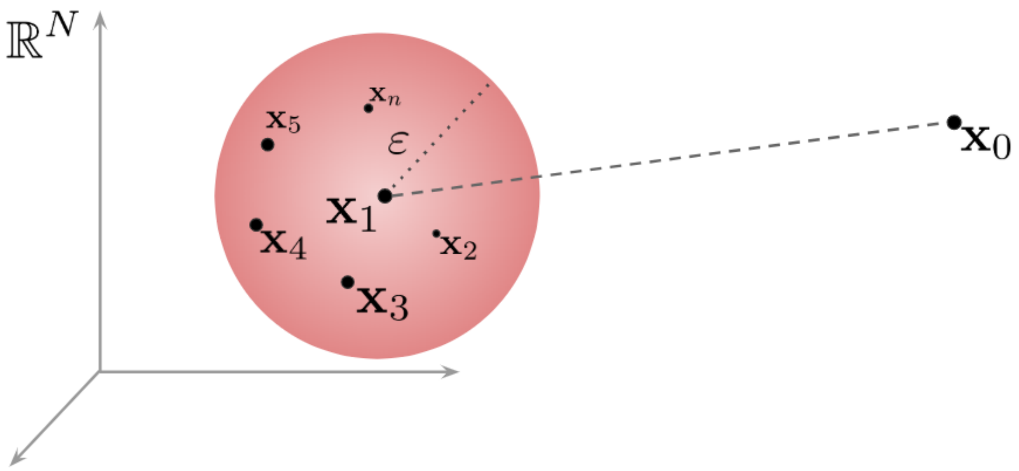


Figure 4.6: Geometric condition to control information leakage in complex comparisons.

Even though this result does not show that the information leakage goes to zero when the distance between $\mathbf{x}_0$ and $\mathbf{x}_1$ grows, it is able to give a non-trivial bound to the mutual information. The most notable consequence of this result is the dependence on epsilon, which, as seen, if it tends to zero, then similar results are obtained to satisfy LLP. Furthermore, the fact that the dimension does not depend on the number of points in the ball $B(\mathbf{x}_1, \varepsilon)$ makes the bound very useful for situations where we do not control the number of points to be transformed.

4.3.2 Leaking conditions

The results presented in the previous section give us guarantees to control the information leakage produced by the comparison between transformed version of a single point $\mathbf{x}_0$ and the transformation of the points $\mathbf{x}_1, ..., \mathbf{x}_n$. In a practical scenario, $\mathbf{x}_0$ may represent the instance a private data point while the set $\{\mathbf{x}_1, ..., \mathbf{x}_n\}$ represent points generated by an adversary to gain information about $\mathbf{x}_0$. We already know that if the set is far enough from the point $\mathbf{x}_0$ and its diameter is also small, then the information leakage is small.

A good question is to know what happens with the leakage information in case these conditions are not met. To address this point, we can think on the persistence property. In fact, by construction, our transformation enables to identify when two points are close just by analyzing a collision, thus, if the distance between two points is small, then the probability of obtaining the same value after applying our transformation is high (larger that 0.5). Hence, if at least one point of the set $\{\mathbf{x}_1, ..., \mathbf{x}_n\}$ is close to $\mathbf{x}_0$, then you could identify that just for simple collision analysis. In information theoretical terms, we can simply use the inequality shown in 4.4 and state that information leakage is not negligible. From an adversarial point of view, this is relevant because the comparison allows to estimate information about the private data $\mathbf{x}_0$.

The key issue is the possibility to infer the actual distance between data points just by comparing the transforming points. In fact, in [105] a triangulation attack is discussed to evaluate the security of hashing methods to protect data. Here it is shown that if an adversary obtain the distance between the private point $\mathbf{x}_0$ and other three random points, it is possible to obtain a good estimation of the private data. In this case, unlike our method, it is possible to estimate any distance value through hashes; in our case that property is limited by the value of the distances, thus, if the distance between the points is large, no useful comparison can be made, because the comparison should be similar to compare two indistinguishable independent random variables. Overall, under the context of limited leakage transformations, an adversary can gain information about data points as long as the adversarial points are close enough to the privata data point.

4.4 Other Metrics

We have analyzed in detail a concrete example of a random transformation that satisfies our privacy requirements. It remains to be asked if it is possible to extend these concepts to other metrics. Below are some guidelines on how to build limited leakage transformations for different metrics.

4.4.1 Mahalanobis Distance

Given a positive definitive matrix S, its corresponding Mahalanobis distance is defined as a metric in $\mathbb{R}^N$, as follows

$$d_S(\mathbf{x}_1\,,\,\mathbf{x}_2) \quad = \quad \sqrt{(\mathbf{x}_1 - \mathbf{x}_2)^\top S^{-1}(\mathbf{x}_1 - \mathbf{x}_2)} \tag{4.13}$$

This distance is extensively used in Statistics and Machine Learning, since it is related to the likelihood obtained under a multivariate Gaussian model. It turns out that it is possible to adapt the Gaussian Modular Hash function to transfer the properties obtained for the Euclidean distance to the Mahalanobis distance.

Definition 4.4.1 **Γ-*Gaussian Modular Hash*.** *Let k be an integer number greater than 1. Let A be a $M \times N$ matrix such that each of its rows are independent and randomly generated according to a multivariate Gaussian distribution $\mathcal{N}(\mathbf{0}, \sigma^2\Gamma \cdot \Gamma^t)$, and U be a vector with M random independent components distributed uniformly between 0 and k. We define the transformation $Q_{k,A,U} : \mathbb{R}^N \to \mathbb{Z}_k^M$ as*

$$Q_{k,A,U}(\mathbf{x}) \quad = \quad \lfloor A\mathbf{x} + U \rfloor \; (mod\ k) \tag{4.14}$$

where $(mod\ k)$ and $\lfloor \cdot \rfloor$ are component-wise, and $\lfloor \cdot \rfloor$ stands for the floor function.

With this hash function, we can obtain the indistinguishability result along with the following,

Proposition 4.4.1 *If $Q_{k,A,U}$ is a $\Gamma-$Gaussian Modular Hash, then then, there exist positive constants C and γ such that for any pair of points $\mathbf{x}_1$, $\mathbf{x}_2 \in \mathbb{R}^N$ the mutual information between $Q_{k,A,U}(\mathbf{x}_1)$ and $Q_{k,A,U}(\mathbf{x}_2)$ is bounded as follows,*

$$I\big(Q_{k,A,U}(\mathbf{x}_1)\,;\,Q_{k,A,U}(\mathbf{x}_2)\big) \quad \leq \quad C\exp\left(-\gamma\sigma\, d_S(\mathbf{x}_1,\mathbf{x}_2)\right) \tag{4.15}$$

where $S^{-1} = \Gamma\Gamma^t$

4.4.2 Manhattan Distance

A more interesting case is the one corresponding to the distance ℓ_1, or also known as Manhattan Distance, where the distance between two vectors $\mathbf{x}$ and $\mathbf{z}$ is given by $\|\mathbf{x}-\mathbf{z}\|_1 = \sum_{i=1}^N |x_i - z_i|$. In order to build a transformation that satisfies LLP with respect the Manhattan distance, it is necessary to use a specific distribution for the key generation.

Definition 4.4.2 *Let X be a random variable and $\sigma > 0$. X is distributed as Cauchy(σ) if its density function is given by,*

$$f_X(x\,;\,\sigma) \quad = \quad \frac{1}{\pi}\cdot\frac{\sigma}{x^2+\sigma^2} \tag{4.16}$$

Simulating this type of distribution is not difficult. For example, we can use the following well known property to obtain samples from a Cauchy distribution.

Proposition 4.4.2 *If $X \sim$ Unif$(0,1)$ and any $\sigma > 0$, then we obtain $\sigma \cdot \tan\left(\pi(X-\frac{1}{2})\right) \sim$ Cauchy(σ).*

The Cauchy distribution allows to define the random transformation described below,

Definition 4.4.3 ***σ-Cauchy Modular Hashing**. Let k be an integer number greater than 1. Let A be an $M \times N$ matrix such that its components are independent and randomly generated according to a Cauchy distribution Cauchy(σ), and U be a vector with M random independent components distributed uniformly between 0 and k. We define the transformation $Q_{k,A,U} : \mathbb{R}^N \to \mathbb{Z}_k^M$ as*

$$Q_{k,A,U}(\mathbf{x}) \quad = \quad \lfloor A\mathbf{x} + U \rfloor \ (mod\ k) \tag{4.17}$$

where $(mod\ k)$ and $\lfloor\cdot\rfloor$ are component-wise, and $\lfloor\cdot\rfloor$ stands for the floor function.

With this change on the distribution in the key generation, we can switch the metric by the Manhattan distance in the formulation of the limited leakage property, as the following proposition states,

Proposition 4.4.3 *If $Q_{k,A,U}$ is a σ-Cauchy Modular Hash, then then, there exist positive constants C and γ such that for any pair of points $\mathbf{x}_1$, $\mathbf{x}_2 \in \mathbb{R}^N$ the mutual information between $Q_{k,A,U}(\mathbf{x}_1)$ and $Q_{k,A,U}(\mathbf{x}_2)$ is bounded as follows,*

$$I\big(Q_{k,A,U}(\mathbf{x}_1)\,;\,Q_{k,A,U}(\mathbf{x}_2)\big) \quad \leq \quad C\exp\left(-\gamma\sigma \left\|\mathbf{x}_1 - \mathbf{x}_2\right\|_1\right) \tag{4.18}$$

4.4.3 Cosine Distance

The *cosine similarity* is a measure extensively used in Machine Learning. This is a similarity measure between two vectors (non-null) given by $\frac{\langle \mathbf{x},\mathbf{z}\rangle}{\|\mathbf{x}\|\cdot\|\mathbf{z}\|}$. Thus, two vectors are considered similar if this number is close to 1. In order to get a kind of metric that captures in a similar way this notion of closeness, *cosine distance* is defined as

$$d_{cos}(\mathbf{x},\mathbf{z}) \quad = \quad 2 - 2\frac{\langle \mathbf{x},\mathbf{z}\rangle}{\|\mathbf{x}\|\cdot\|\mathbf{z}\|} \tag{4.19}$$

Note that this is not a metric strictly speaking. In fact, it is easy to find different values of $\mathbf{x}$ and $\mathbf{z}$ such that $d_{cos}(\mathbf{x},\mathbf{z}) = 0$. However, its use is very common in different tasks.

In order to adapt the proposed hash to the cosine distance, it is enough to notice that this distance corresponds to the squared Euclidean distance with normalized vectors. Therefore, using a Gaussian Modular Hash it is possible to prove the following,

Proposition 4.4.4 *If $Q_{k,A,U}$ is a Gaussian Modular Hash, then then, there exist positive constants C and γ such that for any pair of points $\mathbf{x}_1$, $\mathbf{x}_2 \in \mathbb{R}^N$ the mutual information between $Q_{k,A,U}(\mathbf{x}_1)$ and $Q_{k,A,U}(\mathbf{x}_2)$ is bounded as follows,*

$$I\left(Q_{k,A,U}\left(\frac{\mathbf{x}_1}{\|\mathbf{x}_1\|}\right)\,;\,Q_{k,A,U}\left(\frac{\mathbf{x}_2}{\|\mathbf{x}_2\|}\right)\right) \quad \leq \quad C\exp\left(-\gamma\sigma\, d_{cos}(\mathbf{x}_1,\mathbf{x}_2)\right) \tag{4.20}$$

Note, however, that for this case, the actual distance between the original vectors is bounded between 0 and 2, so the analysis of the information leakage is restricted to this range.

4.5 Comparison with other methods

From a practical perspective, we can compare the proposed approach with different privacy mechanisms. First, we can make a comparison between Limited Leakage Transformations and well known cryptographic tools. In particular we can compare our method with Standard Symmetric Encryption and Cryptographic Hashing. There are three aspects we are interested to analyze: the need of secret keys, the capacity to recover the original message, and the feasibility of performing computation on transformed data. For these three

	Standard Encryption	Cryptographic Hash	LLP Transformation
Keys	Secret	Public	Secret
Recovering Plaintext	No error	Infeasible	Depends on the Structure of Plaintext in case of knowing the key
Computation	Infeasible	Through Collision	*Next Part*

Table 4.2: Comparison between standard cryptographic methods

methods, table 4.2 summarizes our observations. First of all, we can say that both in LLP transformations and in Encryption we need to keep our keys secret, otherwise privacy can be compromised. On the other hand, we know that using the secret key, the plaintext can be recovered with no error using the secret key on the ciphertext obtained from Encryption. In the case of Cryptographic hash, we know that recovering the original message from its hashed value is an unfeasible task. For the case of the transformations we have discussed, we do not have a strong statement; indeed, it is a research topic.

With respect to the kind of computations we can perform after transforming data points, it is well known that, using standard Encryption this task in unfeasible; no operation can be performed on encrypted data using standard probabilistic schemes. In the case of a Cryptographic Hash, the only operation we can do is identifying collisions, that is, we just can identify if two data points are the same or not (with high probability). Finally, we can consider that thetransformation with ε-persistence and LLP are a relaxation of the Cryptographic Hash, in terms of whether we can identify if two points are close enough or not. In the next part of this work we will continue analyzing the effects of comparing transformed points that are close.

On the other hand, the idea of leaking relevant information to improve efficiency in a particular task is not new. For example, in *Order-preserving encryption* (OPE) [106] [107] the objective is to encrypt data such that the order between plaintexts is preserved. Thus, it is possible that a remote untrusted server can index private encrypted data that it receives, using a data structure that permits efficiently answering queries based on ranges; returning ciphertexts whose corresponding plaintexts belong to some range $[a, b]$. Despite the idea of leaking information is the same, the analysis of the privacy requirements is different from our approach. The security of most of the OPE schemes are based on random injections on large sets, being vulnerable to multiple attacks [108]. The main difference with our approach is the fact that we are interested on just leaking *local* information (just when points are close enough), while in OPE any arbitrary order relation can be infer by comparing ciphertexts.

Finally, we can compare Differential Privacy with our method. In fact, even though both approaches take a probabilistic formulation, they have significant differences. The main distinction is the fact that differential privacy mechanisms are algorithm dependent, this means that the method can change if the objective of the algorithm changes. Moreover, most of the differential privacy approaches consider that the data is processed in a trusted infrastructure, where the randomness is added. In contrast, our method just makes assumptions about the geometric relationship between data points, being agnostic about a potential algorithm to be used. In addition, if the transformation satisfies the indistinguishability property, we know that, observing one single transformed point, an adversary cannot infer any relevant information about the original point.

The method presented can be comparable to the work shown in [15], where a binary quantization function is proposed to preserve privacy. Nevertheless, the proposed transformation does not satisfy the indistinguishability property, and the analysis of the decay of the mutual information is done in a particular case. In [105] a privacy mechanism over a type of hashes is proposed. However, to analyze the mutual information between data points distributional assumptions must be made.

4.6 Contributions

We have proposed the concept of Limited Leakage Transformation as a privacy mechanism that allows to transform data points such that it removes relationships between vectors that are far enough, but maintains persistence if they are close.

We have shown a generic theoretical result about how the information leakage can be controlled under determined geometric conditions on the data points. We also have presented particular examples of this kind of transformation, providing a rigorous analysis about the information leakage at different levels of comparisons.

In the next part of this work, we discuss how to take advantage on the information leakage obtained when two points are close in order to perform some computation on transformed points, and analyze its implication in machine learning systems.

Part III

Limited Computation. About the utility of the information leakage.

Chapter 5

Limited Distance Estimation

In the previous chapter we showed a family of transformations that satisfy a privacy guarantee that relates the distance between points to the relation we can learn comparing their corresponding transformed values. We demonstrated that if the points are far enough, the mutual information between transformed points is exponentially close to zero, making the comparison between them meaningless, because comparing transformed points will be indistinguishable from comparing independent random variables.

Hence, the natural question is what should happen when the points are close. In fact, part of the willingness to sacrifice privacy to enhance computation is to understanding properly the kind of operation we can perform comparing transformed values and identify the information we are leaking.

In this chapter, we will focus on the Modular Hash function, as we proposed previously, and we will analyse how to estimate the actual distance between data points using hashes. From the properties we have from the previous chapter, we know that this estimation can be done only if the points are close enough. We will call this kind of estimation *Limited Distance Estimation.* To do so, we analyse two metrics in the hash domain $\mathbb{Z}_k^M$: the Hamming distance and the Modular distance. We establish an analytical relationship between these metrics with the corresponding metric in $\mathbb{R}^N$.

Later, in the following chapter, we will make the connection with some Machine Learning tasks, and in particular how to run Machine Learning Algorithms over the hashed data.

5.1 General Observation

We want to estimate the distance between $\mathbf{x}$ and $\mathbf{z}$ using the values of $Q_R(\mathbf{x})$ and $Q_R(\mathbf{z})$. In particular, we will study the expected value of $f(Q_R(\mathbf{x}), Q_R(\mathbf{z}))$, for some function f. If Q_R satisfies the indistinguishability property along with LLP, and if the distance between $\mathbf{x}$ and $\mathbf{z}$ is large, then $\mathbb{E}\Big[f(Q_R(\mathbf{x}), Q_R(\mathbf{z}))\Big] \approx K$, where K is a constant value which does not depend on either of $\mathbf{x}$ or $\mathbf{z}$. Indeed, under these conditions, we can not

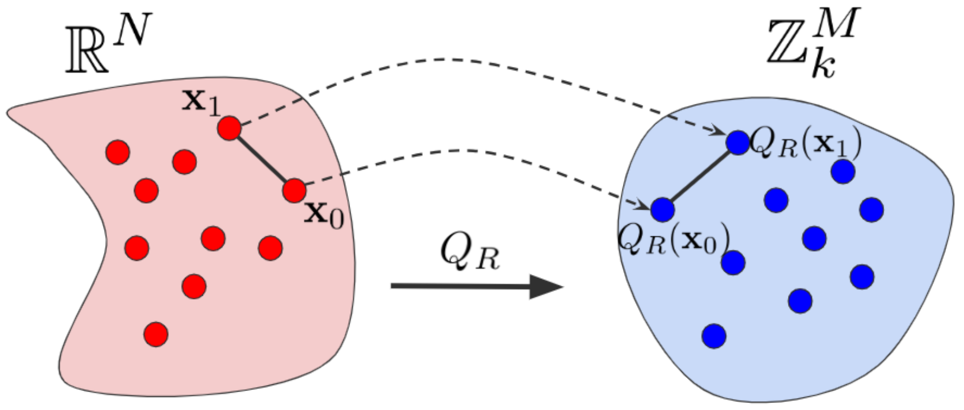


Figure 5.1: We want to study the relationship between transformed points when the original vectors are close enough.

distinguish whether we are comparing $Q_R(\mathbf{x})$ with $Q_R(\mathbf{z})$ or if they are two independent random variables W_1 and W_2 with the same distribution of $Q_R(\mathbf{x})$.

5.2 Hamming Distance between Hashes

The first approach for trying to recover the actual distance between points through the comparison of hashes is analyzing the probability of getting a collision between them. It turns out that we can recover the distance between the original points through hash collision as long as the distance between data points is small enough. Since we are comparing hashes, which are elements in $\mathbb{Z}_k^M$, we need to define a way to take advantage of the collision at component-wise level.

Definition 5.2.1 *Let* $\mathbf{a}$ *and* $\mathbf{b} \in \mathbb{Z}_k^M$*, we can define the* ***Normalized Hamming Distance*** *between these two points as,*

$$d_H(\mathbf{a}\,,\,\mathbf{b}) \quad = \quad \frac{1}{M}\sum_{i=1}^{M} 1[a_i \neq b_i] \tag{5.1}$$

where

$$1[a \neq b] \quad = \quad \begin{cases} 1 & \textit{if } a \neq b \\ 0 & \textit{otherwise} \end{cases} \tag{5.2}$$

Hence, the Normalized Hamming Distance between two elements in $\mathbb{Z}_k^M$ is a number between 0 and 1, and its formulation does not depend directly on k.

One important observation about the Gaussian Modular Hash function is the fact that its components are statistically independent with the same distribution, therefore, it is enough to analyze the probability of collision at component level. Moreover, under this observation it is easy to see that we can consider the Normalized Hamming Distance simply as an average of independent and identically distributed random variables.

The following theorem gives us a precise formulation for the probability of getting a collision of in any component between hashes,

Theorem 5.2.1 *If $\mathbf{x}_1$ and $\mathbf{x}_2 \in \mathbb{R}^N$ and $R = (k, A, U)$ is a random key to compose a σ-Gaussian Modular Hash, then the probability of getting a collision in the j component is given by the following expression,*

$$\mathbb{P}\Big(Q_R(\mathbf{x}_1)_j = Q_R(\mathbf{x}_2)_j\Big) \quad = \quad \frac{1}{k} + \frac{2}{k}\sum_{i=1}^{\infty} sinc^2\left(\frac{i}{k}\right)\exp\left(-2\left(\frac{\pi\,\sigma\,i\,\|\mathbf{x}_1 - \mathbf{x}_2\|}{k}\right)^2\right) \tag{5.3}$$

where $sinc(x) = \frac{\sin(\pi x)}{\pi x}$.

Despite the complexity of the expression obtained in the previous theorem, from this result, we can see that the probability of getting a collision depends just on the Euclidean distance between points $\mathbf{x}_1$ and $\mathbf{x}_2$. To have a better understanding of this expression, we can bound it and study its asymptotic behavior,

Theorem 5.2.2 *If $\mathbf{x}_1$ and $\mathbf{x}_2 \in \mathbb{R}^N$ and $R = (k, A, U)$ is a random key to compose a σ-Gaussian Modular Hash, then*

$$\frac{1}{k} \quad \leq \quad \mathbb{P}\Big(Q_R(\mathbf{x}_1)_j = Q_R(\mathbf{x}_2)_j\Big) \quad \leq \quad \frac{1}{k}\left(1 + \frac{k^2}{3}\exp\left(-2\left(\frac{\pi\sigma\|\mathbf{x}_1 - \mathbf{x}_2\|}{k}\right)^2\right)\right) \tag{5.4}$$

Therefore, if $\|\mathbf{x}_1 - \mathbf{x}_2\| \to \infty$, then

$$\mathbb{P}\Big(Q_R(\mathbf{x}_1)_j = Q_R(\mathbf{x}_2)_j\Big) \quad \to \quad \frac{1}{k} \tag{5.5}$$

Thus, the probability of collision converges exponentially fast to the constant value $\frac{1}{k}$, which corresponds to the probability of getting a collision after drawing two independent and uniform distributed random variables over $\mathbb{Z}_k$.

From theorem 5.2.1, and thanks to the linearity of the expectation, it is very straightforward to obtain the following corollary,

Corollary 5.2.2.1 *If $R = (k, A, U)$ is a random key to compose a σ-Gaussian Modular Hash, then, for any pair of points $\mathbf{x}_1$ and $\mathbf{x}_2 \in \mathbb{R}^N$ the expected value of the Normalized Hamming Distance is given by*

$$\mathbb{E}\Big[d_H\big(Q_R(\mathbf{x}_1)\,,\,Q_R(\mathbf{x}_2)\big)\Big] \quad = \quad 1 - \frac{1}{k}\left(1 + 2\sum_{i=1}^{\infty} sinc^2\left(\frac{i}{k}\right)\exp\left(-2\left(\frac{\pi\,\sigma\,i\,\|\mathbf{x}_1 - \mathbf{x}_2\|}{k}\right)^2\right)\right) \tag{5.6}$$

Figure 5.2 shows the shape of obtained from this expression for different values of k as function of the Euclidean distance.

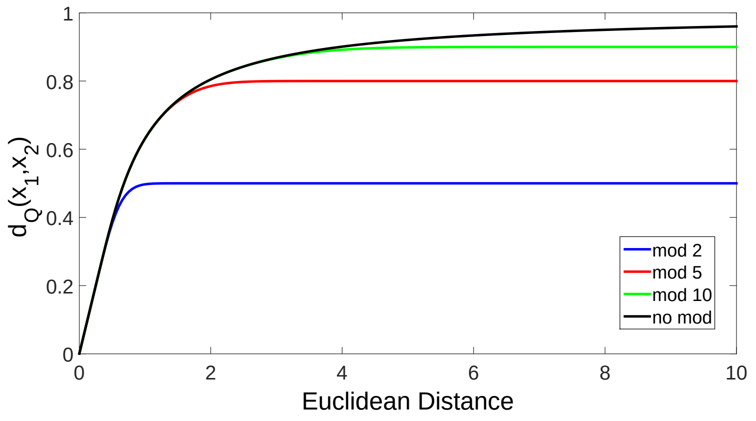


Figure 5.2: Expected value of Normalized Hamming Distance between Gaussian Modular Hashes as function of the Euclidean distance using different values of k.

From here, we can see that when the Euclidean distance is small enough, there is one-to-one relationship between the Euclidean distance and the expected value of the Hamming distance between hashes.

All these results are in expectation and eventually the actual value of the Hamming distance between hashes may differ. However, thanks to the Law of Large Numbers, the Normalized Hamming Distance between $Q_R(\mathbf{x_1})$ and $Q_R(\mathbf{x}_1)$ converges to the expected value for large values of M, since this distance is an average of independent and identically distributed random variables. We see in figure 5.3 that the empirical estimate closely follows the plot for the theoretical expression.

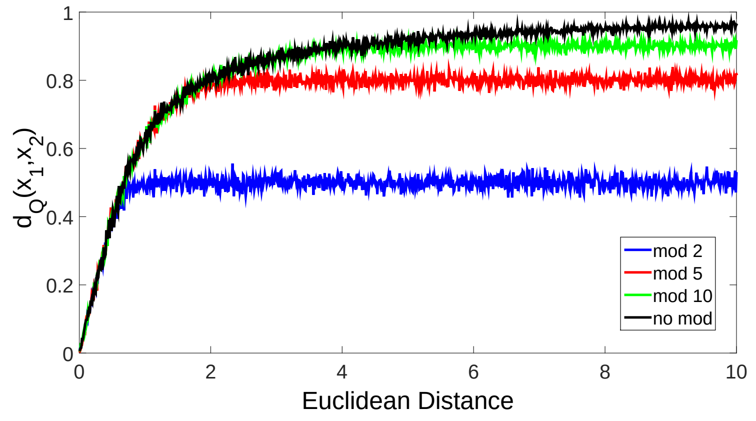


Figure 5.3: Simulations of Normalized Hamming distance between Gaussian Modular Hashes as function of the Euclidean distance between data points

A more formal guarantee is given by following result, where the probability of having a small error in estimation depends on the value of M and the number of points to be transformed,

Proposition 5.2.1 *If we have n points $\mathbf{x}_1, ..., \mathbf{x}_n \in \mathbb{R}^N$, and $M \geq \frac{1}{\varepsilon^2} \log\left(\frac{n}{2\eta}\right)$, then*

$$\mathbb{P}\left(\forall \mathbf{x}_i, \mathbf{x}_j, \left| d_H(Q_R(\mathbf{x}_i), Q_R(\mathbf{x}_j)) - \mathbb{E}\left[d_H(Q_R(\mathbf{x}_i), Q_R(\mathbf{x}_j))\right] \right| < \varepsilon \right) \quad \geq \quad 1 - \eta \tag{5.7}$$

This result allows to control the error in the estimation of the Euclidean distance based on this one-to-one relationship depending on the number of components in the hash function. A larger M allows a better estimation, but also a larger information leakage. Therefore, here we have a clear trade-off between privacy and utility.

5.3 Modular Distance between Hashes

We also can use the topological structure in $\mathbb{Z}_k$, and work with the modular distance. The *modular distance* d_{mod} is a natural metric in $\mathbb{Z}_k$, and it is defined for any pair $(a, b) \in \mathbb{Z}_k \times \mathbb{Z}_k$ as

$$d_{mod}(a, b) \quad = \quad \min\left(|a - b|\,,\, k - |a - b|\right) \tag{5.8}$$

This is equivalent to computing the shortest path between numbers assuming they are arranged as a ring, as figure 5.4 shows.

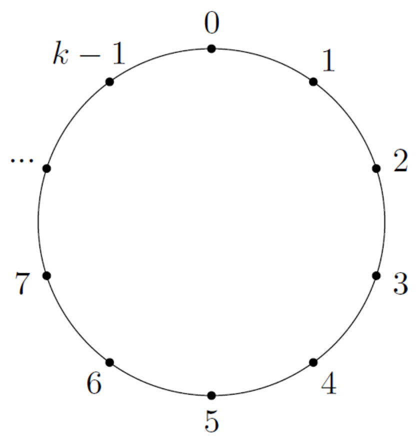


Figure 5.4: Topology of Z_k to define the Modular Distance between its elements.

Then, similarly to the Hamming distance case, we can extend this metric to $\mathbb{Z}_k^M$ using a normalized version of it. The next definition clarifies this point,

Definition 5.3.1 *Let* $\mathbf{a}$ *and* $\mathbf{b} \in \mathbb{Z}_k^M$*, we can define the* ***Normalized Modular Distance*** *between these two points as,*

$$d_{mod}\big(\mathbf{a}\,,\,\mathbf{b}\big) \quad = \quad \frac{1}{M}\sum_{i=1}^{M} d_{mod}(a_i, b_i) \tag{5.9}$$

Note that for the case $k = 2$ we obtain $d_H = d_{mod}$. Using this metric, we can find a similar relationship to the previous case, stating a relation between the modular distance between hashes to the Euclidean distances between the original data points, as the following theorem shows,

Theorem 5.3.1 *If* $R = (k, A, U)$ *is a random key to compose a* σ*-Gaussian Modular Hash, then, for any pair of points* $\mathbf{x}_1\,,\mathbf{x}_2 \in \mathbb{R}^N$*,* $\forall k$ *even, we have:*

$$\mathbb{E}\left[d_{mod}\left(Q_R(\mathbf{x}_1)\ ,\ Q_R(\mathbf{x}_2)\right)\right] \quad = \quad \frac{k}{4} - \frac{2k}{\pi^2}\sum_{j=1}^{\infty}\frac{1}{(2j-1)^2}\exp\left(-2\left(\frac{\pi\,\sigma\,(2j-1)\,\|\mathbf{x}_1 - \mathbf{x}_2\|}{k}\right)^2\right) \tag{5.10}$$

This expression seems to be complex and, even though there is a functional relation between $\|\mathbf{x}_1 - \mathbf{x}_2\|$ and the expected value of the modular distance between the corresponding hashes, it is not clear how to estimate the actual euclidean distance after applying this kind of technique.

The following result gives us an estimation of the error when we want to use directly the modular distance as an estimation of the Euclidean distance:

Theorem 5.3.2 *Setting* $\sigma = \sqrt{\frac{\pi}{2}}$*, and given* $R = (k, A, U)$ *the random key that composes a* $\sigma-$*Gaussian Modular Hash, and defining the error*

$$\epsilon(\|\mathbf{x}_1 - \mathbf{x}_2\|\,,\,k) \quad := \quad \left|\mathbb{E}\Big[d_{mod}\left(Q_R(\mathbf{x}_1)\,,\,Q_R(\mathbf{x}_2)\right)\Big] - \|\mathbf{x}_1 - \mathbf{x}_2\|\right| \tag{5.11}$$

we have that

$$\epsilon(\|\mathbf{x}_1 - \mathbf{x}_2\|\,,\,k) \quad \leq \quad F(\|\mathbf{x}_1 - \mathbf{x}_2\|, k) \tag{5.12}$$

where

$$F(t, k) \quad = \quad t \cdot \exp\left(-\frac{k^2}{4\pi t^2}\right) \tag{5.13}$$

It is easy to see that F is increasing in t and decreasing in k. Moreover, for a fixed t, when k tends to infinity $F(t, k)$ tends to 0. Therefore, we can prove the following proposition

Proposition 5.3.1 *$\forall \varepsilon > 0$, $\forall T > 0$, $\exists k$ even, $\forall \|\mathbf{x}_1 - \mathbf{x}_2\| < T$*

$$\epsilon(\|\mathbf{x}_1 - \mathbf{x}_2\|, k) \quad < \quad \varepsilon \tag{5.14}$$

This proposition states that, given a threshold T, we can find a value of k large enough, such that the difference between $\|\mathbf{x}_1 - \mathbf{x}_2\|$ and the expected value of the modular distance between the corresponding hashes is as small as we want for all the distances smaller than T. Figure 5.5 (left) shows an alternative way of seeing this, which plots the expression given by Theorem 5.3.1, showing an almost linear behavior for distances smaller than some threshold, then converging to $\frac{k}{4}$. Therefore, we can compute an accurate estimate of the Euclidean distance between $\mathbf{x}_1$ and $\mathbf{x}_2$ using their hashes directly.

For practical purposes, we need to use the Normalized Modular distance as the estimate of the Euclidean distance, and similarly to the Hamming distance case, the error depends on M, the number of components that the hash has. The following proposition helps us to control this error in the estimation,

Proposition 5.3.2 *Taking $\sigma = \sqrt{\frac{\pi}{2}}$ and $R = (k, A, U)$ the random key that composes a $\sigma-$Gaussian Modular Hash, if $M \geq \frac{\log(2)\cdot(\beta+1)\cdot k^2}{8\varepsilon^2}$, then*

$$\mathbb{P}\Big(\Big|d_{mod}\left(Q_R(\mathbf{x}_1), Q_R(\mathbf{x}_2)\right) \; - \; \mathbb{E}\left[d_{mod}\left(Q_R(\mathbf{x}_1), Q_{k,A,U}(\mathbf{x}_2)\right)\right]\Big| < \varepsilon\Big) \quad \geq \quad 1 - \frac{1}{2^\beta} \tag{5.15}$$

Thus, the Normalized Modular Distance is similar to its expectation value with high probability if M satisfies the described condition. For example, if $\beta = 10$, $\varepsilon = 0.5$ and $k = 8$, then with $M \geq 244$ we can obtain an estimate of the expected Modular distance, and as we showed, a very close estimate of the actual Euclidean distance with precision 0.5 with probability more than 0.999.

A very relevant consequence of this results is the non-dependency of N, the dimensionality of the data points, on M, the required number of components in the hash. Hence, we may consider this random transformation as a dimensionality reduction technique. Figure 5.5 (right) shows how the normalized modular distance approximates the Euclidean distance when this is smaller than some threshold, using different values of k, being totally consistent with the previous theoretical results.

Finally, it should be mentioned that the results presented here are valid for the Euclidean distance. Using the previously described constructions, such as Γ-Gaussian Modular Hash, or Cauchy Modular Hash, we can obtain analogous results for the Mahalanobis distance and Manhattan distance respectively.

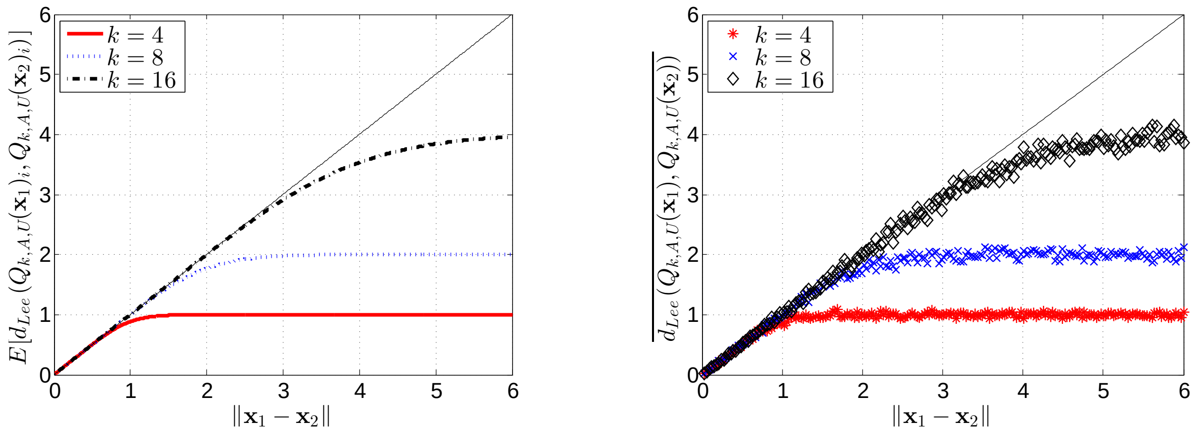


Figure 5.5: **Left:** Expected value of the Modular Distance between $Q_R(\mathbf{x}_1)$ and $Q_R(\mathbf{x}_2)$ as a function of $\|\mathbf{x}_1 - \mathbf{x}_2\|$ using expression given by Theorem 5.3.1 and considering $\sigma = \sqrt{\frac{\pi}{2}}$. **Right:** Simulated data. Normalized modular distance between hashes as a function of $\|\mathbf{x}_1 - \mathbf{x}_2\|$ for $\sigma = \sqrt{\frac{\pi}{2}}$, $N = 5000$ and 500 samples.

5.4 Related Work and Contributions

The idea of finding a transformation that preserves the distance between data instances is not new. The most well known example of this is the transformation provided by the Johnson-Lindenstrauss Lemma, which preserves the Euclidean distance under a multiplicative error, this is

$$(1-\epsilon)\|\mathbf{x}-\mathbf{z}\|^2 \quad \leq \quad \|f(\mathbf{x})-f(\mathbf{z})\|^2 \quad \leq \quad (1+\epsilon)\|\mathbf{x}-\mathbf{z}\|^2 \tag{5.16}$$

with a small value of ϵ.

One way to construct this kind of transformation is using random projections. It is possible to prove that if A is a matrix with components drawn randomly from a particular distribution, the conditions in 5.16 are achieved by the transformation $f(\mathbf{x}) = A\mathbf{x}$ with high probability.

Other examples are Locality Sensitive Hashing (LSH) [10] schemes, that map the input data space into a set of "buckets" using a probabilistic map such that vectors that are close to each other in the input space map into the same bucket with very high probability, whereas those that are distant from one another have a high probability of being mapped into different buckets.

Formally, a locality sensitive hashing function is a map h from a metric space $\mathcal{M}$ to a set B such that, given any two elements $x_1, x_2 \in \mathcal{M}$,

$$\begin{aligned} d(x_1,x_2) < r \quad &\Rightarrow \quad h(x_1) = h(x_2) \text{ with probability } \geq P_1 \\ d(x_1,x_2) > cr \quad &\Rightarrow \quad h(x_1) \neq h(x_2) \text{ with probability } \geq P_2 \end{aligned} \tag{5.17}$$

for some radius of interest r and some constant c. Ideally both P_1 and P_2 are high. Thus, vectors that are less than a distance r apart have a high probability of being hashed to the same value, while that is highly improbable if they are more than cr apart.

Although LSH schemes have been primarily used as a mechanism for fast nearest-neighbor matching [10, 11] they have also been used to both hide the data, and compute distances [12, 13]. However, they do not satisfy the privacy requirements stated in the previous part of this work. Actually, in general it is not difficult to infer information about the hidden element just observing the transformed point.

In order to guarantee privacy they must be combined with homomorphic encryption schemes [14]. In [15] Boufounos *et al.* propose a band-quantization scheme that is shown to be information-theoretically secure, and has been successfully applied to nearest-neighbor problems [15]; however it reveals the length of the input vector.

Therefore, the main contribution of this part is to show that the computational approach presented in this chapter is not inconsistent with our privacy requirements, and in fact, we can use the transformed data to perform a limited distance computation.

Chapter 6

Private Distance-based Machine Learning

Until now we have analyzed how to take advantage of the hash comparisons when the original points are close enough. In the previous chapter we showed two approaches to infer the actual distance between points in the case of Modular Hashes. This can be very useful in cases where you have to do distance calculations where the value of the distance is not entirely sensitive but the data on which it is calculated is.

In this chapter we will study how to use distance-based machine learning models on data to which Modular Hash has been applied in order to preserve privacy in the sense described in the previous chapters. We will show how the algorithms can be adapted as well as conditions on the hashes in order to make the least possible modification to the machine learning methods.

6.1 Use by Substitution

As we mentioned earlier, our focus in this work is distance-based machine learning methods, so the simplest way to use the hashing schemes presented is simply to replace the distance between points by the corresponding metric between hashes. For example, in case the algorithm depends on the Euclidean distance, we can consider the following approximation

$$d_{mod}\big(Q_R(\mathbf{x}_1)\,,\,Q_R(\mathbf{x}_2)\big) \quad \approx \quad \|\mathbf{x}_1 - \mathbf{x}_2\| \tag{6.1}$$

So, instead of uploading the data without protection to the cloud to train a distance-based algorithm, we can apply some of the hash functions discussed before uploading the data to the cloud, so that the estimate of the distances over the data has been limited by limiting the information leakage as we mentioned before.

This applies very directly to several Machine Learning algorithms. For example, in *Clustering*, where the objective is to identify groups in data. The same approach can be used for *Nearest Neighbor Classification*, where it is assumed that we have a data set where each data instance has a label associated with it, which

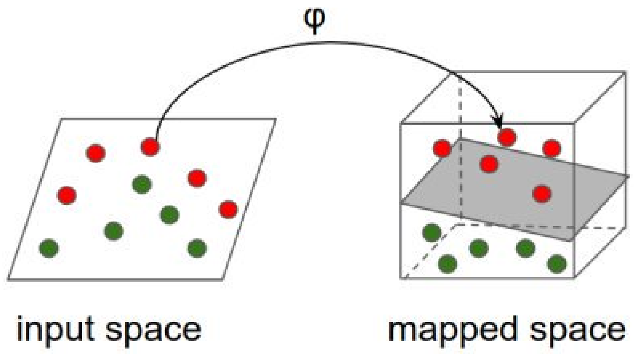


Figure 6.1: The Kernel trick is based on the idea of mapping data points into a higher dimensional space in order to find linear patterns.

corresponds to some class. Thus, if you want to estimate the label of a new instance, the algorithm proposes to associate the label of the closest example.

In both cases, an explicit change of the calculation of the distance under the approximation 6.1 must be made. However, we know that in case of using the Gaussian Modular Hash function, and considering $k = 2$, then we have that $\forall\, \mathbf{a},\, \mathbf{b} \in \mathbb{Z}_2^M$

$$d_H(\mathbf{a}\,,\,\mathbf{b}) \quad = \quad \frac{1}{M}\|\mathbf{a} - \mathbf{b}\|^2 \tag{6.2}$$

So in this case it is not necessary to make any adaptation of the algorithm in view that the Euclidean distance between hashes has a direct correspondence with the Hamming distance, and consequently, with the distance between the original points, being a very favorable point for practical purposes. However, as we will see in the next section, we can make a less obvious connection between our methods with more sophisticated algorithms, in particular, Kernel methods.

6.2 Hashing to Compute Kernels

Kernel methods have been widely used in different machine learning tasks, the most popular being the classification task, through the Support Vector Machine (SVM) model. The main idea is to assume that finding a non-linear pattern is equivalent to finding a linear pattern after mapping the data to a larger space. Thus, a data instance $\mathbf{x} \in \mathbb{R}^N$ is mapped using a function φ as $\varphi(\mathbf{x}) \in \mathbb{R}^P$ where $P >> N$.

In general, many linear algorithms are determined by the inner product between examples, which after applying the mapping corresponds to using $\langle \varphi(\mathbf{x})\,,\, \varphi(\mathbf{z})\rangle$ where $\mathbf{x}$ and $\mathbf{z}$ are two instances of data.

Thus, the kernel trick is based on realizing that, in order to recognize non-linear patterns, instead of modeling the function φ, it is more convenient to directly model this internal product in high dimensions.

Therefore, a kernel function will be any function $K(\mathbf{x}, \mathbf{z})$ where there is a function φ such that

$$K(\mathbf{x}, \mathbf{z}) \quad = \quad \langle \varphi(\mathbf{x}) \, , \, \varphi(\mathbf{z}) \rangle \tag{6.3}$$

All of this, without the need of defining the function φ explicitly.

Different Kernel functions have been proposed. Among those that stand out are the shift-invariant kernels, which are those whose value does not change if the data are moved by the same amount. In particular, if the kernel depends on the norm of the difference between the vectors, then it is shift-invariant.

The best known example of this type of kernel is the Radial Basis Kernel, whose expression is given by

$$K(\mathbf{x}_1 \, , \, \mathbf{x}_2) \quad = \exp\left(-\sigma^2 \, \|\mathbf{x}_1 - \mathbf{x}_2\|^2\right) \tag{6.4}$$

A variant of this kernel is known as Laplacian Kernel, which is given by the following expression,

$$K(\mathbf{x}_1 \, , \, \mathbf{x}_2) \quad = \quad \exp\big(- \gamma \, \pi \, \|\mathbf{x}_1 - \mathbf{x}_2\|_1 \big) \tag{6.5}$$

where γ is a parameter of the model.

In this section we will show how to approximate this kind of kernel using hashing methods, and we will discuss the connection with the Modular Hashes, then finalize with an empirical evaluation of this technique.

6.2.1 A hash function to estimate kernels

The two previous kernel functions are closely linked with the Modular Hashes previously studied. To illustrate this we will take the case of Cauchy Modular Hash, but the analysis is analogous to the Gaussian case. To see this connection, it is necessary to introduce a small variant of these hash functions.

Definition 6.2.1 *Let A be an $M \times N$ matrix such that its components are independent and randomly generated according to a Cauchy(γ) distribution, and U be a vector with M random independent components distributed uniformly between 0 and 2. We define the transformation $H_{A,U} \, : \, \mathbb{R}^N \to \{-\frac{1}{\sqrt{M}}, \frac{1}{\sqrt{M}}\}^M$ as*

$$H_{A,U}(\mathbf{x}) \quad = \quad \frac{1}{\sqrt{M}} \, h(A\mathbf{x} + U) \tag{6.6}$$

where h is taken component-wise and it is given by

$$h(t) \quad = \quad 2 \cdot (\lfloor t \rfloor \; (mod \; 2)) - 1 \tag{6.7}$$

The periodic function h of period 2 is illustrated in Figure 6.2 and can take values of either 1 or -1.

Thus, this transformation takes a real value vector with N components and gives a vector with M components, which can be either $\frac{1}{\sqrt{M}}$ or $-\frac{1}{\sqrt{M}}$. This definition is needed to ease the computation between hashes for approximating a kernel. In fact, the following theoretical result shows this,

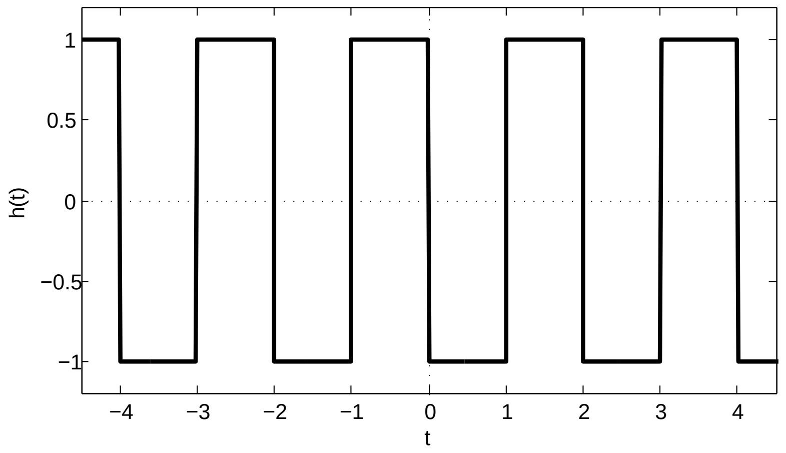


Figure 6.2: The function h quantizes real values to -1 or 1 for each hash component.

Theorem 6.2.1 $\forall \mathbf{x}_1 \,, \mathbf{x}_2 \in \mathbb{R}^N$, *the expected value of inner product between their corresponding transformed points is:*

$$\mathbb{E}\Big(\big\langle H_{A,U}(\mathbf{x}_1)\,,\, H_{A,U}(\mathbf{x}_2)\big\rangle\Big) \;=\; \frac{8}{\pi^2}\sum_{i=1}^{\infty}\frac{1}{(2i-1)^2}\exp\big(-\pi\,\gamma\,(2i-1)\|\mathbf{x}_1-\mathbf{x}_2\|_1\big) \tag{6.8}$$

This expression only depends on the ℓ_1 distance between the original vectors $\mathbf{x}_1$ and $\mathbf{x}_2$. Figure 6.3(a) shows the shape of these curves for fixed values of γ. It is easy to prove that this function is decreasing and tends to 0 when $\|\mathbf{x}_1 - \mathbf{x}_2\|_1$ tends to infinity. In addition, if $\mathbf{x}_1 = \mathbf{x}_2$, the previous expression is 1.

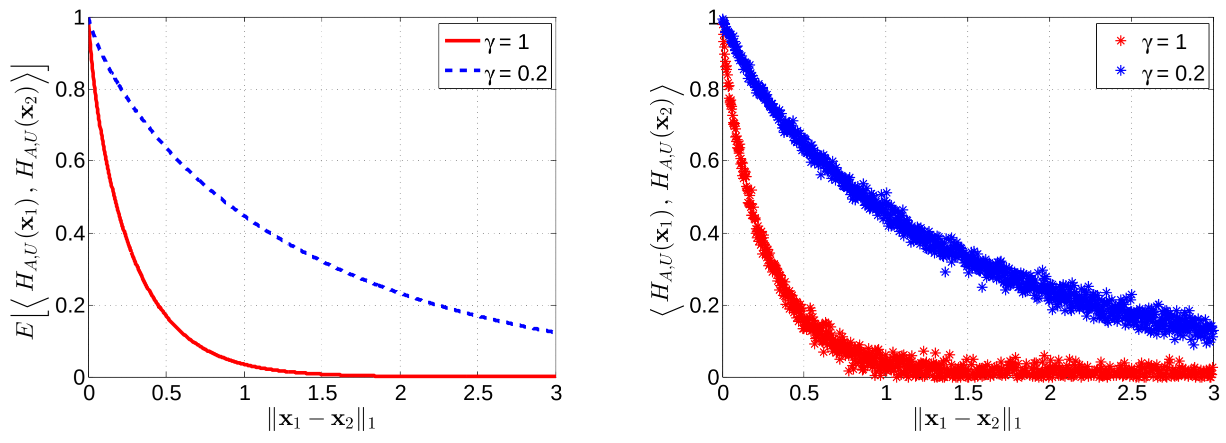
The expression given in Theorem 6.2.1 is not exactly the expression for the Laplacian Kernel described in equation 6.5. However, we can find bounds which depend on this kernel. The following proposition shows an upper and lower bound for the expected value of the inner product between hashes.

Proposition 6.2.1 *The expression given by theorem 6.2.1 can be bounded as follows,*

$$\frac{8}{\pi^2}\exp\big(-\pi\,\gamma\|\mathbf{x}_1-\mathbf{x}_2\|_1\big) \;\le\; \mathbb{E}\Big(\big\langle H_{A,U}(\mathbf{x}_1)\,,\, H_{A,U}(\mathbf{x}_2)\big\rangle\Big) \;\le\; \exp\big(-\pi\,\gamma\|\mathbf{x}_1-\mathbf{x}_2\|_1\big) \tag{6.9}$$

These inequalities show how similar the Laplacian Kernel is compared to the expression given by theorem 6.2.1. We can see that the difference between these two kernels is bounded by $\frac{\pi^2-8}{\pi^2}\exp(-\pi\gamma\|\mathbf{x}_1-\mathbf{x}_2\|_1)$, which means that the difference decreases as $\|\mathbf{x}_1-\mathbf{x}_2\|_1$ increases, converging to 0. Actually, in the worst case, this is when $\|\mathbf{x}_1-\mathbf{x}_2\|_1 = 0$, the difference between these kernels is $\frac{\pi^2-8}{\pi^2} \approx 0.189$.

Nevertheless, both Theorem 6.2.1 and Proposition 6.2.1 are in expectation and eventually the actual result given by the stochastic hashing may differ. However, thanks to the Law of Large Numbers, the Inner



(a) Theoretical Expression (b) Simulations

Figure 6.3: **(a) Theoretical Expression.** Expectation of the inner product between $H_{A,U}(\mathbf{x}_1)$ and $H_{A,U}(\mathbf{x}_2)$ as a function of $\|\mathbf{x}_1 - \mathbf{x}_2\|_1$ using Theorem 6.2.1 considering $\gamma = 1$ and $\gamma = 0.2$. We took the first 5,000 terms of the series. **(b) Simulations.** Inner product between $H_{A,U}(\mathbf{x}_1)$ and $H_{A,U}(\mathbf{x}_2)$ as a function of $\|\mathbf{x}_1 - \mathbf{x}_2\|_1$ with $N = 5000$, $M = 2500$ and $\gamma = 1$ and $\gamma = 0.2$.

Product between $H_{A,U}(\mathbf{x}_1)$ and $H_{A,U}(\mathbf{x}_2)$ converges to the expected value for large values of M. In fact, at the moment of computing the inner product between two hashes, we are performing an operation which is a sum of i.i.d. random variables, divided by M, to obtain a simple average of random variables. Figure 6.3(b) shows a simulation, where the empirical value is plotted as a function of the actual ℓ_1 distance between vectors. We see here that the empirical estimate closely follows the plot for the theoretical expression in figure 6.3(a).

Similarly to the case of distance estimation, a more formal guarantee is given by theorem 6.2.2 based on Hoeffding's inequality, where the probability of having a small error estimation depends on the value of M and the number of points to transform.

Theorem 6.2.2 *If we have n points $\mathbf{x}_1, ..., \mathbf{x}_n \in \mathbb{R}^N$, and $M \geq \frac{4}{\varepsilon^2}\log\left(\frac{n}{2\eta}\right)$, then*

$$\mathbb{P}\Big(\forall \mathbf{x}_i, \mathbf{x}_j\,,\,\Big|\langle H_{A,U}(\mathbf{x}_i)\,, H_{A,U}(\mathbf{x}_j)\rangle - \mathbb{E}\left(\langle H_{A,U}(\mathbf{x}_i)\,,\, H_{A,U}(\mathbf{x}_j)\rangle\right)\Big| < \varepsilon\Big) \quad \geq \quad 1-\eta \tag{6.10}$$

This result gives a way to measure the quality of the approximation made by the hashes. In particular, if η and ε are fixed, the number of components needed in the hashing function depends on the log of the number of points n that we want to transform. Moreover, note that the value of M does not depend on the dimensionality of the input data. Both facts may have important implications when we work with high dimensional data.

6.2.2 Connection with Modular Hash

As we have seen, the hash function $H_{A,U}$ takes a real valued vector $\mathbf{x}$ and outputs a vector with M components whose values can be either $-\frac{1}{\sqrt{M}}$ or $\frac{1}{\sqrt{M}}$. This means that even though M may be smaller than the original data dimension, the final representation of each hash component is a float number.

However, we can recode this hash function using the Modular Hash function we have already studied. In fact, we can note that

$$Q_{2,A,U}(\mathbf{x}) \quad = \quad \frac{1}{2}\Big(H_{A,U}(\mathbf{x}) + 2\Big)$$

Therefore, there is a deterministic relation between the hash $H_{A,U}$ and our modular hash. This means that all the privacy guarantees that the modular hash has are transferred to this new hash.

Moreover, it is quite straightforward to prove that the inner product $H_{A,U}(\mathbf{x}_1)$ and $H_{A,U}(\mathbf{x}_2)$ can be recovered using $Q_{2,A,U}(\mathbf{x}_1)$ and $Q_{2,A,U}(\mathbf{x}_2)$. Indeed, we have the next equation

$$\langle H_{A,U}(\mathbf{x}_1)\,,\, H_{A,U}(\mathbf{x}_2)\rangle \quad = \quad 1 - 2 \cdot d_H\big(Q_{2,A,U}(\mathbf{x}_1)\,,\, Q_{A,U}(\mathbf{x}_2)\big) \tag{6.11}$$

Thus, we can recover the kernel approximation using the Normalized Hamming distance between hashes. However, for this purpose we prefer to use the hash function $H_{A,U}$ since there is no need to adapt any algorithms that work directly with inner products. We just need to transform the data and run the algorithm on it.

Indeed, SVM models are usually trained using its dual form that naturally leads to the kernel trick; its dual form only depends on the inner product between pairs of input vectors. The kernel trick implies the computation of the Kernel matrix using the input features and a non-linear kernel. However, such computation scales poorly because its time and storage complexity are quadratic in the dimensionality and number of the input features. Hence, we can apply our hashing scheme to the SVM. First, we take the input features and generate our hashes as described in the previous sections. Then, we use the hashes to compute the kernel matrix with the inner product, which is a linear operation. This combination approximates a non-linear Laplacian kernel SVM. Formally

$$K(\mathbf{x}, \mathbf{y}) \quad = \quad \langle \phi(\mathbf{x}), \phi(\mathbf{y})\rangle \quad \approx \quad \langle H(\mathbf{x}), H(\mathbf{y})\rangle \tag{6.12}$$

We can name this procedure *Hashing Trick*.

6.2.3 Empirical Evaluation

To show the benefits of using hashing with a linear SVM over using the input features and a non-linear SVM with Laplacian kernel we present an evaluation in the Acoustic Scene Classification task, where the

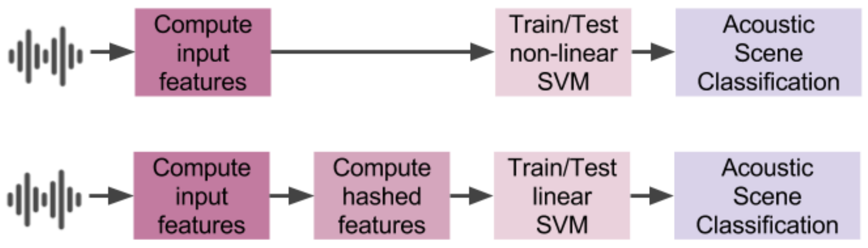


Figure 6.4: The audio recordings are used to extract input features. Then, these features are used to train an SVM in two different ways. One is to pass the features directly to a non-linear SVM, second is to compute hashes and pass them to a linear SVM. Lastly, the trained SVM is used for classification.

objetive is to identify an audio recording as belonging to a predefined set of scene classes that characterizes an environment, for example, *park*, *home*, or *office*.

The experiments presented are in the context of the DCASE Task 1 - Acoustic Scene Classification [68]. The pipelines of the experiments are also illustrated in Figure 6.4.

Acoustic Scenes Data Set

For our experiments we used the "DCASE: TUT Acoustic Scenes 2017" development dataset [68]. It consists of recordings from various acoustic scenes of 3-5 minutes long. The 15 acoustic scenes are: *Bus, Cafe / Restaurant, Car, City center, Forest path, Grocery store, Home, Lakeside beach, Library, Metro station, Office, Residential area, Train, Tram, Urban park.*

Audio Feature Extraction

A typical approach in this task is to capture diverse characteristics from the audio signal by computing different types of features. We extracted the large set of audio features of 6,553 dimensionality employed in [69, 70]. The set includes different features to extract different relevant information from the acoustic scenes, which consist of multiple sound sources. The set is extracted using the toolkit openSMILE [71] with the configuration file *emolarge.conf*. The features are divided in four categories: cepstral, spectral, energy related and voicing features, and are extracted every 10 ms from 25 ms frames. Moreover, we included functionals, such as mean, standard deviation, percentiles and quartiles, linear regression functionals, or local minima/maxima. All these features played the role of input data to train a classifier.

Acoustic Scene Classification

First, we evaluate the input features using the non-linear SVM with the Laplacian kernel. Then, we evaluate the hashes with a linear SVM to approximate the baselines. Lastly, we include from our previous work [7]

Method \ γ	2^{-12}	2^{-14}	2^{-16}	2^{-18}	2^{-20}
Laplacian Kernel	71.21	77.91	**78.64**	78.51	78.51
Hash ($M = 2^6$)	19.80	40.59	37.75	23.71	13.00
Hash ($M = 2^8$)	31.28	53.95	50.96	44.00	28.94
Hash ($M = 2^{10}$)	49.85	68.44	66.28	58.11	46.95
Hash ($M = 2^{12}$)	64.45	74.86	75.18	70.51	64.92
Hash ($M = 2^{14}$)	68.57	76.06	76.31	75.93	72.90
Hash ($M = 2^{16}$)	69.70	76.72	**77.55**	77.38	76.21

Table 6.1: Accuracy performance (%) as γ varies for the Laplacian kernel (with input features), Series kernel (with input features) and different hashing sizes M in bits (with linear SVM). The best results for each type are in bold. Note how as the value of M increases the performance is better approximated.

the random features with a linear SVM to approximate the Laplacian kernel.

We trained SVM classifiers with a one-vs-all setup and tuned the γ parameter. For each experiment we used the feature vectors of the testing set to evaluate performance and measured it with accuracy. As a baseline model, we trained a SVM model with Laplacian kernel using the described 6,533 features components as input.

Then, we used our hashing scheme and tried different hash sizes (M) to compare against the baseline performance. Hence, we used the training and testing acoustic feature vectors of 6,553 dims to compute hashes. The size of the hashes was varied from $M = 2^6$ to $M = 2^{16}$ bits. The hashes were used to train the SVM with a linear kernel.

Results on Acoustic Scenes Classification

For the model with Laplacian kernel we obtained a range of accuracy results from 71.21% to 78.64%, depending on the value of γ, as it is shown in table 6.1.

The accuracy results of the hashes improved as the size M increased to a number close to the baselines. Similar to the baseline experiments, the performance varied depending on the value of the parameter γ. For the best performing hash and largest size (2^{16}) the range was 69.70% ($\gamma = 2^{-12}$) to 77.55% ($\gamma = 2^{-16}$). Table 6.1 shows these values.

Furthermore, we can observe that after applying the hashing technique and training a linear model on the hashes we have a minimal loss of performance with respect the original kernel method. Moreover, we have the computational advantage of working with a linear problem instead of a non-linear quadratic one.

Significant findings are related to the size of the output obtained from our hashing function. In this experiment, each input sample is represented by a vector of 6,553 components, which is more than 2^{18} bits if each component is represented by 8 bytes. In contrast, to achieve similar results with our hashing technique, we need 2^{16} bits, that is, a factor of 4 reduction in size. Hence, this method not only changes a nonlinear

Method	Accuracy	# of Bits
DCASE Challenge [68]	74.8%	-
Laplacian Kernel	78.6%	$> 2^{18}$
Random features $M = 2^{12}$ [7]	75.8%	2^{18}
Hashing $M = 2^{16}$	77.5%	2^{16}

Table 6.2: The performance of the hashing scheme outperforms the reported Challenge score and the random features. Moreover, performance is comparable to using the Laplacian Kernel, however with the benefits of reducing the bit representation by 4 times.

problem to a liner one, but also reduces the dimensionality of the feature vector.

We can compare these techniques with other methods. Particularly, in table 6.2 we include the official DCASE performance based on a Multilayer Perceptron, our baselines, our hashing scheme and the random features [7].

Random features are a similar technique, which allowed us to approximate the Laplacian kernel with an accuracy of 75.8%. Even though the dimensionality of the random features was 3 times lower than the original vector, each random feature component used 2^{18} bits if each component is represented by 8 bytes. In contrast, using the hashing technique, with 2^{14} bits we obtain a performance of 76.31%. Hence, we can reduce the size of the final representation by 16 times and obtain comparable results. Actually, with a small loss of performance, we may use hashes with 2^{12} bits and obtain an accuracy of 75.18%, reducing the size of the final representation by 64 times or 6 orders of magnitude.

6.3 Related Work and Contributions

Through appropriate choice of kernel, SVMs can approximate non-linear functions or decision boundaries given enough training samples, however, the computation of the kernel matrix scales poorly. The Kernel matrix's time and storage complexity increase in the dimensionality and number of the inputs, which contrasts to linear functions, which can be computed much faster [19]. One solution to take the best of both approaches is by computing random features [21, 28, 29], which maps the input features into a randomized lower-dimensional feature space. Then, the resulting random features are combined in a linear manner to approximate a non-linear kernel, but at a much lower cost than direct computations of kernels.

One effective method for quantizing features is through *hashing* schemes [20]. Hashing schemes, roughly speaking, quantize the random features to be represented with just one bit instead of real numbers represented by 32 or 64 bits. This has significant implication in storage, because even if the dimensionality of the random features is the same or larger than the input features dimension, the overall size in bits of the transformed vector can be an order of magnitude smaller. This reduction has benefits in storage, processing and transmission [20].

Random features and hashing schemes are kernel dependent; this means that to approximate different non-linear kernels we need different kinds of random transformations of the input features. The most popular and best-studied kernels are shift-invariant [22, 23], such as Guassian, Laplacian and Cauchy. The Gaussian employs the ℓ_2 norm and the Laplacian employs the ℓ_1 norm, and although both offer different benefits, the latter has been significantly less studied. The Laplacian kernel, in particular, has outperformed the Gaussian kernel without hashing schemes for sound-event classification [24, 25] and with hashing schemes for image processing [26, 27]. However, to the best of our knowledge, the mathematical and theoretical guarantees to employ hashing schemes to approximate Laplacian kernels was unavailable in the literature.

In conclusion, we have introduced a scheme that hashes features, which are combined with a linear function to approximate a non-linear Laplacian kernel. Additionally, the hashing reduces the dimensionality of real-valued vectors into bits-based representations by reducing storage up to *six orders of magnitude*. We evaluated our hashing scheme in the context of the 2017 DCASE Task 1 - Acoustic Scene Classification [68]. Our main contribution are the mathematical and theoretical guarantees to validate the approximation of the Laplacian kernel, which was unavailable in the literature and can be applied to tasks other than audio.

Part IV

Applications

Chapter 7

Speech Signal Protection

7.1 Introduction

Certainly, the most natural communication tools used by human beings are their voices. Hence, it is expected that a lot of research has been devoted to analyzing and understanding speech signals for several applications. One of them is *Speaker Authentication*, where the objective is to determine or verify the identity of the speaker. Other applications include *Automatic Speech Recognition*, where the goal is to transcribe a recorded speech signal into its corresponding sequence of words.

One way to implement these applications is based on using a set of template recordings, where finding a word or identifying a speaker can be made through a distance computation between features of the query signal and each template in the database. Since these databases are frequently large, external cloud-based repositories are used and, as a consequence, the corresponding distance computation is made in the cloud.

However, speech data may carry many privacy concerns. Eventually, we could infer private information from speech, such as gender, age, ethnicity, education and emotional states. Thus, the server could extract this information from templates and queries, and then use it for undesired purposes. Moreover, the recordings may be edited in order to create fake speech that the client never spoke. Furthermore, authentication techniques can be applied to identify the presence of a user in other media such as YouTube or Flickr videos.

Hence, we need a mechanism that enables signal comparison and at the time hides private information to the server without losing performance. In particular, we are interested on speaker authentication through distance computation based on templates. In our context, this equates to determining whether a speaker in a recording matches a target speaker in an example recording, without revealing any information about the signals to the server.

Thus, the private mechanism described in this work may help to solve these privacy concerns as well as allowing comparison between signals, which in this problems means to verify an identity.

Just to provide some context, first we will show a brief description of the tools used in speaker authentication. In general, these kinds of systems do not use any kind of mechanism to protect the speech information. Then, we will describe some of the properties that a protected system must have according to the literature, and finally we will show that the transformation we have presented, the Gaussian and Cauchy modular hash function, satisfy similar properties.

7.2 Speaker Authentication Basics

As we mentioned previously, in most of the speech processing applications we want to extract information from a signal and make an inference about it. In the *Speaker Verification* problem, for example, the purpose is to determine if the user is indeed who she or he claims to be.

In general, this kind of system does not work directly from the speech signal, because audio recordings comprise sequences of samples of varying length. Therefore, if we want to apply a common classifier for verification, the signals must be converted to fixed-length "feature" vectors. To perform this conversion, we employ a representation known as *Gaussian mean supervectors* [17]. Each recording is first parametrized into a sequence of "Mel-frequency cepstral coefficient" (MFCC) vectors [18]. Each vector in the sequence is augmented by the first- and second-order differences between its immediate neighbors, resulting in a sequence of 39-dimensional feature vectors. The distribution of the feature vectors in each recording is modeled by a Gaussian mixture model, learned using a maximum *a posteriori* estimator. The means of the Gaussians are then concatenated to result in the desired supervector, which are considered as feature vector of a recording.

On the other hand, in the client-server model, the speaker verification system is in the server and the client executes a program which computes the features and sends them to the server. Speaker verification has two phases: the **enrollment**, where the user submits a few samples of recordings or feature vectors to the system, and **verification**, where the user submits a test feature vector and the system makes an acceptance decision based on its similarity to the enrollment data. Therefore, in this systems it is necessary that the server contains some templates and receives queries in order to compare them, generating the privacy concerns previously described. Obviously, this scheme can be used for any biometrics signal. Figure 7.1 summarizes both phases.

Here, we use an authentication framework that uses a *nearest neighbor* classifier to perform authentication [38] [40] [44]; as we shall see, even this simple mechanism suffices to provide near-perfect authentication performance in our two-factor setting, that is properly defined below. In the enrollment phase the user u submits a set E_u of supervectors to the server. During verification, the subject claims identity u and submits

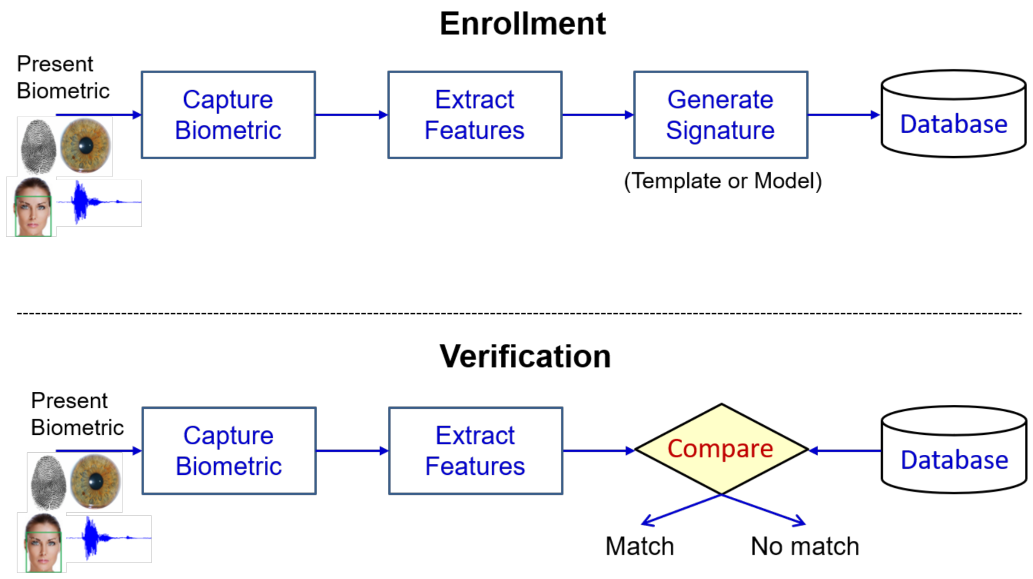


Figure 7.1: Enrollment and verification processes in a biometric system.

a single test recording supervector $\mathbf{x}$. The system computes

$$d(\mathbf{x}\,,\,E_u) \;=\; \min_{\mathbf{y}\in E_u} D(\mathbf{x}\,,\,\mathbf{y}) \tag{7.1}$$

where D is some metric to compare feature vectors. The subject is successfully authenticated if $d(\mathbf{x}\,,\,E_u)\;\leq\;t$, for a predefined threshold t. On the other hand, if $d(\mathbf{x}\,,\,E_u)\;>\;t$, then the subject is rejected. In our work we specifically use the ℓ_2 or ℓ_1 distance to compare vectors. Thus, the server takes a decision based on

$$d(\mathbf{x}\,,\,E_u) \;=\; \min_{\mathbf{y}\in E_u} \|\mathbf{x}\;-\;\mathbf{y}\|_1 \tag{7.2}$$

7.3 Cancelable Biometrics

It is clear that the described setting applies for any kind of biometrics; for example, fingerprints or face recognition, among others. An important problem that is common to all biometric authentication systems, including speech, is that biometrics are unique and *permanent*. A compromised biometric is compromised for ever. Unlike conventional written passwords which can simply be replaced if compromised, once a biometric is compromised it can never be used again.

In order to solve these problems, researchers have introduced the concept of *Cancelable Biometrics* [32] where the client *transforms* the biometric signal (or feature vectors derived from it) before sending it to the server. The server only receives the modified signals that nevertheless retain the information required for authentication. A number of different types of transformations have been proposed, with different properties, advantages and disadvantages [33] [42] [44]. The approach provides the biometric with two desirable properties. First, when properly performed, the transform masks the biometric itself and prevents the inference of undesired information from it. Secondly, the biometric signatures stored by the server now become *cancelable* – if compromised to an outside adversary the client only needs to change the transform used to modify the signal. The original biometric itself stays secure.

Of particular interest to us are *Two Factor Transformations* (TFT) [34] which transform the biometric through a secret parameter, known as a *key*. The secret key itself can be generated using a text password provided by the user, which can seed a random number generator [35]. The system must now work with the TFT-modified signals in order to perform authentication. Ideally, using the correct password/key to transform signals, the system performance on transformed signals must be comparable to that obtained with the original unprotected signal. If the identity claimant is unaware of the correct key, however, the system must absolutely reject the authentication. If properly performed, in addition to the benefits mentioned in the earlier paragraph, this technique adds a further security measure: it effectively converts the biometric authentication to a two-factor process. In order to be authenticated the client must possess both, the correct biometric and the correct password.

Ideally, the TFT “hides” the true biometric from the server which cannot recover the original biometric from the transformed signal. However, simply hiding individual signals may be insufficient: information may nevertheless be gleaned through the comparisons of transformed signals and templates about the general distribution of distances between signals, which may in turn be used to learn about the biometric itself. Thus, an additional desideratum for TFTs is that *comparison* of TFT-transformed signals must also be uninformative. This requirement however is opposed to the fact that authentication is based on comparison of signals and templates. Most solutions thus strike the compromise that they do not necessarily guarantee the uninformativeness of comparison, relying mainly on the privacy of the password/key for security.

To formalize the concept of TFT, we can consider that it is a one-way function that considers as input the feature $\mathbf{x}$ and a user specific key $\mathbf{K}$, to generate a transformed version $T(\mathbf{x}, \mathbf{K})$. As stated by Chikkerur *et al.* [44], if D is a metric to compare features, a TFT should satisfy the following properties:

1. *Local Similarity Preservation.* If two features belong to the same user, they should be similar, so the transformation must preserve similarity, i.e.

$$D(\mathbf{x}_1\,,\,\mathbf{x}_2) < t \quad \Rightarrow \quad D(T(\mathbf{x}_1, \mathbf{K})\,,\,T(\mathbf{x}_2, \mathbf{K})) < \theta \tag{7.3}$$

2. *Mismatch must persist.* If two features vectors do not belong to the same user, the distance between them should be large, so the transformation does not make mistmatched vectors closer. Formally, this

means

$$D(\mathbf{x}_1\,,\,\mathbf{x}_2) > t \quad \Rightarrow \quad D(T(\mathbf{x}_1, \mathbf{K})\,,\,T(\mathbf{x}_2, \mathbf{K})) > \theta \tag{7.4}$$

3. *Uninformativity.* The transformed version of the feature vector should not match with the original signal, this is

$$D(\mathbf{x}\,,\,T(\mathbf{x}, \mathbf{K})) \quad > \quad t \tag{7.5}$$

4. *Unlinkability.* Two transformation of the same vector using different keys must not match. Formally, this is

$$D(T(\mathbf{x}, \mathbf{K}_1)\,,\,T(\mathbf{x}, \mathbf{K}_2)) \quad > \quad \theta \tag{7.6}$$

Note that, according to these properties, to not have any inconsistency with the metric D, both the biometric $\mathbf{x}$ and the transformed biometric $T(\mathbf{x}, \mathbf{K})$ must belong to the same space.

In addition, any authentication system must be resistant to imposter claims. Under this model, where to perform an authentication we need both a feature vector and the secret key, we consider the following attacks:

1. An attacker does not get access to either a valid feature vector $\mathbf{x}$ or the key $\mathbf{K}$
2. An attacker has access to a feature vector only.
3. An attacker has access to the secret key only.

The Two factor transformation should be resistant to all these attacks. Note that, if the attacker gets both elements, a compromised feature vector $\mathbf{x}$ and the secret key $\mathbf{K}$, it is not possible to prevent an attack.

7.4 Limited Leakage Transformation as TFT

We demonstrate that our approach, Limited Leakage Transformations, can be used to protect biometrics, in particular speech features, satisfying analogous properties to the described above.

In fact, we can adapt a speaker verification system using our methods. For the enrollment phase, instead of sending raw supervectors $\mathbf{x}_1, ..., \mathbf{x}_n$ to the server, a user must create a password, fixing an integer value k, and use it as a seed to generate a matrix A and a vector U, according the suggested distributions to compose a Gaussian or Cauchy Modular Hash. These two parameters work as a secret key, allowing him to send $Q_{k,A,U}(\mathbf{x}_1), ..., Q_{k,A,U}(\mathbf{x}_n)$ to the server. In the verification phase, the user recovers the key using the same password and, instead of submitting the supervector $\mathbf{x}$, submits $Q_{k,A,U}(\mathbf{x})$. Then, to make a decision, the server computes the Normalized Modular Distance between the transformed test supervectors and the transformed enrollment vectors as given in Section 7.2 to authenticate the user.

Finally, we note that the relationships we derive are statistical in nature. This is to be expected from any TFT mechanism that makes claims relating to information leakage. Hence, to conclude this section, we can now reformulate the properties of good TFTs presented in probabilistic terms as below, and note that the proposed transform satisfies all of them.

1. *Local Similarity Preservation.*

$$\|\mathbf{x}_1 - \mathbf{x}_2\| < t \quad \Rightarrow \quad \mathbb{E}\Big[d_{mod}\Big(Q_{k,A,U}(\mathbf{x}_1)\,,\, Q_{k,A,U}(\mathbf{x}_2)\Big)\Big] < \theta \tag{7.7}$$

This means that the Normalized Modular Distance between transformed points using the same secret key reveals information about the distance between the original points as long as these are close enough.

2. *Mismatch must persist*

$$\|\mathbf{x}_1 - \mathbf{x}_2\| > t \quad \Rightarrow \quad \mathbb{E}\Big[d_{mod}\Big(Q_{k,A,U}(\mathbf{x}_1)\,,\, Q_{k,A,U}(\mathbf{x}_2)\Big)\Big] > \theta \tag{7.8}$$

In other words, if the distance between the original points is larger than some threshold, the Normalized Modular Distance should be large too. In our case, not only is it large, it will also be uninformative, i.e. not provide any information about the true distance between the original vectors.

3. *Uninformativity.* For any $\mathbf{i} \in \mathbb{Z}_k^M$ and for any $\mathbf{x} \in \mathbb{R}^N$,

$$\mathbb{P}\Big(Q_{k,A,U}(\mathbf{x}) = \mathbf{i}\Big) \quad = \quad \frac{1}{k^M} \tag{7.9}$$

Thus, the probability of obtaining a particular hash value does not depend on the original point, i.e., a hash itself does not reveal information about the original vector. Note that this is different from the uniformativeness of the distance between mismatched hashes, mentioned above.

4. *Unlinkability.* If (k, A_1, U_1) is independent of (k, A_2, U_2), then

$$\mathbb{E}\Big[d_{mod}\big(Q_{k,A_1,U_1}(\mathbf{x})\,,\, Q_{k,A_2,U_2}(\mathbf{x})\big)\Big] \quad = \quad \frac{k}{4} \quad > \quad \theta \tag{7.10}$$

This means that, even if we transform the *same* vector using independent keys, the resulting hashes will be independent. As a consequence, thanks to the Limited Leakage Property, we cannot distinguish the hashes of a pair of distant vectors from two hashes of the same signal using independent keys.

7.5 Empirical evaluation

To illustrate the scope of oue technique, we applied our proposed technique to the speaker verification task using a Cauchy Modular Hash with $k = 2$.

We ran experiments on the YOHO Speaker Verification corpus [16], comprising utterances by 138 speakers. Each utterance contains a spoken sequence of three two-digit numbers. The corpus is split into an enrollment set (96 utterances per speaker) and a verification set (40 utterances per speaker). In order to develop a speaker verification system, we require both recordings from target speakers and a large number of recordings from attackers. In our experiments, however, we did not explicitly record imposters; instead, for each of the 138 speakers, the remaining 137 were used as imposters. All experiments were conducted using the verification approach described in Section 7.2; thus, a subject is accepted in the verification phase if the ℓ_1 distance between the verification query and some element in the corresponding enrollment set is smaller than a threshold.

The first row of Table 7.1 shows the baseline performance obtained using supervectors of the speech recordings directly, without any transformation. The resolution of the supervector depends on the number of Gaussians in the GMM used to model the distributions of the MFCC vectors in the recordings. In fact, the performance improves as the number of Gaussians increases.

The second row shows the performance of the system using our transformation under the ideal condition, where the imposters have neither the key nor the biometric. In this setting the system is perfect, making no mistakes at all. Thus, combining a speech signal with a password using the TFT transform increases the performance of the system. In addition, notice that the server sees neither the password nor the actual supervectors, and thus learns nothing about the speaker.

In adverse scenarios, the performance is not compromised. The third row of table 7.1 shows a system where the adversary has managed to obtain a valid biometric signal from the user, but does not have the secret key. We assume that, under the lack of knowledge about the secret key, the adversary generates an independent key and uses it to transform the compromised biometric. Once again, the system is perfect. Finally, the fourth row shows the scenario where an adversary has managed to steal the user's key, but does not have his biometric. We note only minimal degradation attributable to the TFT with respect to the unprotected baseline. The explanation for this degradation lies in the randomness introduced by the hashes.

In all cases we computed no more than 64 bits in the transformation per component of the supervector. In general, we observed a trade-off between the number of bits per components and the performance in this last task; having fewer bits can be useful for storage and computation, but the performance can be compromised with smaller representations.

Experiment / # gaussians	4	8	16	32
Baseline	5.66	3.21	1.99	1.27
Transform	0	0	0	0
Stolen Voice	0	0	0	0
Stolen Key	5.80	3.40	2.23	1.57

Table 7.1: Experimental Results. EER(%) for each scenario, considering different number of gaussians. We took $\gamma = 0.02$

7.6 Conclusion

In this chapter we have discussed how to use our proposed method as a Two Factor Transformation to protect speech data for speaker verification tasks, which converts speaker verification into a two-factor task. In addition, and more importantly, it *hides* the speech data from both the system and adversaries, protecting the privacy of the user. The proposed TFT allows us to obtain an acceptance response if the distance between two feature vectors of the biometric speech signals is smaller than a threshold without revealing the actual speech data, while providing information-theoretic security at larger distances to protect the biometric. Also, this is a *cancelable* framework where, if the user's account is compromised on the server, he or she only needs to change his key to re-enroll since the actual biometric remains private. Moreover, our experiments also show that this transformation improves the accuracy of the system in all usage scenarios where the adversary does not possess the secret key, while only affecting performance minimally in the most compromised situations.

Chapter 8

Private Image Retrieval

8.1 Introduction

It is common for cloud users to require content-based access to their data. For instance, they may desire to retrieve from the cloud a record that matches a given key or query. To perform this retrieval, the system must compare the query to the contents of the stored data, identify relevant records, and return them to the user. This, however, requires the server to be able to inspect the contents, immediately granting it access to the data, and in consequence, raising some privacy concerns for users. In fact, even if the server is to perform any organization of the data on its own, it requires access to the contents of the data, thereby violating eventual privacy requirements of the user. More generally, if the user requires to retrieve a ranked list of records in response to a similarity-based query, the necessary computation would require exposure of the data to the server.

To protect the data we may apply conventional encryption functions, but they destroy neighborhood by design; if they do not, they become susceptible to differential cryptanalysis. Thus, given any cleartext message x, its encryption $E(x)$ provides no information about x itself, unless the encryption keys are known. It also destroys all the information for similarity search. This is particularly relevant for multimedia files, where a matching between files is based on a fuzzy similarity score instead of a strong match. In the case of image retrieval, it is usual to consider a similarity-based approach, where different informative features are extracted from each image file, and two images are deemed to be similar if their corresponding features are close enough. Thus, typically an image retrieval system operates in two phases, an indexing phase, where the images of a database are organized according to the similarity provided by the closeness between feature vectors, and a retrieval phase, where given an image query we identify the most similar images in the database comparing the feature vectors from the database to the feature vector obtained from the image

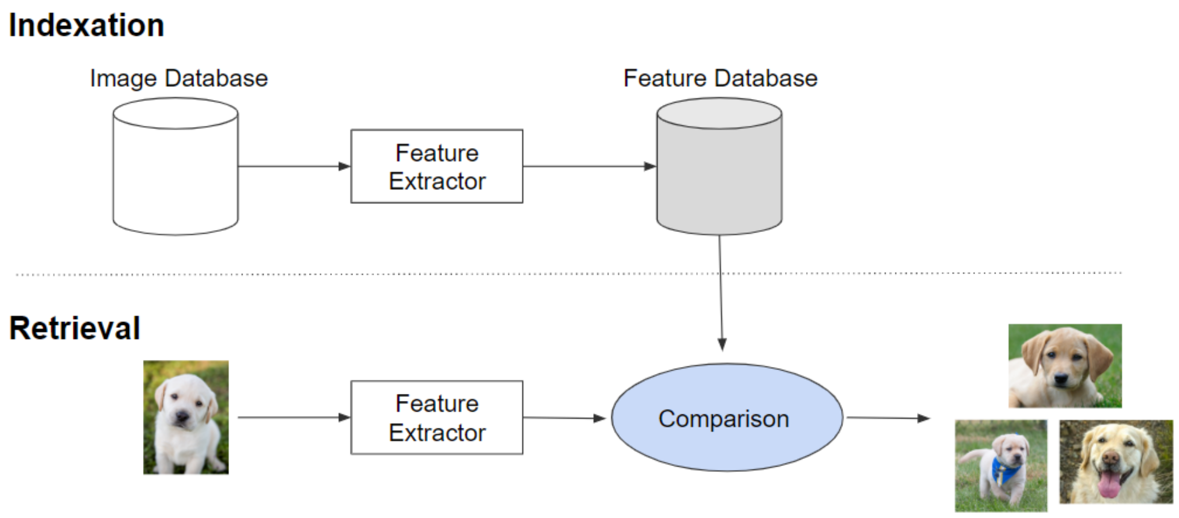


Figure 8.1: Standard design of an Image Retrieval System.

query.

Note that under this system design, generally both the image database and feature database are located in the cloud, and therefore both images and features are unprotected. However, if we assume that the features of the images can be computed locally before being sent to the cloud, then we can obtain a minimum of security by uploading the images encrypted to the cloud, along with their retrieval-relevant features, which are in the clear. Thus, the client can retrieve encrypted images by comparing the corresponding unprotected feature vectors, and then she or he decrypt the encrypted file using a secret key. Despite this security procedure, as expected, the feature vector can reveal information about the content of the image, generating potential privacy risks.

We consider that our proposed approach can contribute to address this problem. The overall solution is very straightforward using the kind of transformation we have studied in this thesis, as we detail in the following section.

8.2 Limited Leakage Transformation for Retrieval

To address the problem of private image retrieval, we believe the utility of our proposed Limited Leakage transformations is immediately obvious. Given a collection of images $x_1, x_2, \cdots, x_n$, we store in the cloud the encrypted image $E(x_i)$ along with the hash $Q_R(f_i)$, where f_i corresponds to the feature vector extracted from the image x_i. The key R for the hash remains with the user; given the security guarantees shown in chapter 4, the hashes in isolation provide minimal information about the data. In order to retrieve records based on any image query q, we compute its feature vector f_q and we transmit the hash $Q_R(f_q)$ to the cloud.

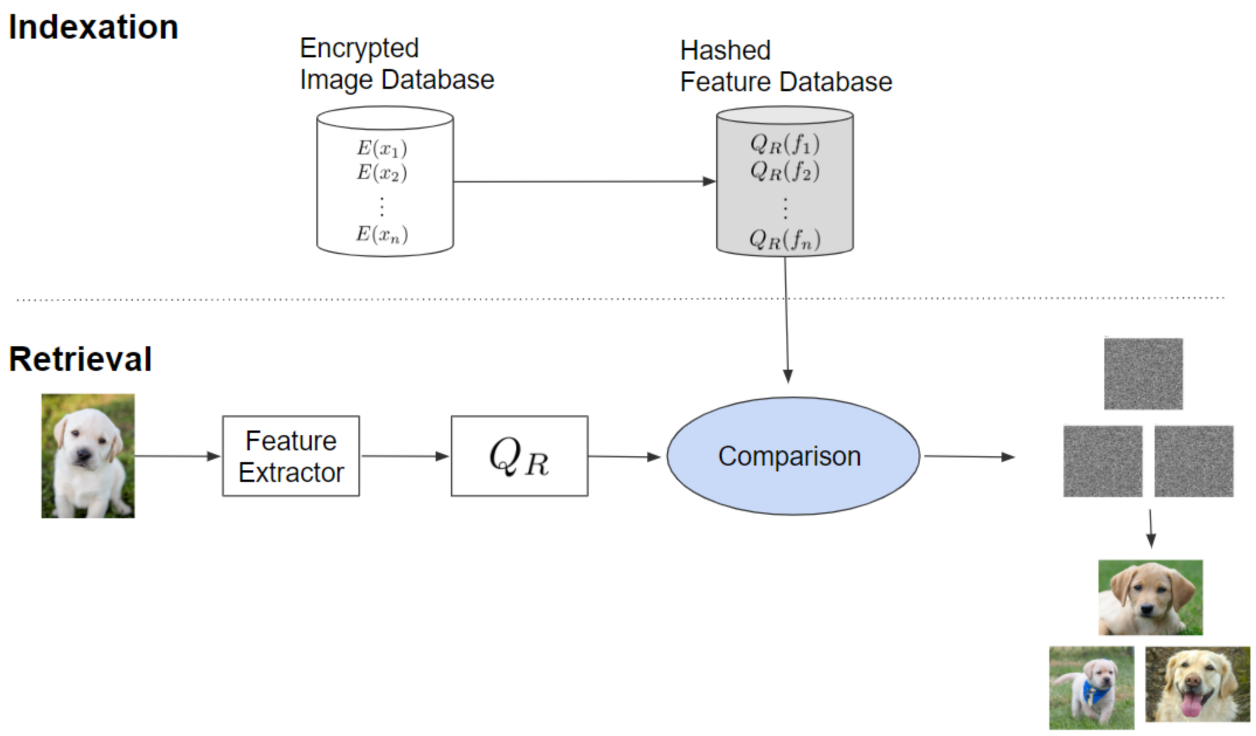


Figure 8.2: Design of a proposed image retrieval system that promoted privacy.

Here, the cloud may find close matches, those that fall within δ of the query, by comparing $Q_R(f_q)$ and the hashes $Q_R(f_i)$ for the image records. Instances which fall within it may be ranked and retrieved.

Moreover, the hashes have the property that they are very fast to compute and the distance computation between the hashes comprises the computation of a Hamming distance, which is also trivially fast to calculate. By appropriately relaxing δ, they also provide means of effecting a trade-off between privacy and computational efficiency.

These ideas have been implemented in an actual image retrieval system [53]. Here, our work is used as a key component of the system in a similar way as is described in figure 8.2. To improve search efficiency, K-means algorithm is used to construct a tree-index, showing comparable results with the unprotected scheme both in search accuracy and speed. In fact, using the tree structure and the hamming distance between hashes it is possible to obtain a cost in search smaller than linear orders of computations.

8.3 Conclusions

In this chapter we have presented an application on image retrieval with privacy constraints. Based on the guarantees presented in the previous chapters, we know that our proposed transformations helps to hide the information provided by a feature vector but preserving similarity relationships if these are strong enough.

These properties fit very well for this problem, where we are just interested on identifying the most similar points for a query. Even though there are other techniques, such as Locality Sensitive Hashing, which allows to speed up this kind of search, they do not have the privacy requirements that we have shown as is is discussed in [1]. Indeed, this kind of hash function do not satisfy either the indistinguishability property nor the limited leakage property or something similar.

Chapter 9

Two-party Computation for Distance Inference

9.1 Introduction

In this chapter we want to show that the particular construction of Gaussian Modular Hash may help us to solve a two party computation problem for distances. In particular, consider two parties, Alice and Bob, who have two real valued vectors $\mathbf{x}_1$ and $\mathbf{x}_2 \in \mathbb{R}^N$ respectively, with a large value of N. One or both parties require the computation of $\|\mathbf{x}_1 - \mathbf{x}_2\|$. However, at the same time, it is required that none besides Alice learns about $\mathbf{x}_1$, and none besides Bob is exposed to $\mathbf{x}_2$.

Different methods have been suggested to solve this problem. These have generally involved the use of secure multi-party computation protocols (GCS) [8] or fully homomorphic encryption [9, 14] which enables the computation of distances over encrypted data. As we already know, even though this provides the participants with the desired privacy, they could be computationally unattractive.

An alternate approach uses *trusted third parties* to facilitate the computation. Here, Alice and Bob interact with a third party, Charles, to compute the distances between their signals with *information theoretical privacy*. No party gets additional information regardless their computational power. Under this approach, protocols based on secret sharing using polynomials on finite fields have been proposed to compute the Euclidean distance [30] between privately held real-valued vectors. While secure, the precision of the computation depends on how well real numbers are represented in the selected field. Moreover, the communication complexity of this kind of protocol is proportional to the dimensionality of the data, which is problematic when Alice and Bob need to compute a distance between high dimensional vectors in the presence of communication constraints. Moreover the third party itself is to be fully trusted, a questionable condition in most practical scenarios.

We consider that the hashes we proposed can provide an advantage to solve this problem. The key observation which helps to address these drawbacks is the fact that the joint probability function of two Gaussian modular hashes just depends on the Euclidean distance between the actual points.

$$\mathbb{P}\Big(Q_{k,A,U}(\mathbf{x}_1) = i\,,\, Q_{k,A,U}(\mathbf{x}_2) = j\Big) \quad = \quad f\big(i\,,\, j\,,\, \|\mathbf{x}_1 - \mathbf{x}_2\|\big) \tag{9.1}$$

So, if a third party just observes the corresponding hashes of $\mathbf{x}_1$ and $\mathbf{x}_2$, then, the only information possible to infer the is Euclidean distance between points. This helps us to introduce the concept of *Somewhat Third Trusted Party*.

9.2 Somewhat Third Trusted Party

Suggesting the use of a third party is always controversial, because once we send a piece of information to it, we do not have any control on the data, which could potentially be misused. The idea of a *Somewhat Trusted Party* is to limit the information that it receives for a particular task. In our case, for distance computation, the Somewhat Trusted Party should obtain just the information necessary to compute the distance. Thus, this party is *trusted* because we believe that the computation will be done correctly and the result will be sent with no modifications, but it is also *somewhat* because we are not willing to send the raw vectors to perform the computation.

We propose to use Gaussian Modular Hash to solve this problem for distance computation. Indeed, we have seen that this transformation provides a good estimate of the actual distance between inputs through the modular distance between hashes. In addition, since this hash satisfies the limited leakage property as well as the indistinguishability property, it hides the information of the inputs preserving just the information needed to estimate the distance between them. We propose the following protocol:

Input: Alice and Bob have $\mathbf{x}_1$ and $\mathbf{x}_2 \in \mathbb{R}^N$ respectively.

Output: Alice and/or Bob obtain an estimation of $\|\mathbf{x}_1 - \mathbf{x}_2\|$, provided it is lower than some threshold T.

Protocol:

1. Alice and Bob agree on a threshold T and precision ϵ.

2. Alice generates (k, A, U) and sends them to Bob securely, this is

   $$a_{ij} \sim \mathcal{N}\left(0, \frac{\pi}{2}\right) \quad \text{and} \quad u_i \sim \text{Unif}(0, k)$$

   The parameters T and ϵ define the value of k and the dimensionality of A and U according to the theorems presented in chapter 5.

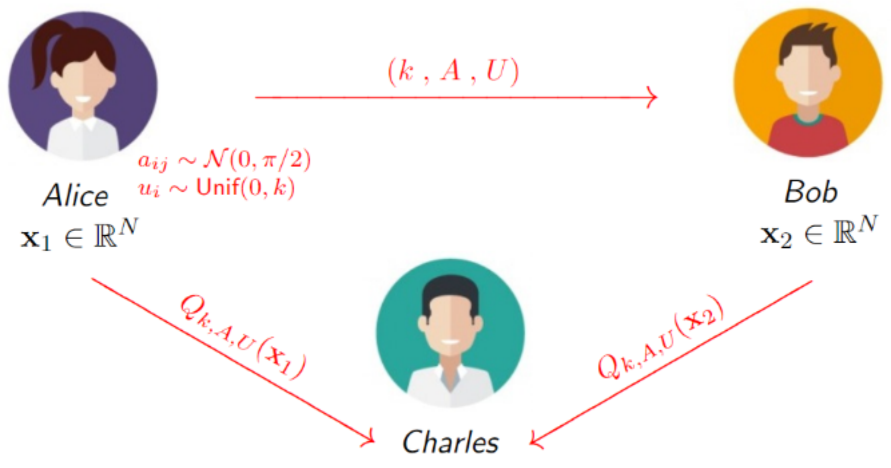

(k, A, U)
$a_{ij} \sim \mathcal{N}(0, \pi/2)$
$u_i \sim \text{Unif}(0, k)$
Alice
$\mathbf{x}_1 \in \mathbb{R}^N$
Bob
$\mathbf{x}_2 \in \mathbb{R}^N$
$Q_{k,A,U}(\mathbf{x}_1)$
$Q_{k,A,U}(\mathbf{x}_2)$
Charles

Figure 9.1: Protocol for distance computation using a Somewhat Third Trusted Party

3. Alice computes $Q_{k,A,U}(\mathbf{x}_1)$ and sends it to Charles.

4. Bob computes $Q_{k,A,U}(\mathbf{x}_2)$ and sends it to Charles.

5. Charles computes $d = d_{mod}(Q_{k,A,U}(\mathbf{x}_1), Q_{k,A,U}(\mathbf{x}_2))$ and sends d to Alice and Bob.

First, we can see that after this protocol no party gets anything more than the estimate of $\|\mathbf{x}_1 - \mathbf{x}_2\|$. Alice and Bob never see each others' vectors. Charles never sees the plain vectors and receives just two vectors in $\mathbb{Z}_k^M$, where each one can be seen as a sequence of independent realizations of draws from a uniform distribution in $\mathbb{Z}_k$. Since Charles does not know (A, U), he does not have any mechanism to reconstruct the actual vectors, or even extract any kind of information more than the distance between them.

Note that it is important that Charles must not know (A, U); otherwise he may reconstruct the original vectors, particularly if he has some knowledge of the domain of the signals. For the same reason Alice and Bob cannot directly share their hashes after sharing the key (k, A, U). Provided these conditions are followed, the scheme can be shown to be information theoretically secure.

One drawback of this protocol is related to the key transmission. Indeed, if the vectors have a high dimension, transmitting the matrix A may cause a communication overhead. To prevent this problem, we can make A public and keeping secret U and introducing a secret random permutation in the hash components.

9.3 Enhancing Privacy with Cryptography

In case we would like to prevent the use of a third party, we can establish a cryptographic protocol which lets us compute the modular distance privately. However, since the calculation of the modular distance involves a minimum between two numbers, it is not combine well with the application of cryptographic techniques, such as homomorphic encryption. Although methods like Secure Multiparty Computation can deal with this type of problem (computing minima), both the communication and computation complexity increase as the length of the hash increases.

Nevertheless, it is possible to reduce the modular distance computation to a Hamming Distance computation, an operation that can be easily performed in the encrypted domain. In fact, we can encode any element in $\mathbb{Z}_k$, with k an even number, as a vector in $\{0,1\}^{\frac{k}{2}}$. If $a \in \mathbb{Z}_k$, we define $\mathbf{c}(a) \in \{0,1\}^{\frac{k}{2}}$ as follows: If $a \leq \frac{k}{2}$,

$$c(a)_i = \begin{cases} 1 & \text{if } i \leq a \\ 0 & \text{otherwise} \end{cases} \tag{9.2}$$

If $a > \frac{k}{2}$,

$$c(a)_i = \begin{cases} 0 & \text{if } i \leq a - \frac{k}{2} \\ 1 & \text{otherwise} \end{cases} \tag{9.3}$$

Example 9.3.1 *In $\mathbb{Z}_6$, the code left*

$$\mathbf{c}(0) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{c}(1) = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{c}(2) = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \quad \mathbf{c}(3) = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad \mathbf{c}(4) = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \quad \mathbf{c}(5) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

This kind of coding has the following property,

$$d_{mod}(a,b) = d_{Hamming}(\mathbf{c}(a)\,,\,\mathbf{c}(b)) = \sum_{i=1}^{k/2} c(a)_i \oplus c(b)_i \tag{9.4}$$

Therefore, an element $\mathbf{z} \in \mathbb{Z}_k^M$ can be coded as $c(\mathbf{z}) \in \{0,1\}^{M \cdot \frac{k}{2}}$, where

$$\mathbf{c}(\mathbf{z}) = \begin{bmatrix} \mathbf{c}(z_1) \\ \mathbf{c}(z_2) \\ \vdots \\ \mathbf{c}(z_M) \end{bmatrix} \tag{9.5}$$

Then, the Normalized Modular Distance between $\mathbf{z}_1$ and $\mathbf{z}_2 \in \mathbb{Z}_k^M$ is equal to

$$d_{mod}(\mathbf{z}_1\,\mathbf{z}_2) \quad = \quad \frac{d_{Hamming}(\mathbf{c}(\mathbf{z}_1),\ \mathbf{c}(\mathbf{z}_2))}{M} \tag{9.6}$$

With this result we enable the estimation of the Euclidean distance between two points in $\mathbb{R}^N$ using the Hamming distance of two binary vectors. As a consequence, we can replace the third party computation by any protocol which computes securely the Hamming distance between binary vectors; for example, [31] defines a two party protocol based on Oblivious Transfer.

To summarize, we define the following protocol,

Protocol:

1. Alice generates (k, A, U) and sends them to Bob.

2. Alice computes $\mathbf{c}(Q_{k,A,U}(\mathbf{x}_1))$ privately.

3. Bob computes $\mathbf{c}(Q_{k,A,U}(\mathbf{x}_2))$ privately.

4. Alice and Bob apply a secure two party protocol to compute the Hamming distance d between $\mathbf{c}(Q_{k,A,U}(\mathbf{x}_1))$ and $\mathbf{c}(Q_{k,A,U}(\mathbf{x}_2))$.

5. Alice and Bob compute the estimate of $\|\mathbf{x}_1 - \mathbf{x}_1\|$ as $\frac{d}{M}$.

Unlike most protocols based on homomorphic encryption to compute the Euclidean distance, the complexity of the presented protocols does not depend on the dimensionality of the vector at the moment of applying the cryptographic technique. Hence, our proposal for computing the Euclidean distance between two vectors is to embed them in binary vectors and compute the Hamming distance between them.

9.4 Conclusions

In this chapter we have presented an application of Gaussian Modular Hashes. We know that, given a threshold T, we can generate hashes from vectors preserving the Euclidean distance if this is smaller than T. These hashes are uninformative if the random parameters of the function are unknown. With this kind of transformation, we enable the description of protocols to compute distances privately and efficiently. In fact, the hash defined is not only uninformative, but it can be seen as a transformation which reduces the data dimension and preserves the Euclidean distance in the modular distance output space. Although the fact that the preservation of the distance holds until some threshold seems to be a drawback, if both Alice and Bob know the maximum possible distance between their vectors (i.e. the threshold T within which they would like to enable distance computation), then they can set an appropriate value of k which lets them preserve all possible values of the distance within that threshold.

In addition, we describe how to prevent the use of a third party, by using a particular encoding of the hashes that transforms the modular distance between hashes into a Hamming distance between binary vectors, and a cryptographically secure two-party computation protocol for Hamming distance computation.

Part V

Conclusions

Chapter 10

Thesis Conclusions

In this thesis, we have analyzed the problem of performing operations on the cloud with private data. We considered a setting where a data owner can transform its data instances prior to sending them to the cloud, in order to prevent an adversary from gaining potentially sensitive information about the data. We proposed the concept of Limited Leakage Transformations, which are functions with two properties: a transformed points provides no information about the original data, and the comparison of transformed points cannot provide information about the points if the distance between them is large, but it is persistent for points that are close.

We considered the utility of this transformation for training distance-based machine learning models in the cloud with sensitive data. Our privacy mechanism allows us to hide information in the cloud, exhibiting just the relevant information related to points that are sufficiently close. We also analyzed concrete potential applications that this kind of mechanism may have. We summarize our results below.

10.1 Summary of Results

Part II

In this part, we studied measures of privacy loss, concluding that the mutual information between transformed points is a reasonable and natural way to measure information leakage between transformed points.

Later, in chapter 4, we introduced the concept of *Limited Leakage Transformation* as a privacy mechanism. This allows to transform data points in a manner that removes relationships between vectors that are distant from one another. In order to understand the different types of comparisons that an adversary can make, we introduced the idea of information leakage under complex comparisons, which basically is the mutual information between two sets of transformed data.

We proved a generic theoretical result about how the information leakage can be controlled under specific geometric conditions of the data points. We also have presented particular examples of this kind of transformation, providing a rigorous analysis about the information leakage at different levels of comparison. Moreover, we note that the proposed particular constructions are relatively fast to compute; they are simply quantized linear operations.

Finally, we make a comparison between our technique and other approaches that aim to preserve privacy. We conclude that our technique is agnostic to the potential calculation to be performed on the cloud, unlike other methods such as differential privacy, where the main objective is preserving privacy after releasing a set of statistics or a machine learning model.

Part III

In this part we studied different kinds of operations that can be useful when the points are sufficiently close. We started by analyzing the Normalized Hamming distance between transformed points, in particular using our proposed Gaussian Modular Hash function. We showed that metric between transformed vectors provides a good estimation of the actual Euclidean distance between original points. The computational advantage is evident; Hamming distance computation is a fast operation in general, with negligible computational cost.

We also analyzed another topology in $\mathbb{Z}_k$, provided by the modular distance. We could show that the normalized modular distance can estimate accurately the Euclidean distance if this is smaller than some threshold. For both methods to compute distance between hashed data, we gave theoretical guarantees over the necessary number of components to bound the error estimation.

Next, in chapter 6, we focus on machine learning algorithms based on distances. One contribution was to make a direct connection with kernel methods, showing that a small modification of the hash function can be connected to kernel estimation, allowing us to covert a non-linear kernel-computation problem using the original vectors into a linear problem with hashed data. This is particularly useful in large scale scenarios, where computing the Gram-kernel matrix is an infeasible task.

We also presented an empirical evaluation of this technique, for the particular case of the Laplacian kernel, showing a minimal loss of performance in the problem of audio scene classification (as an exemplar task) using support vector machines.

Part IV

In this last part, we analyze the scope of our proposed concept in different applications. In all of them the use of distance computation is a key element. First, we went through the problem of protecting speech signals, which is a relevant topic since an audio recording may contain a lot of sensitive information about the speaker. One popular application of speech processing is speaker identification, where speech is used as a signature for individuals. Beside the fact the speech signal may be used for undesirable purposes, other problems arise in this application. As in any biometric, if the input signal is compromised, security concerns appear; for example, an imposter may try to impersonate a particular person for accessing a system. To deal with these problems, the concept of cancelable biometrics has been proposed in the past. In this work, we could show that Limited Leakage transformations can be used to define a Two Factor Transformation, satisfying the required conditions to obtain a cancelable function, and therefore, allowing to protect the biometric signals from potential privacy risks.

We also explored the problem of retrieving images with privacy constraints. We discusse how our method, besides enhancing privacy, can be used to perform fast search of files. A system for private image retrieval has been implemented using our approach.

Finally, we study a secure two-party computation protocol based on our technique to compute distances without revealing the corresponding vectors. To do so, we first introduced the notion of "somewhat-trusted" third party, who receives the information to estimate the distance but can not infer any other information about the corresponding vectors involved. Since our hash function also works as a dimensionality reduction method, this method can be used to speed up the calculation. Moreover, if the use of a third part is not desired, we can combine this technique with cryptographic approaches. Since the distance estimation can be done using a Hamming distance between transformed points, we can use any cryptographic technique that is optimized to perform this operation.

10.2 Discussion

We have presented the concept of Limited Leakage Transformation, which has the property that comparing transformed vectors reveals information about their similarity in a limited way, giving information about their similarity if they are close, but providing no information about it if the points are far enough. Based on Information Theory tools, we study generic theoretical results and we propose concrete constructions of transformations that satisfy our requirements. In general, these transformations are fairly efficient to compute, which is a relevant property since in the context of cloud computing, the evaluation of these functions must be done before sending the data to the cloud.

We also analyzed the different operations that can be performed on the transformed data. We studied how to estimate distances and how to train distance-based machine learning models. We observe that, in general, these operations are fast to compute and also reduce the complexity of some machine learning algorithms. Finally, both theoretically and empirically, we could observe that our method also provides a new way to set the trade-off between privacy and utility. In fact, controlling the information leakage has an effect on the performance of the quality of distance estimation and the accuracy of classification models. For our concrete examples, we observe that the way to control the information leakage is by varying the number of components in the transformed vector or through the variance of the random keys used by the transform. From a user perspective, this is very convenient; she or he can determine, based on their own risk tolerance, the information that they are willing to leak.

In this way, we believe that our proposed approach allows us to perform fast and private computations on the cloud.

Chapter 11

Future Work

In this thesis, we have developed the concept of Limited Leakage Transformation, presenting both theoretical and empirical results, as well as, analyzing different applications. However, there is a lot of room to continue developing this theory.

11.1 Multiparty Formulation

An open question relates to obtaining a multiparty scheme based on our method. It is very often to assume that the data comes from different sources, for example, different users of a platform. If all the users need to send the data to a common repository to perform computation there, it is reasonable to think that they need to have their own secret key. However, if users generate their keys independently, we know that they produce independently transformed values, and therefore, the comparison between them will be useless.

Thus, something interesting to explore is how to combine different, independent keys in order to obtain similar properties as that presented in this work.

Another direction is related to controlling the leakage based on different users' role. For example, for the interactions of some particular users we would like a small level of leakage, while for other interactions we are willing to tolerate larger information leakage.

11.2 Transforming Non-vector objects

The concept of Limited Leakage Transformation and $\varepsilon-$persistence were defined for any metric space. This provides a generic framework to work with. Nevertheless, the practical constructions we presented are just for the Euclidean space with some norm. Therefore, we think that it could be very relevant to explore transformations for other kind of metric spaces, such as graphs or variable length strings.

The challenge is to find computationally efficient transformations which allow us to control the information leakage in an easy manner.

11.3 Signal Reconstruction using Keys

One problem, which comes from the Compressed Sensing field, is whether we can recover the original vector $\mathbf{x}$ from $Q_R(\mathbf{x})$ and R. All the privacy statements stated in this work make the assumption that R must be kept secret. However, it is not evident how to reconstruct the original signal knowing the secret key. If this can be done with an efficient algorithm, then we can also consider our transformation as a compressing technique. If not, it means that the keys can be released and the main risk related to the privacy of the data are dictionary attacks. Nevertheless, the evidence seems to point to the former scenario. In [110] some algorithm have been presented to reconstruct signal under specific geometric conditions with a similar (but not the same) scheme to compress data.

Appendix A

Random Convolutional Features

A.1 Introduction

One particularly important task in time signal processing is signal classification, where given a time signal, we must determine the class it belongs to. Applications range from classification of medical time signals (such as ECGs) to automatic speech recognition, classification of financial series, astronomy, and even modeling language.

The most common approaches employ statistical models, such as Hidden Markov Models [78] [79] [80], conditional random fields [81] [82], or convolutional or recurrent neural networks [83] [84] [85] [86]. While highly effective, these approaches are typically data intensive, prone to overfitting, and often need high tuning of hyperparameters.

An alternate approach is through *kernel* methods, such as support vector machines, which are generally less data intensive and easier to optimize. However, these come with a concomitant challenge: that of defining an appropriate kernel that captures the similarity between instances. Conventional similarity measures such as the inner product will not suffice, since one must also consider the *alignment* between time signals when computing their similarity. Consider, for instance, the two signals $sin(\omega k)$ and $sin(\omega k + \phi)$, where k is the running time index. Although the two are structurally identical, the inner product between the two will vary with the phase shift ϕ. In order to correctly compute the similarity between the two signals, they must be *aligned* to line up; however the amount of shift will generally not be known *a priori*. Hence, a good similiarity measure between time signals must be *shift invariant*.

Several common solutions for computing kernels between time signals view them as regular vectors and do not consider the alignment between them [87]. Other approaches do implicitly consider the issue, but generally construct expensive statistical machinery to do so [88][89] [90] [91]. Fourier-transform-based methods transform the signals into the Fourier domain, to deal with the issue of shift invariance [92], however they effectively consider every alignment, not just the right one, making them susceptible to noise [93]. The kernels that best account for the alignment between the two signals are DTW-based kernels [94] [95] that explicitly find the optimal warping between them in the process of computing their similarity. However, the computational complexity of finding the optimal warp is quadratic, or even cubic in the length of the sequences, making them impractical in many scenarios.

Possibly the most appropriate similarity measure in this setting is the *peak cross-correlation* between the signals. The cross-correlation of two time signals computes the inner product between the signals at *every* alignment between the two. Formally, given two time signals $\mathbf{f} = (f_1, ..., f_N)$ and $\mathbf{g} = (g_1, ..., g_N)$ of length N, the cross-correlation between the two, represented as $\mathbf{f} \star \mathbf{g}$, is the signal computed as

$$(\mathbf{f} \star \mathbf{g})_i \quad = \quad \sum_k f_k \cdot g_{k+i} \tag{A.1}$$

Different settings of the limits of the summation in k result in different types of cross-correlation. The *linear* cross-correlation between $\mathbf{f}$ and $\mathbf{g}$ is obtained by summing k from $-\infty$ to ∞, assuming that both $\mathbf{f}$ and $\mathbf{g}$ take value 0 outside the index range $1 \cdots N$. The *circular* cross-correlation is obtained by summing k from 1 to N, and assuming $g_{k+i} = g_{N-k-i}$ for $k + i > N$ [96]. The linear cross-correlation may also be obtained from the circular cross-correlation by zero-padding both signals out to length $2N$ and computing the circular cross-correlation over the longer zero-padded sequences. In either case, the outcome of Equation A.1 is a complete time signal, where each term $(\mathbf{f} \star \mathbf{g})_i$ computes the inner product between $\mathbf{f}$ and $\mathbf{g}$ when the latter has been shifted by i in order to line it up with $\mathbf{f}$.

The *peak* cross correlation is the maximal value of the cross correlation, $\max_i (\mathbf{f} \star \mathbf{g})_i$, which is the inner product under the best scoring alignment between the two time signals. Although this quantity has good properties such as symmetry, and accounts well for the alignment of the two signals, it does not define a valid *kernel*; the corresponding gram matrix may not be positive definite [97]. Instead, we define a *cross-correlation kernel* as a weighted combination of the elements of the cross-correlation, such that the maximum value of the cross-correlation has a larger weight compared to the remaining values:

$$\mathcal{K}(\mathbf{f}\ , \mathbf{g}) = \frac{1}{N} \sum_i \exp\left(\gamma \cdot (\mathbf{f} \star \mathbf{g})_i\right), \tag{A.2}$$

where γ is a positive parameter which governs the relative contribution of the individual terms – as γ increases, the contribution of the maximum values of the cross-correlation is larger. It can be proved that $\mathcal{K}(\mathbf{f}\ , \mathbf{g})$ as defined above is indeed a proper kernel [97].

For the *circular* cross-correlation, one way to compute this is through the Fourier Transform. Using the properties of the Fourier Transform it is straightforward to see that [96]

$$\mathcal{K}(\mathbf{f}\,,\,\mathbf{g}) = \frac{1}{N}\sum_i \exp\left(\gamma \cdot \mathcal{F}^{-1}\left(\mathcal{F}(\mathbf{f}) \cdot \mathcal{F}(\mathbf{g})^*\right)_i\right) \tag{A.3}$$

where $\mathcal{F}(\mathbf{f})$ is the Fourier transform of $\mathbf{f}$, $\mathcal{F}^{-1}$ is the inverse Fourier transform operator, and $\mathcal{F}(\mathbf{g})^*$ is the component-wise complex conjugate of $\mathcal{F}(\mathbf{g})$. Using the Fast Fourier Transform algorithm to compute the Fourier transforms, the complexity of computing $\mathcal{K}(\mathbf{f}\,,\,\mathbf{g})$ is $\mathcal{O}(N \log N)$ [98]. While this is less computationally expensive to compute than other time-signals kernels, it is still significantly more expensive than computing a simple inner product, which has a complexity of only $\mathcal{O}(N)$. The difference is even greater when the sequences must be increased in length by zero-padding, in order to compute the linear cross-correlation via circular cross-correlation.

This leads us to the key challenge addressed in this work. The cost of computing the kernel is a significant component of both training and evaluating kernel machines such as SVMs. For SVMs in particular, for non-linear kernels such as the cross-correlation kernel, we must solve the dual form of the optimization to train the model, which requires the computation of a gram matrix. Gram matrix computation scales quadratically (both in time and storage complexity) with the number of training samples. Thus, for the cross-correlation kernel, training the SVM with L training samples would require $\mathcal{O}(L^2 N \log N)$ computations, just for the gram matrix. In addition, given a model with V support vectors, the inference would require $\mathcal{O}(VN \log N)$ computations per sample to be classified.

In such situations, it is well known that *random projections* of the data can be used to derive simpler, lower-dimensional representations of the data on which the kernels can be approximated, to derive savings in both computation and storage [29] [99]. This method has been successfully applied to both inner-product kernels of high-dimensional data [100] [21], and to a variety of non-linear kernels [101] [102], although in the latter case the actual manner of computing the random features depends on the kernel.

In this work we extend this principle to derive a randomized scheme for computing lower-dimensional representations of time-signals data, from which cross-correlation kernels can be computed through a simple inner-product computation. We call these *random convolutional features.*

The proposed method has multiple desirable features. The computation of random features makes no assumption about the length of the time signals; this is, the number of components in the low-dimensional representation depends only on the *energy* in the signal, and not on the number of samples in it. In practice, the number of components required is generally smaller than the length of the time signal itself. Thus the cost of computing a kernel thus becomes that of computing a low-dimensional inner product. For energy-normalized signals, this becomes a *fixed*-dimensional computation. The final benefit is one that is common to most random-feature-based approximations of kernel functions: since kernel computation now becomes an inner product, the training of the SVM can be solved in its primal form, eliminating the need for the gram matrix. The time and space complexity of the model too simply become the size (number of components) in the random feature itself; the cost of inference reduces to that of deriving the projection itself.

This part is organized as follows. In Section 2 we describe our proposed random feature construction providing some theoretical guarantees to validate its use. In Section 3, we present a experimental results on a number of data sets, showing that these benefits come at no cost to classification performance, which remains comparable to that obtained with the full cross-correlation kernel. Finally, in Section 4, we conclude analyzing the scope of this techniques and discussing future directions.

A.2 Methods

In this section we describe the construction of our random feature scheme that approximates the cross-correlation kernel.

First, we present our scheme, named *random convolutional features*, which defines a random mapping from time signals to a low dimensional space, approximating the cross-correlation kernel through the computation of inner products between random features. Later, we show theoretical guarantees to validate the approximation to the cross-correlation kernel.

A.2.1 Random Convolutional Features

As its name says, the random feature scheme presented is based on a convolutional operation. The randomness of this features comes from its parameters which are randomly generated according to some particular distributions.

Definition. Let γ be a positive number. Let $\mathcal{W} = \{\mathbf{w}_1, ..., \mathbf{w}_M\}$ be a set of M random time signals of length N, where each component is random and independently generated according to a Gaussian distribution $\mathcal{N}(0, \gamma)$, and $\mathcal{U}$ a set M scalars where each one is random and independently generated using a uniform distribution between 0 and 2π. We define a *random convolutional feature* as the random mapping $\psi_{\mathcal{W},\mathcal{U},\gamma} : \mathbb{R}^N \to \mathbb{R}^M$ as follows:

$$\psi_{\mathcal{W},\mathcal{U},\gamma}(\mathbf{f})_i \quad = \quad \frac{\sqrt{2}}{N\sqrt{M}} \exp\left(\frac{\gamma}{2}\|\mathbf{f}\|^2\right) \sum_{j=1}^{N} \cos\left((\mathbf{w}_i * \mathbf{f})_j + u_i\right) \tag{A.4}$$

where $*$ denotes the convolution operation between signals.

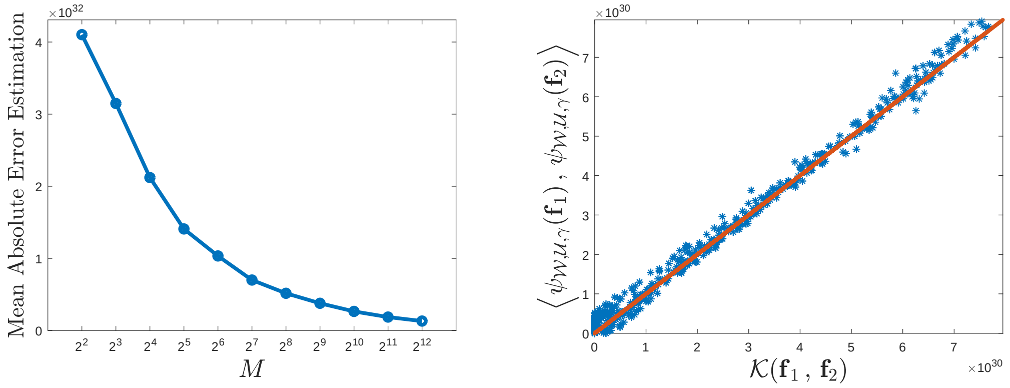


Figure A.1: Approximation of Cross-correlation Kernel through inner product of Random Features. (Left) Effect of the number of components in the random mapping on the approximation error for Kernel estimation. (Right) Pairwise comparison between the cross-correlation and its estimation through random features. Each dot is a corresponds to a pair $(\mathbf{f}_1, \mathbf{f}_2)$. We consider $M = 1024$. Both figures where generated using the Gun-Point data set from [103]

A.2.2 Theoretical Results

The following results show the relation of the random convolutional features to the cross-correlation kernel. The first result analyzes the expected values of inner products between random features while the second shows how to use these inner products.

Theorem 1. For any time signals $\mathbf{f}_1$, $\mathbf{f}_2 \in \mathbb{R}^N$, the expected value of the inner product between their corresponding random mappings $\psi_{\mathcal{W},\mathcal{U},\gamma}(\mathbf{f}_1)$ and $\psi_{\mathcal{W},\mathcal{U},\gamma}(\mathbf{f}_2)$ is given by:

$$\mathbb{E}\Big(\langle \psi_{\mathcal{W},\mathcal{U},\gamma}(\mathbf{f}_1) \ , \ \psi_{\mathcal{W},\mathcal{U},\gamma}(\mathbf{f}_2)\rangle\Big) \quad = \quad \mathcal{K}\big(\mathbf{f}_1 \ , \ \mathbf{f}_2\big) \quad = \quad \frac{1}{N}\sum_{i=1}^{N} \exp\left(\gamma \cdot (\mathbf{f}_1 \star \mathbf{f}_2)_i\right) \tag{A.5}$$

Proof

Since $\mathbf{w}$ are totally random, we can interchange the convolutional operation by a cross-correlation.

$$\begin{aligned}
\mathbb{E}\left(\psi_{\mathcal{W},\mathcal{U},\gamma}(\mathbf{f}_1)_i \cdot \psi_{\mathcal{W},\mathcal{U},\gamma}(\mathbf{f}_2)_i\right) \quad &= \quad \frac{2}{N^2 M}\exp\left(\frac{\gamma}{2}\|\mathbf{f}_1\|^2\right)\cdot\exp\left(\frac{\gamma}{2}\|\mathbf{f}_2\|^2\right)\cdot \\
&\quad\; \mathbb{E}\left[\left(\sum_{j=1}^{N}\cos\left((\mathbf{w}_i\star\mathbf{f}_1)_j + u_i\right)\right)\cdot\left(\sum_{j=1}^{N}\cos\left((\mathbf{w}_i\star\mathbf{f}_2)_j + u\right)\right)\right] \\
&= \quad \frac{2}{N^2 M}\exp\left(\frac{\gamma}{2}\|\mathbf{f}_1\|^2\right)\cdot\exp\left(\frac{\gamma}{2}\|\mathbf{f}_2\|^2\right)\cdot \\
&\quad\; \mathbb{E}\left[\sum_{j=1,k=1}^{N}\cos\left((\mathbf{w}_i\star\mathbf{f}_1)_j + u_i\right)\cdot\cos\left((\mathbf{w}_i\star\mathbf{f}_2)_k + u_i\right)\right]
\end{aligned}$$

but, by definition of cross-validation, we know that $(\mathbf{w} \star \mathbf{f})_j = \langle \mathbf{w} \ , \ \mathbf{f}_{(j)}\rangle$, where $\mathbf{f}_{(j)}$ is the circular shifted version of $\mathbf{f}$ shifted by j positions. Then, from [21], we have

$$\begin{aligned}\mathbb{E}\left[\psi_{\mathcal{W},\mathcal{U},\gamma}(\mathbf{f}_1)_i \cdot \psi_{\mathcal{W},\mathcal{U},\gamma}(\mathbf{f}_2)_i\right] &= \frac{2}{N^2M}\exp\left(\frac{\gamma}{2}\|\mathbf{f}_1\|^2\right)\cdot\exp\left(\frac{\gamma}{2}\|\mathbf{f}_2\|^2\right)\cdot \\ &\quad \sum_{j=1,k=1}^{N}\mathbb{E}\Big[\cos\left((\mathbf{w}_i\star\mathbf{f}_1)_j+u_i\right)\cdot\cos\left((\mathbf{w}_i\star\mathbf{f}_2)_k+u_i\right)\Big] \\ &= \frac{1}{N^2M}\sum_{j=1,k=1}^{N}\exp(\gamma\langle\mathbf{f}_{1(j)}\,,\,\mathbf{f}_{2(k)}\rangle)\end{aligned}$$

but, we know that $\langle\mathbf{f}_{1(j)}\,,\,\mathbf{f}_{2(k)}\rangle = \langle\mathbf{f}_1\,,\,\mathbf{f}_{2(k-j)}\rangle$. Therefore,

$$\begin{aligned}\mathbb{E}\left[\psi_{\mathcal{W},\mathcal{U},\gamma}(\mathbf{f}_1)_i \cdot \psi_{\mathcal{W},\mathcal{U},\gamma}(\mathbf{f}_2)_i\right] &= \frac{1}{N^2M}\cdot N\sum_{j=1}^{N}\exp(\gamma\langle\mathbf{f}_1\,,\,\mathbf{f}_{2(j)}\rangle) \\ &= \frac{1}{NM}\sum_{j=1}^{N}\exp(\gamma(\mathbf{f}_1\star\mathbf{f}_2)_j)\end{aligned}$$

Finally, using the linearity of expectation, we get the desired result.

□

Thus, the cross-correlation kernel corresponds to the expected value of the inner product between the proposed random features. Furthermore, we can analyze the convergence of actual inner product to the kernel function. The following statement, based on applying Hoeffding's inequality, provides some guarantees.

Theorem 2. If we have a set of L time signals of length N, $\mathcal{T} = \{\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_L\}$, and $C \geq \|\mathbf{f}_i\|^2$ for all i, then, considering $M \geq \frac{16\exp(2\gamma C)}{\varepsilon^2}\log\left(\frac{L}{\eta}\right)$, we have

$$\mathbb{P}\Big(\forall\mathbf{f}_i,\mathbf{f}_j\in\mathcal{T}\,,\,\Big|\big\langle\psi_{\mathcal{W},\mathcal{U},\gamma}(\mathbf{f}_i)\;,\;\psi_{\mathcal{W},\mathcal{U},\gamma}(\mathbf{f}_j)\big\rangle - \mathcal{K}\big(\mathbf{f}_i,\mathbf{f}_j\big)\Big| < \varepsilon\Big) \quad\geq\quad 1-\eta \tag{A.6}$$

Therefore, the probability of obtaining a small error in the estimation of the kernel using the inner product between random features depends on the number of components in the random features and the energy of the time signals. Figure A.1 illustrates this.

Applying this technique to SVMs has several advantages. In case of using directly the cross-correlation kernel, we know that for training we need to obtain the gram matrix which requires $\mathcal{O}(L^2\,N\log N)$ computations. Alternatively, we can compute random features and train a linear model using a fast implementation. Obtaining the random features requires $\mathcal{O}(MLN\log N)$. So, if $M \ll L$, we can observe a natural advantage in the training phase. In the case of inference, using the kernel function directly requires to store $\mathcal{O}(VN)$ values and perform $\mathcal{O}(VN\log N)$ operations for each prediction, where V is the number of support vectors. On the other hand, using random features, we must store $\mathcal{O}(M)$ values and perform $\mathcal{O}(MN\log N)$ computations for each prediction. Then, in case of $M \ll V$ we get a benefit for inference computation, while having $M \ll VN$ gives benefits in model storage.

A.3 Experiments

As a proof of concept, we applied the presented technique over 15 different data sets for the task of time signals binary classification. This section aims to show the benefits of the modeling capabilities of the cross-correlation kernel as well as analyze the benefits of using random convolutional features with linear SVM over the raw time signals with the cross-correlation kernel.

A.3.1 Datasets

For our experiments, we used 15 data sets provided by the University of California, Riverside time series classification archive [103]. Each data set consists of equal-length time series belonging to one of two classes. The data sets selected and details about the size of the training and testing sets, as well as the time series length are presented in table A.1.

Beside the sample points, this archive also provides the prediction error of three different classification methods; 1-Nearest Neighbor using Euclidean distance, 1-Nearest Neighbor using Best Warping Window DTW, and 1-Nearest Neighbor using DTW with no Warping window.

A.3.2 Results using Nonlinear SVM with Cross-correlation kernel

First, we evaluate the capabilities of the cross-correlation kernel over the 15 data sets presented in table A.1 using a SVM classifier based on this kernel. To select the hyperparameters γ that defines the kernel, and C corresponding to the SVM cost, we performed a grid search considering $\gamma \in \{2^{-10}, 2^{-9}, ..., 2^{-1}, 1\}$ and $C \in \{10^{-4}$, 10^{-2},1 ,10^{2} , $10^{4}\}$, doing the selection through jack-knife cross-validation. In table A.1 the selected parameters can be found, as well as their corresponding error classification rate.

We can observe that in 11 of 15 data sets, SVM models with cross-correlation kernel outperform alternative methods. This shows empirically the utility of this kernel function.

A.3.3 Results using Linear SVM with Random Features

To analyze the utility of our proposed random features we trained linear SVMs using different values of M. To select the hyperparameters we proceeded as before. In table A.1 we present the error rates we obtained. We can observe how the error rate changes as the number of components on the random features increases. In general, we observe that error rate decreases as M increases; except for the case of the data set *BirdChicken*.

Moreover, in 7 of 15 data sets the difference between the nonlinear method compared to the linear method is less than 1% using 256 or less components in random features, and in other 4 data sets the difference is

Table A.1: Results in Binary Classification Task.

Data set		SVM with Cross-Correlation Kernel			SVM with Random Convolutional Features				Other Methods		
	Signal Length	γ	C	Error Rate	$M = 2^5$	$M = 2^6$	$M = 2^7$	$M = 2^8$	1-NN Euclidean Distance	1-NN Best Warping Window DTW	1-NN DTW, no Warping Window
Gun-Point	150	0.0625	0.0001	**0.013**	0.035	0.031	0.028	0.025	0.087	0.087	0.093
Lightning-2	637	0.0156	0.0001	0.230	0.282	0.252	**0.224**	0.212	0.246	0.131	0.131
ECG	96	0.0313	1	**0.100**	0.159	0.142	0.130	0.129	0.120	0.120	0.230
Coffee	286	0.0313	0.0001	0.000	0.012	**0.007**	**0.003**	**0.000**	0.000	0.000	0.000
ECGFiveDays	136	0.1250	0.0001	**0.005**	0.009	0.003	0.006	**0.002**	0.203	0.203	0.232
MoteStrain	84	0.2500	0.0001	0.265	0.274	0.246	0.238	**0.255**	0.121	0.134	0.165
SonyAIBORobot Surface	70	0.1250	0.0001	**0.068**	0.237	0.212	0.183	0.182	0.305	0.305	0.275
SonyAIBORobot SurfaceII	65	0.0625	0.0100	**0.132**	0.199	0.194	0.187	0.168	0.141	0.141	0.169
TwoLeadECG	82	0.2500	0.0001	**0.011**	0.036	0.024	0.021	**0.016**	0.253	0.132	0.096
BeetleFly	512	0.0156	0.0001	**0.050**	0.303	0.292	0.270	0.242	0.250	0.300	0.300
BirdChicken	512	0.0078	0.0100	**0.200**	0.171	0.176	0.185	**0.204**	0.450	0.300	0.250
Ham	431	0.0313	0.0001	**0.305**	**0.309**	0.311	**0.309**	**0.300**	0.400	0.400	0.533
Herring	512	0.0010	100,000	**0.359**	0.418	0.424	0.421	0.407	0.484	0.469	0.469
ToeSegmentation1	277	0.0313	0.0001	**0.118**	0.274	0.248	0.225	0.221	0.320	0.250	0.228
ToeSegmentation2	343	0.0039	1.0000	0.123	0.182	0.157	0.140	0.136	0.192	0.092	0.162

less than 5%. This diversity is due the complexity of finding the decision boundary, as is established in [100]. We note that, even with 32 components the loss of performance be can considered acceptable in several cases.

A.4 Conclusions

In this work we presented a random feature scheme that allows us to transform time signals into a low dimensional vectors, and used them to approximate the cross-correlation kernel through a simple inner product computation. Our main contribution was to show theoretical guarantees to validate this approximation. We studied this scheme with SVMs for Time Signal Classification. The proposed random feature provides minimal loss of performance in several cases, reducing storage and transforming a non-linear learning problem into a linear one. This has a significant implication in big data scenarios, where a large number of signals must be processed.

Moreover, the presented scheme has other potential applications. For example, with this technique we could train compact models on devices; linear models are much simpler to train than kernel-based models, specially with low dimensional data. Another application is related to cloud computing and privacy. In case we need to process sensitive time signals (e.g. medical or financial data), we can apply this random mapping keeping the random parameters $\mathcal{W}$ and $\mathcal{U}$ private. Then, we can still process the transformed data in the cloud, hiding in somehow information without exposing the original data.

Finally, we think this work can provide some guidelines to understand other methods based on cross-correlation, such as Convolutional Neural Networks. In fact, previous works have shown that it is possible to obtain good performance even without training the random initialized filters [104], being consistent with our results.

Appendix B

Proofs

B.1 Proof Theorem 4.2.1

Theorem 4.2.1.
Consider that $\{\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_n\} \subseteq B(\mathbf{x}_0, \varepsilon_1)$ and $\{\mathbf{z}_0, \mathbf{z}_1, ..., \mathbf{z}_m\} \subseteq B(\mathbf{z}_0, \varepsilon_2)$. If Q_R is a random transformation that satisfies LLP with constants C and γ, and Q_R is $\varepsilon-$persistent with $\varepsilon > \max\{\varepsilon_1, \varepsilon_2\}$, then

$$\begin{aligned} I\Big(Q_R(\mathbf{x}_0)\,,\,Q_R(\mathbf{x}_1)\,,...,\,Q_R(\mathbf{x}_n)\ \ ;\ \ Q_R(\mathbf{z}_0)\,,\,Q_R(\mathbf{z}_1)\,,...,\,Q_R(\mathbf{z}_m)\Big) \quad \leq \quad & C\ \exp\Big(-\gamma\, d(\mathbf{x}_0\,,\,\mathbf{z}_0)\Big) \\ & + H(p_1) + H(p_2) \\ & + (1-p_1)\log(|\mathcal{E}|^n - 1) \\ & + (1-p_2)\log(|\mathcal{E}|^m - 1) \end{aligned}$$

where $p_i = \rho(\varepsilon_i, Q_R)$, and $H(p)$ is the entropy of a random variable distributed as a *Bernoulli*(p), this is $H(p) = -p\log p - (1-p)\log(1-p)$.

Proof

First, we will prove the following result,

Proposition. For any set of points $\{\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_n\}$ and $\{\mathbf{z}_0, \mathbf{z}_1, ..., \mathbf{z}_m\}$ is always true the next inequality,

$$\begin{aligned} I\Big(Q_R(\mathbf{x}_0)\,,\,Q_R(\mathbf{x}_1)\,,...,\,Q_R(\mathbf{x}_n)\ \ ;\ \ Q_R(\mathbf{z}_0)\,,\,Q_R(\mathbf{z}_1)\,,...,\,Q_R(\mathbf{z}_m)\Big) \quad \leq \quad & I\big(Q_R(\mathbf{x}_0)\,;\,Q_R(\mathbf{z}_0)\big) \\ & + H\Big(Q_R(\mathbf{x}_1), ..., Q_R(\mathbf{x}_n)\,\Big|\,Q_R(\mathbf{x}_0)\Big) \\ & + H\Big(Q_R(\mathbf{z}_1), ..., Q_R(\mathbf{z}_m)\,\Big|\,Q_R(\mathbf{z}_0)\Big) \end{aligned}$$

Proof Proposition

It is easy to observe that

$$\mathbb{P}\Big(Q_R(\mathbf{x}_0) = i_0, ..., Q_R(\mathbf{x}_n) = i_n, Q_R(\mathbf{z}_0) = j_0, ..., Q_R(\mathbf{z}_m) = j_m\Big) \quad \leq \quad \mathbb{P}\Big(Q_R(\mathbf{x}_0 = i_0 \, , Q_R(\mathbf{z}_0) = j_0\Big)$$

Then, we have

$$\begin{aligned}\log \frac{\mathbb{P}\Big(Q_R(\mathbf{x}_0) = i_0, ..., Q_R(\mathbf{x}_n) = i_n, Q_R(\mathbf{z}_0) = j_0, ..., Q_R(\mathbf{z}_m) = j_m\Big)}{\mathbb{P}\Big(Q_R(\mathbf{x}_0) = i_0, ..., Q_R(\mathbf{x}_n) = i_n\Big) \cdot \mathbb{P}\Big(Q_R(\mathbf{z}_0) = j_0, ..., Q_R(\mathbf{z}_m) = j_m\Big)} \quad &\leq \quad \log \frac{\mathbb{P}\Big(Q_R(\mathbf{x}_0) = i_0 \, , \, Q_R(\mathbf{z}_0) = j_0\Big)}{\mathbb{P}\Big(Q_R(\mathbf{x}_0) = i_0\Big)\mathbb{P}\Big(Q_R(\mathbf{z}_0) = j_0\Big)} \\ &+ \log \frac{\mathbb{P}\Big(Q_R(\mathbf{x}_0) = i_0\Big)}{\mathbb{P}\Big(Q_R(\mathbf{x}_0) = i_0, ..., Q_R(\mathbf{x}_n) = i_n\Big)} \\ &+ \log \frac{\mathbb{P}\Big(Q_R(\mathbf{z}_0) = j_0\Big)}{\mathbb{P}\Big(Q_R(\mathbf{z}_0) = j_0, ..., Q_R(\mathbf{z}_n) = j_n\Big)}\end{aligned}$$

Then, multiplying by the joint probability of all the transformed points and doing the proper marginalization, we have the result. □

Proof of Theorem

Under the hypothesis the theorem, and using the previous proposition, we just need to find an upper bound for the conditional entropy terms. Without loss of generality, we can just focus on the set $\{\mathbf{x}_0, ..., \mathbf{x}_n\}$.

To do so, we use a similar approach used to prove Fano's inequality. Let's consider the random variable E as follows,

$$E = \begin{cases} 1 & \text{if } Q_R(\mathbf{x}_0) = Q_R(\mathbf{x}_1) = ... = Q_R(\mathbf{x}_n) \\ 0 & \text{otherwise} \end{cases}$$

With this, using the well known properties of the entropy (and conditional entropy), we have the following equality,

$$\begin{aligned} H\Big(Q_R(\mathbf{x}_1), ..., Q_R(\mathbf{x}_n), E \,\Big|\, Q_R(\mathbf{x}_0)\Big) \quad &= \quad H\Big(Q_R(\mathbf{x}_1), ..., Q_R(\mathbf{x}_n) \,\Big|\, Q_R(\mathbf{x}_0)\Big) \\ &\quad + H\Big(E \,\Big|\, Q_R(\mathbf{x}_0), Q_R(\mathbf{x}_1), ..., Q_R(\mathbf{x}_n)\Big) \end{aligned}$$

but, since E is a deterministic function of $Q_R(\mathbf{x}_0), Q_R(\mathbf{x}_1), ..., Q_R(\mathbf{x}_n)$, we have

$$H\Big(E \,\Big|\, Q_R(\mathbf{x}_0), Q_R(\mathbf{x}_1), ..., Q_R(\mathbf{x}_n)\Big) \quad = \quad 0$$

On the other hand, we can also write the following,

$$H\Big(Q_R(\mathbf{x}_1), ..., Q_R(\mathbf{x}_n), E \,\Big|\, Q_R(\mathbf{x}_0)\Big) \quad = \quad H\Big(E \,\Big|\, Q_R(\mathbf{x}_0)\Big) + H\Big(Q_R(\mathbf{x}_1), ..., Q_R(\mathbf{x}_n) \,\Big|\, Q_R(\mathbf{x}_0), E\Big)$$

Note that, since

$$\frac{1}{2} \quad \leq \quad \rho(\varepsilon_1, Q_R) \quad \leq \quad \mathbb{P}(E=1)$$

therefor,

$$H\Big(E \,\Big|\, Q_R(\mathbf{x}_0)\Big) \quad \leq \quad H(E) \quad \leq \quad H(\rho(\varepsilon_1, Q_R))$$

With all this, we have

$$H\Big(Q_R(\mathbf{x}_1), ..., Q_R(\mathbf{x}_n) \,\Big|\, Q_R(\mathbf{x}_0)\Big) \quad \leq \quad H(\rho(\varepsilon_1, Q_R)) + H\Big(Q_R(\mathbf{x}_1), ..., Q_R(\mathbf{x}_n) \,\Big|\, Q_R(\mathbf{x}_0), E\Big)$$

Finally, we can note the next equation,

$$\begin{aligned} H\Big(Q_R(\mathbf{x}_1), ..., Q_R(\mathbf{x}_n) \,\Big|\, Q_R(\mathbf{x}_0), E\Big) \quad = \quad & \sum_{q \in \mathcal{E}} \mathbb{P}\left(Q_R(\mathbf{x}_0) = q \,,\, E = 1\right) H\Big(Q_R(\mathbf{x}_1), ..., Q_R(\mathbf{x}_n) \,\Big|\, Q_R(\mathbf{x}_0) = q, E = 1\Big) \\ & + \mathbb{P}\left(Q_R(\mathbf{x}_0) = q \,,\, E = 0\right) H\Big(Q_R(\mathbf{x}_1), ..., Q_R(\mathbf{x}_n) \,\Big|\, Q_R(\mathbf{x}_0) = q, E = 0\Big) \end{aligned}$$

But note that, if $E = 1$, then, all the transformed points have the same value, so, if we know the value of $Q_R(\mathbf{x}_0)$, we can determine the value of the remaining points, so

$$H\Big(Q_R(\mathbf{x}_1), ..., Q_R(\mathbf{x}_n) \,\Big|\, Q_R(\mathbf{x}_0) = q, E = 1\Big) \quad = \quad 0$$

On the other hand, if $E = 0$, we know that, there one case that can not happen. Thus, we can bound the othe conditional entropy term as follows,

$$H\Big(Q_R(\mathbf{x}_1), ..., Q_R(\mathbf{x}_n) \,\Big|\, Q_R(\mathbf{x}_0) = q, E = 0\Big) \quad \leq \quad \log\Big(|\mathcal{E}|^n - 1\Big)$$

Overall, we have the following inequality

$$H\Big(Q_R(\mathbf{x}_1), ..., Q_R(\mathbf{x}_n) \,\Big|\, Q_R(\mathbf{x}_0), E\Big) \quad \leq \quad \mathbb{P}(E=0) \, \log\Big(|\mathcal{E}|^n - 1\Big)$$

but, since $\rho(\varepsilon, Q_R) \leq \mathbb{P}(E = 1)$, we have $\mathbb{P}(E = 0) \leq 1 - \rho(\varepsilon_1, Q_R)$, obtaining

$$H\Big(Q_R(\mathbf{x}_1), ..., Q_R(\mathbf{x}_n) \,\Big|\, Q_R(\mathbf{x}_0), E\Big) \quad \leq \quad (1 - \rho(\varepsilon_1, Q_R)) \, \log\Big(|\mathcal{E}|^n - 1\Big)$$

getting our result.

□

B.2 Proof Theorem 4.3.1

Theorem 4.3.1.

Indistinguishability. If (k, A, U) satisfies the conditions to compose a Gaussian Modular Hash function $Q_{k,A,U}$, then, for all $\mathbf{x} \in \mathbb{R}^N$ and for all $\mathbf{i} \in \mathbb{Z}_k^M$ we have

$$\mathbb{P}\big(Q_{k,A,U}(\mathbf{x})\big) \quad = \quad \frac{1}{k^M}$$

Proof

First, we need to prove the following proposition,

Proposition. *Let X be a continuous random variable, and U a continuous uniform random variable between 0 and k. If X and U are independent, then $X+U(\mathrm{mod}\, k)$ is uniformly distributed between 0 and k.*

Proof Proposition.

It is easy to see that, the density function of $X + U$ is given by

$$f_{X+U}(x) \quad = \quad \frac{1}{k}\left[F_X(x+k) - F_X(x)\right]$$

where F_X is the cdf of X. Then, if $x \in [0, k]$

$$\begin{aligned} f_{X+U \pmod k}(x) &= \sum_{i\in\mathbb{Z}} f_{X+U}(x+ik) \\ &= \sum_{i\in\mathbb{Z}} \frac{1}{k}\left[F_X(x+k+ik) - F_X(x+ik)\right] \\ &= \frac{1}{k}\sum_{i\in\mathbb{Z}} \left[F_X(x+(i+1)k) - F_X(x+ik)\right] \end{aligned}$$

Clearly we have a telescope summation. Therefore,

$$\begin{aligned} f_{X+U \pmod k}(x) &= \frac{1}{k}\left[\left(\lim_{i\to\infty} F(x+ik)\right) - \left(\lim_{i\to-\infty} F(x+ik)\right)\right] \\ &= \frac{1}{k} \end{aligned}$$

□

Proof Theorem

Since all the hash components are independent, we can just focus in a single value, which is equivalent to consider the number of columns in A as $M = 1$.

Notice that

$$\lfloor A\mathbf{x} + U \rfloor \,(\mathrm{mod}\, k) \quad = \quad \lfloor A\mathbf{x} + U\,(\mathrm{mod}\, k) \rfloor \tag{B.1}$$

and, by the previous proposition, if X is a random variable and U a uniform between 0 and k, then $X + U(\mathrm{mod}\, k)$ is distributed as a Uniform between 0 and k. Then, $\lfloor A\mathbf{x} + U(\mathrm{mod}\, k) \rfloor$ is uniformly distributed over $\mathbb{Z}_k$

□

B.3 Proof Theorem 4.3.2

Theorem 4.3.2.

Limited Leakage Property. If (k, A, U) satisfies the conditions to compose a Gaussian Modular Hash function $Q_{k,A,U}$, then, there exist positive constants C and γ such that for any pair of points $\mathbf{x}_1$, $\mathbf{x}_2 \in \mathbb{R}^N$ the mutual information between $Q_{k,A,U}(\mathbf{x}_1)$ and $Q_{k,A,U}(\mathbf{x}_2)$ is bounded as follows,

$$I\big(Q_{k,A,U}(\mathbf{x}_1)\,;\, Q_{k,A,U}(\mathbf{x}_2)\big) \quad \leq \quad C \exp\left(-\gamma\sigma\|\mathbf{x}_1 - \mathbf{x}_2\|\right) \tag{B.2}$$

Proof

Similar to the previous case, since all the hash components are independent, we can just focus in a single value, which is equivalent to consider the number of columns in A as $M = 1$. Hence, the a bound for the mutual information must just be multiplied by M for the more generic setting.

To prove this theorem, we need to demonstrate several previous results:

Lemma. Consider that (k, A, U) composes a Gaussian Modular Hash function, and $\mathbf{x}_1$ and $\mathbf{x}_2$ are any points in $\mathbb{R}^N$. If we define the random variables $Q_1 = Q_{k,A,U}(\mathbf{x}_1)$ and $Q_2 = Q_{k,A,U}(\mathbf{x}_2)$, then the following statements are true:

1. $\forall i \in \mathbb{Z}_k$, $\mathbb{P}(Q_1 - Q_2 = i) = \mathbb{P}(Q_1 - Q_2 = -i)$
2. If $n_1 - m_1 = n_2 - m_2$, then $\mathbb{P}(Q_1 = n_1\,,\, Q_2 = m_1) = \mathbb{P}(Q_1 = n_2\,,\, Q_2 = m_2)$
3. If $n_1 - m_1 = n_2 - m_2 \pm k$, then $\mathbb{P}(Q_1 = n_1\,,\, Q_2 = m_1) = \mathbb{P}(Q_1 = n_2\,,\, Q_2 = m_2)$

Proof Lemma

1. We know that, as in the previous arguments, that

$$\mathbb{P}(Q_1 - Q_2 = i) \quad = \quad \int_0^\infty g_{i,k}(l) f_L(l)\,\mathrm{d}l \tag{B.3}$$

where f_L is the density function of $L \sim \mathcal{N}\left(0\,,\,\sigma^2\|\mathbf{x}_1 - \mathbf{x}_2\|^2\right)$, and $g_{i,k} = tri * h_{i,k}$ where

$$h_{i,k}(x) \quad = \quad \sum_{j=0}^{\infty} \delta(x - i - jk) + \sum_{j=0}^{\infty} \delta(x + i + jk) \tag{B.4}$$

Similarly

$$\mathbb{P}(Q_1 - Q_2 = -i) = \int_{-\infty}^{0} g_{i,k}(l) f_L(l)\,\mathrm{d}l \tag{B.5}$$

and by the symmetry of $g_{i,k}$ and f_L we have the result. This can be seen in the figure B.1.

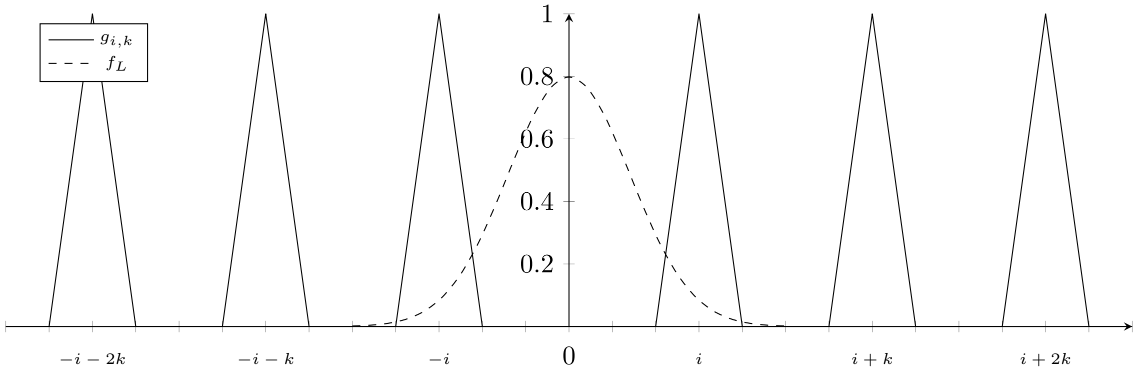


Figure B.1: Illustration of functions $g_{i,k}$ and f_L

2. First, we have to notice that, if X and Y are continuous random variables, and $U \sim \text{Unif}(0,k)$, then

$$\begin{aligned}
\mathbb{P}(X + U \leq x\,,\, Y + U \leq y) &= \mathbb{P}(X \leq x - U\,,\, Y \leq y - U) \\
&= \int_0^k \mathbb{P}(X \leq x - u\,,\, Y \leq y - u)\frac{1}{k}\,\mathrm{d}u \\
&= \frac{1}{k}\int_0^k F_{X,Y}(x - u, y - u)\,\mathrm{d}u \\
&= \frac{1}{k}\int_{x-k}^{x} F_{X,Y}(t, y - x + t)\,\mathrm{d}t \\
&= \frac{1}{k}\int_{y-k}^{y} F_{X,Y}(x - y + t, t)\,\mathrm{d}t
\end{aligned}$$

Besides, if $n, m \in \mathbb{Z}$

$$\begin{aligned}\mathbb{P}\left(\lfloor X+U \rfloor = n\,,\, \lfloor Y+U \rfloor = m\right) &= \mathbb{P}(X+U \leq n+1\,,\, Y+U \leq m+1)\\ &-\mathbb{P}(X+U \leq n+1\,,\, Y+U \leq m)\\ &-\mathbb{P}(X+U \leq n\,,\, Y+U \leq m+1)\\ &+\mathbb{P}(X+U \leq n\,,\, Y+U \leq m)\end{aligned}$$

then, if $n, m \in \mathbb{Z}$ such that $0 \leq n < k$ and $0 \leq m < k$

$$\begin{aligned}\mathbb{P}(\lfloor X+U \rfloor = n\,,\, \lfloor Y+U \rfloor = m \bmod k) &= \sum_{i,j \in \mathbb{Z}} \mathbb{P}\left(\lfloor X+U \rfloor = n+ik\,,\, \lfloor Y+U \rfloor = m+jk\right)\\ &= \sum_{i,j \in \mathbb{Z}} [\mathbb{P}(X+U \leq n+ik+1\,,\, Y+U \leq m+jk+1)\\ &\quad -\mathbb{P}(X+U \leq n+ik+1\,,\, Y+U \leq m+jk)\\ &\quad -\mathbb{P}(X+U \leq n+ik\,,\, Y+U \leq m+jk+1)\\ &\quad +\mathbb{P}(X+U \leq n+ik\,,\, Y+U \leq m+jk)]\\ &= \sum_{i,j \in \mathbb{Z}} \frac{1}{k} \int_{n+ik+1-k}^{n+ik+1} F_{X,Y}(t, m-n+(j-i)k+t)\,\mathrm{d}t\\ &\quad -\frac{1}{k} \int_{n+ik+1-k}^{n+ik+1} F_{X,Y}(t, m-n+(j-i)k-1+t)\,\mathrm{d}t\\ &\quad -\frac{1}{k} \int_{n+ik-k}^{n+ik} F_{X,Y}(t, m-n+(j-i)k+1+t)\,\mathrm{d}t\\ &\quad +\frac{1}{k} \int_{n+i+k-k}^{n+ik} F_{X,Y}(t, m-n+(j-i)k+t)\,\mathrm{d}t\end{aligned}$$

Doing a change of variables in the indexes in the summation, $j = j - i$ and $i = i$, we have

$$\begin{aligned}\mathbb{P}(\lfloor X+U \rfloor = n\,,\, \lfloor Y+U \rfloor = m \bmod k) &= \sum_{i,j \in \mathbb{Z}} \frac{1}{k} \int_{n+(i-1)k+1}^{n+ik+1} F_{X,Y}(t, m-n+jk+t)\\ &\quad -F(t, m-n+jk-1+t)\,\mathrm{d}t\\ &\quad -\frac{1}{k} \int_{n+(i-1)k}^{n+ik} F_{X,Y}(t, m-n+jk+1+t)\\ &\quad -F_{X,Y}(t, m-n+jk+t)\,\mathrm{d}t\\ &= \sum_{j \in \mathbb{Z}} \frac{1}{k} \int_{-\infty}^{\infty} 2F_{X,Y}(t, m-n+jk+t)\\ &\quad -F_{X,Y}(t, m-n+jk-1+t)\\ &\quad -F_{X,Y}(t, m-n+jk+1+t)\,\mathrm{d}t\end{aligned}$$

Thus, this last expression depends on the difference between m and n.

3. From the last expression, we have

$$\begin{aligned}
\mathbb{P}(\lfloor X+U \rfloor = n_1 \,,\, \lfloor Y+U \rfloor = m_1 \bmod k) &= \sum_{j\in\mathbb{Z}} \frac{1}{k} \int_{-\infty}^{\infty} 2F_{X,Y}(t, m_1 - n_1 + jk + t) \\
&\quad -F_{X,Y}(t, m_1 - n_1 + jk - 1 + t) \\
&\quad -F_{X,Y}(t, m_1 - n_1 + jk + 1 + t)\,\mathrm{d}t \\
&= \sum_{j\in\mathbb{Z}} \frac{1}{k} \int_{-\infty}^{\infty} 2F_{X,Y}(t, m_2 - n_2 \pm k + jk + t) \\
&\quad -F_{X,Y}(t, m_2 - n_2 \pm k + jk - 1 + t) \\
&\quad -F_{X,Y}(t, m_2 - n_2 \pm k + jk + 1 + t)\,\mathrm{d}t \\
&= \sum_{j\in\mathbb{Z}} \frac{1}{k} \int_{-\infty}^{\infty} 2F_{X,Y}(t, m_2 - n_2 + jk + t) \\
&\quad -F_{X,Y}(t, m_2 - n_2 + jk - 1 + t) \\
&\quad -F_{X,Y}(t, m_2 - n_2 + jk + 1 + t)\,\mathrm{d}t \\
&= \mathbb{P}(\lfloor X+U \rfloor = n_2 \,,\, \lfloor Y+U \rfloor = m_2 \bmod k)
\end{aligned}$$

□

We also need to prove the next result

Proposition 1. If (k, A, U) composes a Gaussian Modular Hash function, then $\forall \mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^N$ and $\forall i \in \mathbb{Z}_k$, defining the random variables $Q_i = Q_{k,A,U}(\mathbf{x}_i)$, we have

$$\mathbb{P}\Big(\big|Q_1 - Q_2\big| = i\Big) + \mathbb{P}\Big(\big|Q_1 - Q_2\big| = k - i\Big) \;\leq\; 2 \cdot \mathbb{P}(Q_1 = Q_2) \tag{B.6}$$

Proof of Proposition 1

We know that

$$\mathbb{P}\Big(\big|Q_1 - Q_2\big| = i\Big) = \int_{-\infty}^{\infty} f_L(l) \cdot (tri * h_{i,k}\,(l))\;\mathrm{d}l \tag{B.7}$$

where f_L is the density function of $L \sim \mathcal{N}\left(0\,,\,\sigma^2\|\mathbf{x}_1 - \mathbf{x}_2\|^2\right)$, and $g_{i,k} = tri * h_{i,k}$ where

$$h_{i,k}(x) = \sum_{j=0}^{\infty} \delta(x - i - jk) + \sum_{j=0}^{\infty} \delta(x + i + jk)$$

Therefore

$$\begin{aligned}
\mathbb{P}\Big(\big|Q_1 - Q_2\big| = i\Big) + \mathbb{P}\Big(\big|Q_1 - Q_2\big| = k - i\Big) &= \int_{-\infty}^{\infty} f_L(l) \cdot \Big(tri * h_{i,k}\,(l)\Big)\,dl + \int_{-\infty}^{\infty} f_L(l) \cdot \Big(tri * h_{k-i,k}\,(l)\Big)\;\mathrm{d}l \\
&= \int_{-\infty}^{\infty} f_L(l) \cdot \Big(tri * \Big(h_{i,k} + h_{k-i,k}\Big)\,(l)\Big)\;\mathrm{d}l
\end{aligned}$$

However,

$$\begin{aligned}
h_{i,k}(x) + h_{k-i,k}(x) &= \sum_{j=0}^{\infty}\delta(x-i-jk) + \sum_{j=0}^{\infty}\delta(x+i+jk) + \sum_{j=0}^{\infty}\delta(x-(k-i)-jk) + \sum_{j=0}^{\infty}\delta(x+(k-i)+jk) \\
&= \sum_{j=0}^{\infty}\delta(x-i-jk) + \sum_{j=0}^{\infty}\delta(x+i+jk) + \sum_{j=0}^{\infty}\delta(x+i-k-jk) + \sum_{j=0}^{\infty}\delta(x+i+k+jk) \\
&= \sum_{j=0}^{\infty}\delta(x-i-jk) + \sum_{j=0}^{\infty}\delta(x+i+jk) + \sum_{j=1}^{\infty}\delta(x+i-jk) + \sum_{j=1}^{\infty}\delta(x+i+jk) \\
&= \sum_{j\in\mathbb{Z}}\delta(x-i-jk) + \sum_{j\in\mathbb{Z}}\delta(x+i+jk) \\
&= \sum_{j\in\mathbb{Z}}\delta(x+jk) * (\delta(x-i) + \delta(x+i))
\end{aligned}$$

Then, applying the Parseval's Theorem, we have

$$\begin{aligned}
\mathbb{P}\Big(\big|Q_1 - Q_2\big| = i\Big) + \mathbb{P}\Big(\big|Q_1 - Q_2\big| = k-i\Big) &= \int_{-\infty}^{\infty} \hat{f}_L(\xi)\cdot(tri * (\widehat{h_{i,k} + h_{k-i,k}}))(\xi)\,\mathrm{d}\xi \\
&= \int_{-\infty}^{\infty} \hat{f}_L(\xi)\cdot \hat{tri}(\xi)(\widehat{h_{i,k} + h_{k-i,k}})(\xi)\,\mathrm{d}\xi \\
&= \int_{-\infty}^{\infty} \hat{f}_L(\xi)\cdot \hat{tri}(\xi)\left(\widehat{\sum_{j\in\mathbb{Z}}\delta(x+jk) * (\delta(x-i)+\delta(x+i))}\right)(\xi)\,\mathrm{d}\xi \\
&= \int_{-\infty}^{\infty} \hat{f}_L(\xi)\cdot \hat{tri}(\xi)\cdot\frac{1}{k}\sum_{j\in\mathbb{Z}}\delta\left(\xi+\frac{j}{k}\right)\cdot 2\cos(2\pi\xi)\,\mathrm{d}\xi \\
&\leq 2\int_{-\infty}^{\infty} \hat{f}_L(\xi)\cdot \hat{tri}(\xi)\cdot\frac{1}{k}\sum_{j\in\mathbb{Z}}\delta\left(\xi+\frac{j}{k}\right)\mathrm{d}\xi \\
&= 2\int_{-\infty}^{\infty} e^{-2\left(\frac{\pi\|x_1-x_2\|\xi}{\delta}\right)^2}\cdot sinc^2(\xi)\cdot\frac{1}{k}\sum_{j\in\mathbb{Z}}\delta\left(\xi+\frac{j}{k}\right)\mathrm{d}\xi \\
&= 2\cdot\mathbb{P}\big(Q_1 = Q_2\big)
\end{aligned}$$

□

The last result we need to prove our theorem is the following,

Proposition 2. If (k, A, U) composes a Gaussian Modular Hash function, then $\forall \mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^N$ and $\forall n\,,\, m \in \mathbb{Z}_k$, defining the random variables $Q_i = Q_{k,A,U}(\mathbf{x}_i)$, we have

$$\mathbb{P}\left(Q_1 = n\,,\, Q_2 = m\right) \quad\leq\quad \frac{\mathbb{P}\big(Q_1 = Q_2\big)}{k} \tag{B.8}$$

Proof of Proposition 2.

Let i be equal to $n-m$. From the previous lemma, we know that if $|n-m| = |n_2-m_2|$, so

$$\mathbb{P}\left(Q_1 = n\,,\, Q_2 = m\right) \quad = \quad \mathbb{P}\left(Q_1 = n_2\,,\, Q_2 = m_2\right) \tag{B.9}$$

Then

$$\begin{aligned}\mathbb{P}\left(|Q_1 - Q_2| = i\right) &= \sum_{|n_2-m_2|=i} \mathbb{P}\left(Q_1 = n_2\,,\, Q_2 = m_2\right)\\ &= 2\cdot(k-i)\cdot\mathbb{P}\left(Q_1 = n\,,\, Q_2 = m\right)\end{aligned}$$

Moreover, from the previous lemma we can also see that, if $|n-m| = k - |n_2 - m_2|$

$$\mathbb{P}\left(Q_1 = n\,,\, Q_2 = m\right) \quad = \quad \mathbb{P}\left(Q_1 = n_2\,,\, Q_2 = m_2\right)$$

with this, we can get the following

$$\begin{aligned}\mathbb{P}\left(|Q_1 - Q_2| = k-i\right) &= \sum_{|n_2-m_2|=k-i} \mathbb{P}\left(Q_1 = n_2\,,\, Q_2 = m_2\right)\\ &= 2\cdot i\cdot\mathbb{P}\left(Q_1 = n\,,\, Q_2 = m\right)\end{aligned}$$

Thus,

$$\begin{aligned}\mathbb{P}\Big(\big|Q_1 - Q_2\big| = i\Big) + \mathbb{P}\Big(\big|Q_1 - Q_2\big| = k-i\Big) &= 2\cdot(k-i)\cdot\mathbb{P}\left(Q_1 = n\,,\, Q_2 = m\right)\\ &\quad +2\cdot i\cdot\mathbb{P}\left(Q_1 = n\,,\, Q_2 = m\right)\\ &= 2k\mathbb{P}\left(Q_1 = n\,,\, Q_2 = m\right)\end{aligned}$$

And using the previous proposition, we have

$$2k\,\mathbb{P}\left(Q_1 = n\,,\, Q_2 = m\right) \quad \leq \quad 2\,\mathbb{P}\left(Q_1 = Q_2\right)$$

□

Proof of Theorem Defining the random variables $Q_i = Q_{k,A,U}(\mathbf{x}_i)$,, we have that the mutual information between Q_1 and Q_2 is given by

$$I(Q_1\,;\,Q_2) \quad = \quad \sum_{n,m=0}^{k-1} \mathbb{P}(Q_1 = n\,,\, Q_2 = m)\log\left(\frac{\mathbb{P}(Q_1 = n\,,\, Q_2 = m)}{\mathbb{P}(Q_1 = n)\,\mathbb{P}(Q_2 = m)}\right)$$

but, we know that for any $i \in \mathbb{Z}_k$ we have

$$\mathbb{P}(Q_1 = i) \quad = \quad \mathbb{P}(Q_2 = i) \quad = \quad \frac{1}{k}$$

Then

$$I(Q_1\,;\,Q_2) \quad = \quad \sum_{n,m=0}^{k-1} \mathbb{P}(Q_1 = n\,,\,Q_2 = m)\log\left(\mathbb{P}(Q_1 = n\,,\,Q_2 = m)\right) - \log\frac{1}{k^2}$$

Moreover, since

$$\mathbb{P}\left(Q_1 = n\,,\,Q_2 = m\right) \quad \leq \quad \frac{\mathbb{P}(Q_1 = Q_2)}{k}$$

we have

$$\begin{aligned} I(Q_1\,;\,Q_2) \quad &\leq \quad \sum_{n,m=0}^{k-1} \mathbb{P}(Q_1 = n\,,\,Q_2 = m)\log\left(\frac{\mathbb{P}(Q_1 = Q_2)}{k}\right) - \log\frac{1}{k^2} \\ &= \quad \log\left(\frac{\mathbb{P}(Q_1 = Q_2)}{k}\right) - \log\frac{1}{k^2} \end{aligned}$$

and, since we know (see theorem 5.2.2)

$$\mathbb{P}(Q_1 = Q_2) \quad \leq \quad \frac{1}{k}\left(1 + \frac{k^2}{3}e^{-2\left(\frac{\pi\sigma\|\mathbf{x}_1 - \mathbf{x}_2\|}{k}\right)^2}\right)$$

and finally, using the Mean Value Theorem, we have

$$\begin{aligned} I(Q_1\,;\,Q_2) \quad &\leq \quad \log\left(\frac{1}{k^2}\left(1 + \frac{k^2}{3}e^{-2\left(\frac{\pi\sigma\|\mathbf{x}_1 - \mathbf{x}_2\|}{k}\right)^2}\right)\right) - \log\frac{1}{k^2} \\ &\leq \quad k^2\cdot\left(\frac{1}{k^2} + \frac{1}{3}e^{-2\left(\frac{\pi\sigma\|\mathbf{x}_1 - \mathbf{x}_2\|}{k}\right)^2} - \frac{1}{k^2}\right) \\ &= \quad \frac{k^2}{3}\exp\left(-2\left(\frac{\pi\sigma\|\mathbf{x}_1 - \mathbf{x}_2\|}{k}\right)^2\right) \end{aligned}$$

□

B.4 Proof Theorem 4.3.3

Theorem 4.3.3.

If (k, A, U) satisfies the conditions to compose a Gaussian Modular Hash function $Q_{k,A,U}$, where σ^2 is the variance a_{ij}, then, the degree of persistence is given by

$$\rho(\varepsilon, Q_{k,A,U}) = \left(\frac{\gamma\left(\frac{N}{2}, \frac{1}{4\varepsilon^2\sigma^2}\right)}{\Gamma\left(\frac{N}{2}\right)} - 2\sqrt{2}\sigma\varepsilon \frac{\gamma\left(\frac{N+1}{2}, \frac{1}{4\varepsilon^2\sigma^2}\right)}{\Gamma\left(\frac{N}{2}\right)} \right)^M$$

where $\gamma(s,x) = \int_0^x t^{s-1}e^{-t}\mathrm{d}t$ and it is known as the lower incomplete gamma function, and $\Gamma(z) = \int_0^\infty x^{z-1}e^{-x}\mathrm{d}x$ is the well known gamma function.

Proof

Without loss of generality, we can consider $M = 1$.

Using the same principle if theorem 4.3.2, we can show that, since the hash is based on a random projection, the following

$$\rho(\varepsilon, Q_R) = \int_0^1 (1-x) f_X(x)\mathrm{d}x$$

where f_X is the density function of the random variable X, which is $X = 2\varepsilon\|A\|$ Note that, the cumulative probability function of X is given by

$$\begin{aligned} F_X(x) &= \mathbb{P}(2\varepsilon\|A\| \leq x) \\ &= \mathbb{P}\left(W \leq \frac{x^2}{4\varepsilon^2\sigma^2}\right) \end{aligned}$$

where W is a sum N of square independent Gaussians with zero-mean and standard deviation 1, i.e., W is distributed as a χ^2_N. Thus, it is possible to prove that

$$\begin{aligned} \rho(\varepsilon, Q_R) &= F_X(1) - \int_0^1 x f_X(x)\mathrm{d}x \\ &= F_{\chi^2_N}\left(\frac{1}{4\varepsilon^2\sigma^2}\right) - \int_0^1 f_{\chi^2_N}\left(\frac{x^2}{4\varepsilon^2\sigma^2}\right)\frac{x^2}{2\varepsilon^2\sigma^2}\mathrm{d}x \end{aligned}$$

and doing the change of variables $y = \frac{x^2}{4\varepsilon^2\sigma^2}$ it is possible to show that

$$\rho(\varepsilon, Q_{k,A,U}) = \frac{\gamma\left(\frac{N}{2}, \frac{1}{4\varepsilon^2\sigma^2}\right)}{\Gamma\left(\frac{N}{2}\right)} - 2\sqrt{2}\sigma\varepsilon \frac{\gamma\left(\frac{N+1}{2}, \frac{1}{4\varepsilon^2\sigma^2}\right)}{\Gamma\left(\frac{N}{2}\right)}$$

For values of $M > 1$, we just need to multiply M times, since is a condition over M independent random variables.

□

B.5 Proof Theorem 4.3.4

Theorem 4.3.4.

If we consider the $Q_{2,A,U}$ is Gaussian Modular Hash, then the mutual information between $Q_{2,A,U}(\mathbf{x}_0)$ and $(Q_{2,A,U}(\mathbf{x}_1), Q_{2,A,U}(\mathbf{x}_2))$ goes exponentially fast to 0 when the minimum distance from $\mathbf{x}_0$ to $\mathbf{x}_1$ and $\mathbf{x}_2$ goes to infinity.

Proof

First, we need to prove the following proposition

Proposition. Consider $R = (2, A, U)$ composes a Gaussian Modular Hash. Given any three points in $\mathbb{R}^N$, $\mathbf{x}_0$, $\mathbf{x}_1$ and $\mathbf{x}_2$, then

$$\begin{aligned}\mathbb{P}\left(Q_R(\mathbf{x}_0) = i_0\, , \, Q_R(\mathbf{x}_1) = i_1\, , \, Q_R(\mathbf{x}_2) = i_2\right) \quad = \quad & \frac{1}{2}\, \mathbb{P}\left(Q_R(\mathbf{x}_0) = i_0\, , \, Q_R(\mathbf{x}_1) = i_1\right) \\ & + \frac{1}{2}\, \mathbb{P}\left(Q_R(\mathbf{x}_0) = i_0\, , \, Q_R(\mathbf{x}_2) = i_2\right) \\ & + \frac{1}{2}\, \mathbb{P}\left(Q_R(\mathbf{x}_1) = i_1\, , \, Q_R(\mathbf{x}_2) = i_2\right) \\ & - \frac{1}{4}\end{aligned}$$

Proof of Proposition.

It is possible to show that

$$\mathbb{P}\Big(Q_R(\mathbf{x}_0) = i_0\, , \, Q_R(\mathbf{x}_1) = i_1\, , \, Q_R(\mathbf{x}_2) = i_2\Big) = \mathbb{P}\Big(Q_R(\mathbf{x}_0) = i_0{+}1\, , \, Q_R(\mathbf{x}_1) = i_1{+}1\, , \, Q_R(\mathbf{x}_2) = i_2{+}1\Big)$$

being the sum in modulo. Thus, we just need to focus of the half of the probability distribution.

Just to simplify the notation, we define $q_i := Q_2(\mathbf{x}_i)$. Thus, we have the following linear system

$$\begin{pmatrix} 0.5 \\ \mathbb{P}\big(q_1 = 0\, , \, q_2 = 0\big) \\ \mathbb{P}\big(q_0 = 0\, , \, q_2 = 0\big) \\ \mathbb{P}\big(q_0 = 0\, , \, q_1 = 0\big) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbb{P}\big(q_0 = 0\, , \, q_1 = 0\, , \, q_2 = 0\big) \\ \mathbb{P}\big(q_0 = 1\, , \, q_1 = 0\, , \, q_2 = 0\big) \\ \mathbb{P}\big(q_0 = 0\, , \, q_1 = 1\, , \, q_2 = 0\big) \\ \mathbb{P}\big(q_0 = 0\, , \, q_1 = 0\, , \, q_2 = 1\big) \end{pmatrix}$$

This linear system is invertible and it can be written as follows

$$\begin{pmatrix} \mathbb{P}\big(q_0 = 0\,,\, q_1 = 0\,,\, q_2 = 0\big) \\ \mathbb{P}\big(q_0 = 1\,,\, q_1 = 0\,,\, q_2 = 0\big) \\ \mathbb{P}\big(q_0 = 0\,,\, q_1 = 1\,,\, q_2 = 0\big) \\ \mathbb{P}\big(q_0 = 0\,,\, q_1 = 0\,,\, q_2 = 1\big) \end{pmatrix} = \begin{pmatrix} -0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & -0.5 & -0.5 \\ 0.5 & -0.5 & 0.5 & -0.5 \\ 0.5 & -0.5 & -0.5 & 0.5 \end{pmatrix} \begin{pmatrix} 0.5 \\ \mathbb{P}\big(q_1 = 0\,,\, q_2 = 0\big) \\ \mathbb{P}\big(q_0 = 0\,,\, q_2 = 0\big) \\ \mathbb{P}\big(q_0 = 0\,,\, q_1 = 0\big) \end{pmatrix}$$

We also need to note that

$$-\mathbb{P}(q_i = 0, q_j = 0) \quad = \quad \mathbb{P}(q_i = 0, q_j = 1) - \frac{1}{2}$$

Thus, the theorem is proved. □

Proof of Theorem

First, using the previous proposition, definig $Q_i = Q_R(\mathbf{x}_i)$, we can note that

$$\frac{\mathbb{P}\Big(Q_0 = i_0\,,\, Q_1 = i_1\,,\, Q_2 = i_2\Big)}{\mathbb{P}\Big(Q_0 = i_0\Big)\,\mathbb{P}\Big(Q_1 = i_1\,,\, Q_2 = i_2\Big)} \quad = \quad 1 + \frac{\mathbb{P}\Big(Q_0 = i_0\,,\, Q_1 = i_1\Big) + \mathbb{P}\Big(Q_0 = i_0\,,\, Q_2 = i_2\Big) - 1/2}{\mathbb{P}\Big(Q_1 = i_1\,,\, Q_2 = i_2\Big)}$$

Moreover, we can see that, always

$$\mathbb{P}\Big(Q_R(\mathbf{x}) = i\,,\, Q_R(\mathbf{z}) = j\Big) \quad \leq \quad \frac{1}{2}\mathbb{P}\Big(Q_R(\mathbf{x}) = Q_R(\mathbf{z})\Big) \quad \leq \quad \frac{1}{4}\left(1 + \frac{4}{3}\exp\left(-2\left(\frac{\pi\sigma\|\mathbf{x} - \mathbf{z}\|}{2}\right)^2\right)\right)$$

Thus,

$$\frac{\mathbb{P}\Big(Q_0 = i_0\,,\, Q_1 = i_1\,,\, Q_2 = i_2\Big)}{\mathbb{P}\Big(Q_0 = i_0\Big)\,\mathbb{P}\Big(Q_1 = i_1\,,\, Q_2 = i_2\Big)} \quad \leq \quad 1 + \frac{1}{3}\frac{\exp\left(-2\left(\frac{\pi\sigma\|\mathbf{x}_0 - \mathbf{x}_1\|}{2}\right)^2\right) + \exp\left(-2\left(\frac{\pi\sigma\|\mathbf{x}_0 - \mathbf{x}_2\|}{2}\right)^2\right)}{\mathbb{P}\Big(Q_1 = i_1\,,\, Q_2 = i_2\Big)}$$

Since $\mathbf{x}_1$ and $\mathbf{x}_2$ are fixed, we can consider $C = \min_{\{i_1, i_2\}} \mathbb{P}\Big(Q_1 = i_1\,,\, Q_2 = i_2\Big)$ and $\log(x) \leq x - 1$, we have

$$I(Q_0\,;\, Q_1, Q_2) \quad \leq \quad \frac{1}{3C}\left[\exp\left(-2\left(\frac{\pi\sigma\|\mathbf{x}_0 - \mathbf{x}_1\|}{2}\right)^2\right) + \exp\left(-2\left(\frac{\pi\sigma\|\mathbf{x}_0 - \mathbf{x}_2\|}{2}\right)^2\right)\right]$$

and we have our result □

B.6 Proof Theorem 4.3.5

Theorem 4.3.5.

Let $\mathbf{x}_0$ a point in $\mathbb{R}^N$. Consider the set of points $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\} \subseteq B(\mathbf{x}_1, \varepsilon)$. If we consider the $Q_{2,A,U}$ is Gaussian Modular Hash, then

$$\begin{aligned} I\Big(Q_R(\mathbf{x}_0)\ ;\ Q_R(\mathbf{x}_1), ..., Q_R(\mathbf{x}_n)\Big) \quad &\leq \quad I(Q_R(\mathbf{x}_0)\ ;\ Q_R(\mathbf{x}_1)) \\ &\quad +1 - \rho(\varepsilon, Q_R) + \mathbb{P}\Big(Q_R(\mathbf{x}_1) = Q_R(\mathbf{x}^*)\Big) \log \frac{1}{\rho(\varepsilon, Q_R)} \end{aligned}$$

where $\mathbf{x}^*$ is the farthest point to $\mathbf{x}_1$ in the set $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$.

Proof

Based on the definition of $\rho(\varepsilon, Q_R)$, we can observe that

$$\rho(\varepsilon, Q_R) \quad \leq \quad \mathbb{P}\Big(Q_R(\mathbf{x}_1) = Q_R(\mathbf{x}_2) = ... = Q_R(\mathbf{x}_n)\Big) \quad \leq \quad \mathbb{P}\Big(Q_R(\mathbf{x}_1) = Q_R(\mathbf{x}^*)\Big)$$

On the other hand, the summation that defines the mutual information can be splitted in two terms, when $Q_R(\mathbf{x}_1) = ... = Q_R(\mathbf{x}_n)$ and when not. We can analyze this last case. Note that, defining the random variable $Q_i = Q_R(\mathbf{x}_i)$, we have

$$\begin{aligned} &\sum_{i_0} \sum_{\text{not}(i_1=...=i_n)} \mathbb{P}\Big(Q_0 = i_0,\ , Q_1 = i_1\ , \ ...\ , \ Q_n = i_n\Big) \log \frac{\mathbb{P}\Big(Q_0 = i_0,\ , Q_1 = i_1\ , \ ...\ , \ Q_n = i_n\Big)}{\mathbb{P}\Big(Q_0 = i_0\Big)\, \mathbb{P}\Big(Q_1 = i_1\ , \ ...\ , \ Q_n = i_n\Big)} \\ &\leq \sum_{i_0} \sum_{\text{not}(i_1=...=i_n)} \mathbb{P}\Big(Q_0 = i_0,\ , Q_1 = i_1\ , \ ...\ , \ Q_n = i_n\Big) \log \frac{\mathbb{P}\Big(Q_1 = i_1\ , \ ...\ , \ Q_n = i_n\Big)}{1/2\, \mathbb{P}\Big(Q_1 = i_1\ , \ ...\ , \ Q_n = i_n\Big)} \\ &= \sum_{i_0} \sum_{\text{not}(i_1=...=i_n)} \mathbb{P}\Big(Q_0 = i_0,\ , Q_1 = i_1\ , \ ...\ , \ Q_n = i_n\Big) \\ &= 1 - \mathbb{P}\Big(Q_1 = ... = Q_n\Big) \quad \leq \quad 1 - \rho(\varepsilon, Q_R) \end{aligned}$$

The other case can be written as follows

$$\begin{aligned} &\sum_{i_0} \sum_{i} \mathbb{P}\Big(Q_0 = i_0\ , \ Q_1 = i, ..., Q_n = i\Big) \log \frac{\mathbb{P}\Big(Q_0 = i_0\ , \ Q_1 = i, ..., Q_n = i\Big)}{1/2\, \mathbb{P}\Big(Q_1 = i, ..., Q_n = i\Big)} \\ \leq \quad &\sum_{i_0} \sum_{i} \mathbb{P}\Big(Q_0 = i_0\ , \ Q_1 = i, ..., Q_n = i\Big) \log \frac{\mathbb{P}\Big(Q_0 = i_0\ , \ Q_1 = i\Big)}{1/2 \cdot 1/2 \cdot \rho(\varepsilon, Q_R)} \\ = \quad &\sum_{i_0} \sum_{i} \mathbb{P}\Big(Q_0 = i_0\ , \ Q_1 = i, ..., Q_n = i\Big) \log \frac{\mathbb{P}\Big(Q_0 = i_0\ , \ Q_1 = i\Big)}{1/2 \cdot 1/2} \\ &+ \sum_{i_0} \sum_{i} \mathbb{P}\Big(Q_0 = i_0\ , \ Q_1 = i, ..., Q_n = i\Big) \log \frac{1}{\rho(\varepsilon, Q_R)} \end{aligned}$$

But note that

$$\begin{aligned}
& \sum_{i_0}\sum_{i}\mathbb{P}\Big(Q_0=i_0\,,\,Q_1=i,...,Q_n=i\Big)\log\frac{\mathbb{P}\Big(Q_0=i_0\,,\,Q_1=i\Big)}{1/2\,\cdot 1/2} \\
\leq\ & \sum_{i_0}\sum_{i}\mathbb{P}\Big(Q_0=i_0\,,\,Q_1=i\Big)\log\frac{\mathbb{P}\Big(Q_0=i_0\,,\,Q_1=i\Big)}{1/2\,\cdot 1/2} \\
=\ & I(Q_R(\mathbf{x}_0)\,;\,Q_R(\mathbf{x}_1))
\end{aligned}$$

On the other hand

$$\begin{aligned}
& \sum_{i_0}\sum_{i}\mathbb{P}\Big(Q_0=i_0\,,\,Q_1=i,...,Q_n=i\Big)\log\frac{1}{\rho(\varepsilon,Q_R)} \\
=\ & \sum_{i}\mathbb{P}\Big(\,Q_1=i,...,Q_n=i\Big)\log\frac{1}{\rho(\varepsilon,Q_R)} \\
\leq\ & \mathbb{P}\Big(Q_R(\mathbf{x}_1)=Q_R(\mathbf{x}^*)\Big)\log\frac{1}{\rho(\varepsilon,Q_R)}
\end{aligned}$$

Therefore, we have

$$\begin{aligned}
I\Big(Q_R(\mathbf{x}_0)\,;\,Q_R(\mathbf{x}_1),...,Q_R(\mathbf{x}_n)\Big) \leq\ & I(Q_R(\mathbf{x}_0)\,;\,Q_R(\mathbf{x}_1)) \\
& +1-\rho(\varepsilon,Q_R)+\mathbb{P}\Big(Q_R(\mathbf{x}_1)=Q_R(\mathbf{x}^*)\Big)\log\frac{1}{\rho(\varepsilon,Q_R)}
\end{aligned}$$

□

B.7 Proof Theorem 5.2.1

Theorem 5.2.1.

If $\mathbf{x}_1$ and $\mathbf{x}_2 \in \mathbb{R}^N$ and $R = (k, A, U)$ a random key to compose a σ-Gaussian Modular Hash, then the probability of getting a collision in j-component is given by the following expression,

$$\mathbb{P}\Big(Q_R(\mathbf{x}_1)_j = Q_R(\mathbf{x}_2)_j\Big) \quad = \quad \frac{1}{k} + \frac{2}{k}\sum_{i=1}^{\infty} \operatorname{sinc}^2\left(\frac{i}{k}\right) \exp\left(-2\left(\frac{\pi\,\sigma\, i\,\|\mathbf{x}_1 - \mathbf{x}_2\|}{k}\right)^2\right)$$

where $\operatorname{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$.

Proof

Without loss of generality, we assume A is a $1 \times N$ matrix. Using the same analysis as in the proof of theorem 4.3.2 , we have that the $\mathbb{P}(Q_R(\mathbf{x}_1) = Q_R(\mathbf{x}_2) \mid \|A(\mathbf{x}_1 - \mathbf{x}_2)\|)$ is given by the function g_k depending on $L = \|A(\mathbf{x}_1 - \mathbf{x}_2)\|$. In particular, we have

$$g_k(x) = \begin{cases} x - ik + 1 & \text{if } x \in [ik-1, ik] \text{ for some } i \\ -x + ik + 1 & \text{if } x \in [ik, ik+1] \text{ for some } i \\ 0 & \text{otherwise} \end{cases}$$

An illustration of this function is shown in the figure B.2. Besides, we know that the density

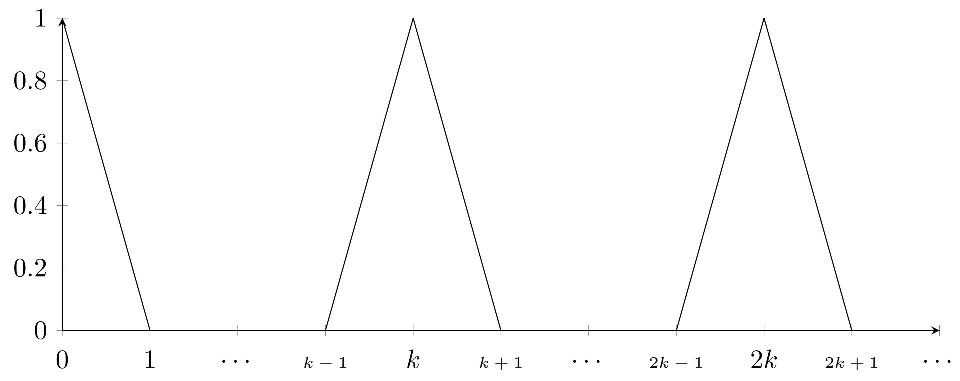


Figure B.2: Illustration of function g_k

function of L is

$$f_L(l) \quad = \quad \sqrt{\frac{2}{\pi}}\frac{1}{\sigma\|\mathbf{x}_1 - \mathbf{x}_2\|}\exp\left(-\frac{l^2}{2\sigma^2\|\mathbf{x}_1 - \mathbf{x}_2\|^2}\right) \tag{B.10}$$

Then

$$\mathbb{P}(Q_R(\mathbf{x}_1) = Q_R(\mathbf{x}_2)) \quad = \quad \int_0^\infty g_k(l) f_L(l)\,\mathrm{d}l$$

But, since both g_k and f_L are positive, we can extend these functions over real line, defining $\tilde{g}_k$ as

$$\tilde{g}_k(x) = \begin{cases} g_k(x) & \text{if} \quad x \geq 0 \\ g_k(-x) & \text{if} \quad x < 0 \end{cases}$$

Similarly we define $\tilde{f}_L$. Then

$$\mathbb{P}(Q_R(\mathbf{x}_1) = Q_R(\mathbf{x}_2)) \quad = \quad \frac{1}{2}\int_{-\infty}^\infty \tilde{g}_k(l) \tilde{f}_L(l)\,\mathrm{d}l$$

However, we can see that

$$\tilde{g}_k(l) \quad = \quad \text{train}_k * \text{tri}\,(l)$$

where

$$\text{train}_k(l) = \sum_{i=-\infty}^{\infty} \delta(l - ik) \qquad \text{and} \qquad \text{tri}(l) = \begin{cases} 1 - l & \text{if } |l| \leq 1 \\ 0 & \text{if } |l| > 1 \end{cases}$$

Additionally, using the Parseval's theorem we have

$$\mathbb{P}(Q_R(\mathbf{x}_1) = Q_R(\mathbf{x}_2)) \quad = \quad \frac{1}{2}\int_{-\infty}^\infty \hat{\tilde{g}}_k(\xi) \hat{\tilde{f}}_L(\xi) d\xi$$

where $\hat{\tilde{g}}_k$ and $\hat{\tilde{f}}_L$ are the Fourier transform of $\tilde{g}_k$ and $\tilde{f}_L$ respectively. But, using the definition of $\tilde{g}_k$ we have

$$\begin{aligned} \hat{\tilde{g}}_k(\xi) \quad &= \quad \hat{\text{tri}}(\xi) \cdot \hat{\text{train}}_k(\xi) \\ &= \quad \text{sinc}^2(\xi) \cdot \left(\frac{1}{k} \sum_{i=-\infty}^{\infty} \delta\left(\xi - \frac{i}{k}\right)\right) \end{aligned}$$

On the other hand, since $\tilde{f}_L$ is a Gaussian, it is easy to see

$$\hat{\tilde{f}}_L(\xi) \quad = \quad 2\exp\big(-2\,(\pi\sigma\|\mathbf{x}_1 - \mathbf{x}_2\|\xi)^2\,\big)$$

Therefore

$$\begin{aligned} \mathbb{P}\left(Q_R(\mathbf{x}_1) = Q_R(\mathbf{x}_2)\right) \quad &= \quad \int_{-\infty}^\infty \frac{1}{k}\text{sinc}^2(\xi) \sum_{i=-\infty}^{\infty} \delta\left(\xi - \frac{i}{k}\right) \exp\left(-2\left(\pi\sigma\|\mathbf{x}_1 - \mathbf{x}_2\|\xi\right)^2\right) \mathrm{d}\xi \\ &= \quad \frac{1}{k}\sum_{i=-\infty}^{\infty} \text{sinc}^2\left(\frac{i}{k}\right) \exp\left(-2\left(\frac{\pi\sigma\|\mathbf{x}_1 - \mathbf{x}_2\| i}{k}\right)^2\right) \\ &= \quad \frac{1}{k}\left(1 + 2\sum_{i=1}^{\infty}\text{sinc}^2\left(\frac{i}{k}\right) \exp\left(-2\left(\frac{\pi\sigma i\|\mathbf{x}_1 - \mathbf{x}_2\|}{k}\right)^2\right)\right) \end{aligned}$$

□

B.8 Proof Theorem 5.2.2

Theorem 5.2.2.

If $\mathbf{x}_1$ and $\mathbf{x}_2 \in \mathbb{R}^N$ and $R = (k, A, U)$ a random key to compose a σ-Gaussian Modular Hash, then

$$\frac{1}{k} \quad \leq \quad \mathbb{P}\Big(Q_R(\mathbf{x}_1)_j = Q_R(\mathbf{x}_2)_j\Big) \quad \leq \quad \frac{1}{k}\left(1 + \frac{k^2}{3}\exp\left(-2\left(\frac{\pi\sigma\|\mathbf{x}_1 - \mathbf{x}_2\|}{k}\right)^2\right)\right) \tag{B.11}$$

Therefore, if $\|\mathbf{x}_1 - \mathbf{x}_2\| \to \infty$, then

$$\mathbb{P}\Big(Q_R(\mathbf{x}_1)_j = Q_R(\mathbf{x}_2)_j\Big) \quad \to \quad \frac{1}{k} \tag{B.12}$$

Proof

The left-hand side inequality is trivial noting that in the series there are positive elements only.

For the second inequality, we have to notice that $\exp(-x)$ is decreasing, then $\forall i \in \mathbb{N}$

$$\exp\left(-2\left(\frac{\pi\sigma\|\mathbf{x}_1 - \mathbf{x}_2\|}{k}\right)^2\right) \quad \geq \quad \exp\left(-2\left(\frac{\pi\sigma\, i\,\|\mathbf{x}_1 - \mathbf{x}_2\|}{k}\right)^2\right)$$

Besides, $\operatorname{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$, and if $x > 0$

$$-\frac{1}{\pi x} \quad \leq \quad \operatorname{sinc}(x) \quad \leq \quad \frac{1}{\pi x}$$

then

$$\operatorname{sinc}^2(x) \quad \leq \quad \frac{1}{\pi^2 x^2}$$

Therefore

$$\operatorname{sinc}^2\left(\frac{i}{k}\right) \quad \leq \quad \frac{k^2}{\pi^2 i^2}$$

Then,

$$\begin{aligned}
\sum_{i=1}^{\infty}\operatorname{sinc}^2\left(\frac{i}{k}\right)\exp\left(-2\left(\frac{\pi\sigma\|\mathbf{x}_1 - \mathbf{x}_2\| i}{k}\right)^2\right) \quad &\leq \quad \sum_{i=1}^{\infty}\operatorname{sinc}^2\left(\frac{i}{k}\right)\exp\left(-2\left(\frac{\pi\sigma\|\mathbf{x}_1 - \mathbf{x}_2\|}{k}\right)^2\right) \\
&\leq \quad \exp\left(-2\left(\frac{\pi\sigma\|\mathbf{x}_1 - \mathbf{x}_2\|}{k}\right)^2\right)\sum_{i=1}^{\infty}\frac{k^2}{\pi^2 i^2} \\
&= \quad \exp\left(-2\left(\frac{\pi\sigma\|\mathbf{x}_1 - \mathbf{x}_2\|}{k}\right)^2\right)\frac{k^2}{\pi^2}\sum_{i=1}^{\infty}\frac{1}{i^2} \\
&= \quad \exp\left(-2\left(\frac{\pi\sigma\|\mathbf{x}_1 - \mathbf{x}_2\|}{k}\right)^2\right)\frac{k^2}{\pi^2}\cdot\frac{\pi^2}{6}
\end{aligned}$$

Thus, replacing this inequality we get the result.

□

B.9 Proof Theorem 5.3.1

Theorem 5.3.1.

If $R = (k, A, U)$ is a random key to compose a σ-Gaussian Modular Hash, then, for any pair of points $\mathbf{x}_1\,,\mathbf{x}_2 \in \mathbb{R}^N$, $\forall k$ even, we have:

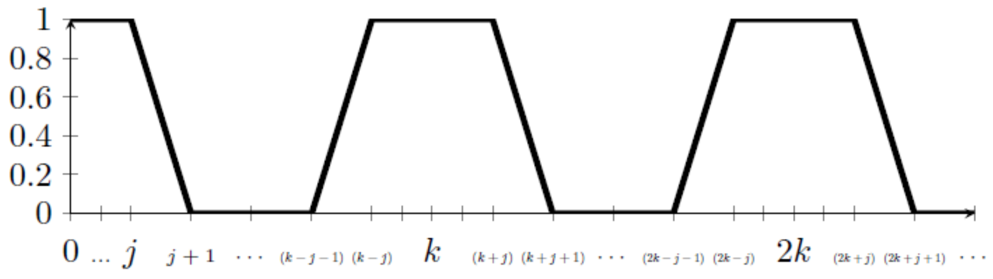
$$\mathbb{E}\left[d_{mod}\left(Q_R(\mathbf{x}_1)\ ,\ Q_R(\mathbf{x}_2)\right)\right] \quad = \quad \frac{k}{4} - \frac{2k}{\pi^2}\sum_{j=1}^{\infty}\frac{1}{(2j-1)^2}\exp\left(-2\left(\frac{\pi\,\sigma\,(2j-1)\,\|\mathbf{x}_1-\mathbf{x}_2\|}{k}\right)^2\right)$$

Proof

We can consider, with loss of generality, that $M = 1$. We have that

$$\mathbb{P}\left(d_{mod}(Q_R(\mathbf{x}_1), Q_R(\mathbf{x}_2)) \leq j \;\middle|\; \|A(\mathbf{x}_1-\mathbf{x}_2)\|\right)$$

is given by the function g_k depending on $L = \|A(\mathbf{x}_1 - \mathbf{x}_2)\|$, described in the following figure



Besides, we know that the density function of L is

$$f_L(l) \quad = \quad \sqrt{\frac{2}{\pi}}\frac{1}{\sigma\|\mathbf{x}_1-\mathbf{x}_2\|}\exp\left(-\frac{l^2}{2\sigma^2\|\mathbf{x}_1-\mathbf{x}_2\|^2}\right)$$

Then

$$\mathbb{P}\left(d_{mod}(Q_R(\mathbf{x}_1)\,,\,Q_R(\mathbf{x}_2)) \leq j\right) \quad = \quad \int_0^{\infty} g_k(l) f_L(l)\,\mathrm{d}l$$

But, since both g_k and f_L are positive, we can extend these functions over the entire real line, defining $\tilde{g}_k$ as

$$\tilde{g}_k(x) = \begin{cases} g_k(x) & \text{if} \quad x \geq 0 \\ g_k(-x) & \text{if} \quad x < 0 \end{cases}$$

Similarly we define $\tilde{f}_L$. Then

$$\mathbb{P}\left(d_{mod}(Q_R(\mathbf{x}_1)\,,\,Q_R(\mathbf{x}_2)) \leq j\right) \quad = \quad \frac{1}{2}\int_{-\infty}^{\infty} \tilde{g}_k(l)\tilde{f}_L(l)\,\mathrm{d}l$$

However, we can see that

$$\tilde{g}_k(l) = \text{train}_k * h\,(l)$$

where

$$\text{train}_k(l) = \sum_{i=-\infty}^{\infty} \delta(l - ik)$$

and

$$h(x) = \begin{cases} x + j + 1 & \text{if } x \in [-j-1, -j] \\ \\ 1 & \text{if } x \in [-j, j] \\ \\ -x + j + 1 & \text{if } x \in [j, j+1] \\ \\ 0 & \text{otherwise} \end{cases}$$

Additionally, using the Parseval's theorem we have

$$\mathbb{P}\big(d_{mod}(Q_R(\mathbf{x}_1), Q_R(\mathbf{x}_2)) \le j\big) \quad = \quad \frac{1}{2}\int_{-\infty}^{\infty} \hat{\tilde{g}}_k(\xi)\hat{\tilde{f}}_L(\xi)\,\mathrm{d}\xi$$

where $\hat{\tilde{g}}_k$ and $\hat{\tilde{f}}_L$ are the Fourier transform of $\tilde{g}_k$ and $\tilde{f}_L$ respectively. But, using the definition of $\tilde{g}_k$ we have

$$\begin{aligned} \hat{\tilde{g}}_k(\xi) &= \hat{h}(\xi) \cdot \widehat{\text{train}}_k(\xi) \\ &= \frac{1}{2\pi^2\xi^2}\left(\cos(2\pi\xi j) - \cos(2\pi\xi(j+1))\right) \cdot \left(\frac{1}{k}\sum_{i=-\infty}^{\infty} \delta\left(\xi - \frac{i}{k}\right)\right) \end{aligned}$$

because

$$\hat{h}(\xi) \quad = \quad \frac{1}{2\pi^2\xi^2}\left(\cos(2\pi\xi j) - \cos(2\pi\xi(j+1))\right)$$

On the other hand, since $\tilde{f}_L$ is a Gaussian, it is easy to see

$$\hat{\tilde{f}}_L(\xi) \quad = \quad 2\exp\left(-2\left(\pi\sigma\|\mathbf{x}_1 - \mathbf{x}_2\|\xi\right)^2\right)$$

Therefore

$$\begin{aligned} \mathbb{P}\big(d_{mod}(Q_R(\mathbf{x}_1)\,,\,Q_R(\mathbf{x}_2)) \le j\big) &= \int_{-\infty}^{\infty} \frac{1}{k}\frac{1}{2\pi^2\xi^2}\left(\cos(2\pi\xi j) - \cos(2\pi\xi(j+1))\right) \\ &\quad \cdot \sum_{i=-\infty}^{\infty} \delta\left(\xi - \frac{i}{k}\right)\exp\left(-2\left(\pi\sigma\|\mathbf{x}_1 - \mathbf{x}_2\|\xi\right)^2\right)\,\mathrm{d}\xi \\ &= \frac{1}{k}\sum_{i=-\infty}^{\infty} \frac{1}{2\left(\frac{\pi i}{k}\right)^2}\left[\cos\left(2\pi j\frac{i}{k}\right) - \cos\left(2\pi(j+1)\frac{i}{k}\right)\right] e^{-2\left(\frac{\pi\sigma\|\mathbf{x}_1-\mathbf{x}_2\| i}{k}\right)^2} \\ &= \frac{k}{\pi^2}\sum_{i=1}^{\infty} \frac{\left[\cos\left(2\pi j\frac{i}{k}\right) - \cos\left(2\pi(j+1)\frac{i}{k}\right)\right]}{i^2} e^{-2\left(\frac{\pi\sigma\, i\|\mathbf{x}_1-\mathbf{x}_2\|}{k}\right)^2} + \frac{2j+1}{k} \end{aligned}$$

Finally, noting that if k is even, we have

$$0 \quad \leq \quad d_{mod}(Q_R(\mathbf{x}_1)\,,\,Q_R(\mathbf{x}_2)) \quad \leq \quad \frac{k}{2}$$

and

$$\mathbb{E}\left[d_{mod}(Q_R(\mathbf{x}_1)\,,\,Q_R(\mathbf{x}_2))\right] \quad = \quad \sum_{i=0}^{k/2} 1 - \mathbb{P}\big(d_{mod}(Q_R(\mathbf{x}_1)\,,\,Q_R(\mathbf{x}_2)) \leq j\big)$$

we can prove that

$$\mathbb{E}\Big[d_{mod}(Q_R(\mathbf{x}_1)\,,\,Q_R(\mathbf{x}_2))\Big] \quad = \quad \frac{k}{4} - \frac{2k}{\pi^2}\sum_{i=1}^{\infty}\frac{1}{(2i-1)^2}e^{-2\left(\frac{\pi\|\mathbf{x}_1-\mathbf{x}_2\|(2i-1)}{\delta k}\right)^2}$$

$\square$

B.10 Proof Theorem 5.3.2

Theorem 5.3.2.

Setting $\sigma = \sqrt{\frac{\pi}{2}}$, and being $R = (k, A, U)$ the random key that composes a $\sigma-$Gaussian Modular Hash, and defining the error

$$\epsilon(\|\mathbf{x}_1 - \mathbf{x}_2\|\,,\,k) \quad := \quad \Big|\mathbb{E}\Big[d_{mod}\left(Q_R(\mathbf{x}_1)\,,\,Q_R(\mathbf{x}_2)\right)\Big] - \|\mathbf{x}_1 - \mathbf{x}_2\|\Big| \tag{B.13}$$

we have that

$$\epsilon(\|\mathbf{x}_1 - \mathbf{x}_2\|\,,\,k) \quad \leq \quad F(\|\mathbf{x}_1 - \mathbf{x}_2\|, k) \tag{B.14}$$

where

$$F(t,k) \quad = \quad t \cdot \exp\left(-\frac{k^2}{4\pi t^2}\right)$$

Proof

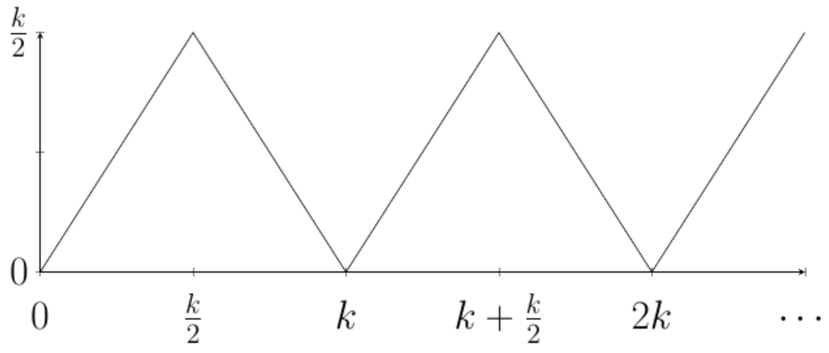
Using the same scheme as in the proof of theorem 5.3.1, we can see that

$$\mathbb{E}\left[d_{mod}\left(Q_R(\mathbf{x}_1)\,,\,Q_R(\mathbf{x}_2)\right)\right] \quad = \quad \int_0^{\infty} f_L(u)w(u), \mathrm{d}u$$

where f_L is given by

$$f_L(u) \quad = \quad \sqrt{\frac{2}{\pi}}\frac{1}{\sigma\|\mathbf{x}_1 - \mathbf{x}_2\|}\exp\left(-\frac{u^2}{2\sigma^2\|\mathbf{x}_1 - \mathbf{x}_2\|^2}\right)$$

and w by the function shown in the following plot



It is easy to see that

$$w(u) \leq u \qquad \forall u$$

then

$$\begin{aligned}\mathbb{E}\Big[d_{mod}\left(Q_R(\mathbf{x}_1)\,,\,Q_R(\mathbf{x}_2)\right)\Big] &\leq \int_0^\infty f_L(u)u\,\mathrm{d}u\\ &= \sqrt{\frac{2}{\pi}}\,\sigma\,\|\mathbf{x}_1-\mathbf{x}_2\|\end{aligned}$$

Besides, noting that $d_{mod}\big(Q_R(\mathbf{x}_1)\,,\,Q_R(\mathbf{x}_2)\big) \geq 0$, we can write

$$\left|\mathbb{E}\Big[d_{mod}\left(Q_R(\mathbf{x}_1)\,,\,Q_R(\mathbf{x}_2)\right)\Big] - \sqrt{\frac{2}{\pi}}\cdot\sigma\|\mathbf{x}_1-\mathbf{x}_2\|\right| = \left|\int_0^\infty f_L(u)\cdot(w(u)-u)\,\mathrm{d}u\right|$$

but

$$\left|\int_0^\infty f_L(u)\cdot(w(u)-u)\,\mathrm{d}u\right| \leq \int_{\frac{k}{2}}^\infty f_L(u)\cdot u\,\mathrm{d}u$$

and we have

$$\int_{\frac{k}{2}}^\infty f_L(u)\cdot u\,\mathrm{d}u = \sqrt{\frac{2}{\pi}}\sigma\|\mathbf{x}_1-\mathbf{x}_2\|\,\exp\left(-\frac{k^2}{8\sigma^2\|\mathbf{x}_1-\mathbf{x}_2\|^2}\right)$$

Hence, setting $\sigma=\sqrt{\frac{\pi}{2}}$ we have

$$\left|\mathbb{E}\Big[d_{mod}\left(Q_R(\mathbf{x}_1)\,,\,Q_R(\mathbf{x}_2)\right)\Big] - \|\mathbf{x}_1-\mathbf{x}_2\|\right| \leq \|\mathbf{x}_1-\mathbf{x}_2\|\cdot\exp\left(-\frac{k^2}{4\pi\|\mathbf{x}_1-\mathbf{x}_2\|^2}\right)$$

And defining

$$F(\|\mathbf{x}_1-\mathbf{x}_2\|) = \|\mathbf{x}_1-\mathbf{x}_2\|\cdot\exp\left(-\frac{k^2}{4\pi\|\mathbf{x}_1-\mathbf{x}_2\|^2}\right)$$

we have the result.

□

Bibliography

[1] Jiménez, Abelino, et al. *"Secure modular hashing."* 2015 IEEE international workshop on information forensics and security (WIFS). IEEE, 2015. 5, 72

[2] Jiménez, Abelino, and Bhiksha Raj. *"A two factor transformation for speaker verification through l1 comparison."* 2017 IEEE Workshop on Information Forensics and Security (WIFS). IEEE, 2017. 5, 6

[3] Jiménez, Abelino, and Bhiksha Raj. *"Privacy preserving Distance computation using somewhat-trusted third parties."* 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2017. 5, 6

[4] Jiménez, Abelino, Benjamín Elizalde, and Bhiksha Raj. *"Acoustic Scene Classification Using Discrete Random Hashing for Laplacian Kernel Machines"*. 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2018. 5

[5] Jiménez, Abelino, and Bhiksha Raj. *"Time Signal Classification Using Random Convolutional Features"*. 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2019. 5

[6] Nautsch, Andreas, Jiménez, Abelino, et al., *"Preserving Privacy in Speaker and Speech Characterisation"*. Computer Speech and Language. 2019. 6

[7] Jimenez, Abelino and Elizalde, Benjamin and Raj, Bhiksha. *"DCASE 2017 Task 1: Acoustic Scene Classification Using Shift-Invariant Kernels and Random Features"*. Workshop on Detection and Classification of Acoustic Scenes and Events. Munich, Germany. 2017. 56, 58

[8] M. Pathak and B. Raj, *"Privacy-Preserving Speaker Verification and Identification Using Gaussian Mixture Models"*. IEEE Transactions on Audio, Speech and Language Processing, Vol 21:2, pp. 397-406, 2013. 73

[9] M. Naehrig, K. Lauter and V. Vaikuntanathan, *"Can homomorphic encryption be practical?."* Proceedings of the 3rd ACM workshop on Cloud computing security workshop, 2011. 73

[10] P. Indyk and R. Motwani. *"Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality"*. Proceedings of 30th Symposium on Theory of Computing, 1998. 48, 49

[11] A. Andoni and P. Indyk, *"Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions"*. Communications of the ACM, Vol. 51(1), pp. 117-122, 2008. 49

[12] M. A. Pathak and B. Raj, *"Privacy Preserving Speaker Verification as Password Matching"*. Proceedings of ICASSP, 2012. 49

[13] M. Datar, N. Immorlica, P. Indyk and V.S. Mirrokni, *"Locality-Sensitive Hashing Scheme Based on p-Stable Distributions"*. Proceedings of the Symposium on Computational Geometry, 2004. 49

[14] S. Rane and P. T. Boufounos, *"Privacy-Preserving Nearest Neighbor Methods: Comparing Signals Without Revealing Them"*. IEEE Signal Processing Magazine, Vol. 30(2), pp. 18-28, 2013. 49, 73

[15] P. T. Boufounos and S. Rane, *"Secure Binary Embeddings for Privacy Preserving Nearest Neighbors"*. Proceedings of WIFS, 2011. 39, 49

[16] J. P. Campbell, *"Testing with the YOHO CD-ROM Voice Verification Corpus"*. *Proc. ICASSP*, Detroit, Michigan, USA, May 1995. 67

[17] W. M. Campbell, D. E. Sturim, D. A. Reynolds, *"Support vector machines using GMM supervectors for speaker verification"*. IEEE Signal Processing Letters, Vol. 13:5, pp. 308–311, 2006. 62

[18] B. Davis, and P. Mermelstein, *"Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences"*. IEEE Trans. ASSP, Vol. 28(4), pp. 357–366, 1980. 62

[19] Yuan, Guo-Xun and Ho, Chia-Hua and Lin, Chih-Jen, *"Recent advances of large-scale linear classification"*. Proceedings of the IEEE, 2012. 58

[20] Petros T. Boufounos and Hassan Mansour, *"Universal embeddings for kernel machine classification"*. SampTA, 2015. 58

[21] Rahimi, Ali and Recht, Benjamin, *"Random features for large-scale kernel machines"*. Advances in neural information processing systems, 2008. 58, 88, 91

[22] Aman Sinha and John C. Duchi, *"Learning Kernels with Random Features"*, NIPS, 2016. 59

[23] Lu, Zhiyun and May, Avner and Liu, Kuan and Garakani, Alireza Bagheri and Guo, Dong and Bellet, Aurélien and Fan, Linxi and Collins, Michael and Kingsbury, Brian and Picheny, Michael and others,

"How to scale up kernel methods to be as good as deep neural nets". arXiv preprint arXiv:1411.4000, 2014. 59

[24] Mikami, Tsuyoshi and Kojima, Yohichiro and Yonezawa, Kazuya and Yamamoto, Masahito and Furukawa, Masashi, *"An SVM-based classification of oral and nasal snoring sounds with Kullback-Leibler kernel"*. SICE Annual Conference (SICE), 2012 Proceedings of, 2012. 59

[25] Tsuyoshi Mikami and Yohichiro Kojima and Kazuya Yonezawa and Masahito Yamamoto and Masashi Furukawa, *"Spectral classification of oral and nasal snoring sounds using a support vector machine"*. Journal of Advanced Computational Intelligence and Intelligent Informatics, 2013. 59

[26] Jiyan Yang and Vikas Sindhwani and Quanfu Fan and Haim Avron and Michael Mahoney, *"Random Laplace Feature Maps for Semigroup Kernels on Histograms"*. CVPR, 2014. 59

[27] Goh, King-Shy and Chang, Edward and Cheng, Kwang-Ting, *"Support vector machine pairwise classifiers with error reduction for image classification"*, Proceedings of the 2001 ACM workshops on Multimedia: multimedia information retrieval, 2001. 59

[28] Fuxin Li and Catalin Ionescu and Cristian Sminchisescu, *"Random Fourier approximations for skewed multiplicative histogram kernels"*. DAGM 2010: Pattern Recognition pp 262-271, 2010. 58

[29] Andrea Vedaldi and Andrew Zisserman, *"Efficient additive kernels via explicit feature maps"*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 3(34):480âĂŞ492, 2012. 58, 88

[30] Y. Wang, P. Ishwar and S. Rane, *"Information-theoretically secure three-party computation with one active adversary"*. 2012. [Online]. Available: http://arxiv.org/abs/1206.2669 73

[31] J. Bringer, H. Chabanne and A. Patey, *"SHADE: Secure HAmming DistancE Computation from Oblivious Transfer"*. Financial Cryptography and Data Security Volume 7862 of the series Lecture Notes in Computer Science pp 164-176. 2013. 77

[32] V. Patel, N. Ratha and R. Chellappa. *"Cancelable Biometrics: A review"*. IEEE Signal Processing Magazine, Vol 32(5), pp. 54-65, 2015. 63

[33] C. Rathgeb and C. Busch. *"Multi-biometrics Template Protection : Issues and Challenges"*, INTECH, 2012 63

[34] A. Teoh, D. N. C. Ling, and A. Goh, *"Biohashing: Two factor authentication featuring fingerprint data and tokenised random number"*. Pattern Recogn., vol. 37, no. 11, pp. 2245 - 2255, 2004. 64

[35] M. Savvides, B. Kumar and P. Khosla. *"Cancelable biometric filters for face recognition"*. Proceedings of the 17th International Conference on Pattern Recognition. 2004. 64

[36] N. Ratha, S. Chikkerur, J. Connell, and R. Bolle, *"Generating cancelable fingerprint templates"*. IEEE Trans. Pattern Anal. Mach. Intell., vol. 29, no. 4, pp. 561-572, Apr. 2007.

[37] B. Yang, D. Hartung, K. Simoens, and C. Busch, *"Dynamic random projection for biometric template protection"*. in Proc. IEEE Int. Conf. Biometrics: Theory Applications and Systems, Sept. 2010, pp. 1-7.

[38] P. Das, K. Karthik, and B. C. Garai, *"A robust alignment-free fingerprint hashing algorithm based on minimum distance graphs"*. Pattern Recogn., vol. 45, no. 9, pp. 3373-3388, 2012. 62

[39] J. Hmmerle-Uhl, E. Pschernig, and A. Uhl, *"Cancelable iris biometrics using block re-mapping and image warping"*. in Information Security (Lecture Notes in Computer Science, vol. 5735), P. Samarati, M. Yung, F. Martinelli, and C. Ardagna, Eds. Berlin, Germany: Springer, 2009, pp. 135-142.

[40] J. K. Pillai, V. M. Patel, R. Chellappa, and N. K. Ratha, *"Sectored random projections for cancelable iris biometrics"*. in Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing, Mar. 2010, pp. 1838-1841. 62

[41] A. Teoh, A. Goh, and D. Ngo, *"Random multispace quantization as an analytic mechanism for biohashing of biometric and random identity inputs"*. IEEE Trans. Pattern Anal. Mach. Intell., vol. 28, no. 12, pp. 1892-1901, Dec. 2006.

[42] M. Jeong, C. Lee, J. Kim, J. Choi, K. Toh, and J. Kim. *"Changeable biometrics for appearance based face recognition"*. In Proc. of Biometric Consortium Conf., 2006 Biometrics Symposium, pages 1-5, 2006. 63

[43] H. Zhu, Q. He and Y. Li. *"A two-step hybrid approach for voiceprint-biometric template protection"*. International Conference on Machine Learning and Cybernetics. 2012.

[44] A. Chikkerur, N. Ratha, J. Connell and R. Bolle, *"Generating Registration-free Cancelable Fingerprint Templates"*. Proceedings of BTAS, 2008. 62, 63, 64

[45] M.S. Pinsker. *"On Estimation of Information via Variation"*. Probl. Peredachi Inf., 2005, vol. 41, no. 2, pp. 3âĂŞ8 [Probl. Inf. Trans. (Engl. Transl.), 2005, vol. 41, no. 2, pp. 71-75].

[46] Gentry, Craig. *"Fully homomorphic encryption using ideal lattices"*. Stoc. Vol. 9. No. 2009. 2009. 8

[47] Paillier, Pascal. *"Public-key cryptosystems based on composite degree residuosity classes"*. International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin, Heidelberg, 1999. 8

[48] ElGamal, Taher. *"A public key cryptosystem and a signature scheme based on discrete logarithms"*. IEEE transactions on information theory 31.4 (1985): 469-472. 8

[49] Hoffstein, Jeffrey, Jill Pipher, and Joseph H. Silverman. *"NTRU: A ring-based public key cryptosystem"*. International Algorithmic Number Theory Symposium. Springer, Berlin, Heidelberg, 1998. 8

[50] Acar, Abbas, et al. *"A survey on homomorphic encryption schemes: Theory and implementation"*. ACM Computing Surveys (CSUR) 51.4 (2018): 79. 8

[51] Dwork, Cynthia. *"Differential privacy"*. Encyclopedia of Cryptography and Security (2011): 338-340. 13

[52] Dwork, Cynthia, and Aaron Roth. *"The algorithmic foundations of differential privacy"*. Foundations and Trends in Theoretical Computer Science 9.3-4 (2014): 211-407. 14, 15

[53] Wang, Yuan, et al. *"Towards efficient privacy-preserving encrypted image search in cloud computing"*. Soft Computing 23.6 (2019): 2101-2112 71

[54] Rane, Shantanu, and Petros T. Boufounos. *"Privacy-preserving nearest neighbor methods: Comparing signals without revealing them"*. IEEE Signal Processing Magazine 30.2 (2013): 18-28. 10

[55] Yao, Andrew Chi-Chih. *"Protocols for secure computations"*. FOCS. Vol. 82. 1982. 11

[56] Cover, Thomas M., and Joy A. Thomas. *"Elements of information theory"*. John Wiley & Sons, 2012. 19

[57] Massey, James L. *"An introduction to contemporary cryptology"*. Proceedings of the IEEE 76.5 (1988): 533-549. 20

[58] Maurer, Ueli, and Stefan Wolf. *"Information-theoretic key agreement: From weak to strong secrecy for free"*. International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin, Heidelberg, 2000. 21

[59] CsiszÃąr, Imre. *"Almost independence and secrecy capacity"*. Problemy Peredachi Informatsii 32.1 (1996): 48-57. 21

[60] Arimoto, Suguru. *"An algorithm for computing the capacity of arbitrary discrete memoryless channels"*. IEEE Transactions on Information Theory 18.1 (1972): 14-20. 21

[61] Blahut, Richard. *"Computation of channel capacity and rate-distortion functions"*. IEEE transactions on Information Theory 18.4 (1972): 460-473. 21

[62] Cynthia, Dwork. *"Differential privacy"*. Automata, languages and programming (2006): 1-12. 14

[63] Pathak, Manas A., and Bhiksha Raj. *"Large margin Gaussian mixture models with differential privacy"*. IEEE Transactions on dependable and secure computing 9.4 (2012): 463-469. 15

[64] Pathak, Manas, Shantanu Rane, and Bhiksha Raj. *"Multiparty differential privacy via aggregation of locally trained classifiers"*. Advances in Neural Information Processing Systems. 2010. 15

[65] Shokri, Reza, and Vitaly Shmatikov. *"Privacy-preserving deep learning"*. Proceedings of the 22nd ACM SIGSAC conference on computer and communications security. ACM, 2015. 15

[66] Abadi, Martin, et al. *"Deep learning with differential privacy"*. Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2016. 15

[67] Phan, NhatHai, et al. *"Differential privacy preservation for deep auto-encoders: an application of human behavior prediction"*. Thirtieth AAAI Conference on Artificial Intelligence. 2016. 15

[68] Mesaros, Annamaria, et al. *"DCASE 2017 challenge setup: Tasks, datasets and baseline system"*. DCASE 2017-Workshop on Detection and Classification of Acoustic Scenes and Events. 2017. 56, 58, 59

[69] Geiger, Jurgen T., Bjorn Schuller, and Gerhard Rigoll. *"Large-scale audio feature extraction and SVM for acoustic scene classification"*. 2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics. IEEE, 2013. 56

[70] Zhang, Zixing, et al. *"Learning audio sequence representations for acoustic event classification"*. arXiv preprint arXiv:1707.08729 (2017). 56

[71] Eyben, Florian, Martin Wollmer, and Bjorn Schuller. *"Opensmile: the munich versatile and fast open-source audio feature extractor"*. Proceedings of the 18th ACM international conference on Multimedia. ACM, 2010. 56

[72] Bunn, Paul, and Rafail Ostrovsky. *"Secure two-party k-means clustering"*. Proceedings of the 14th ACM conference on Computer and communications security. ACM, 2007. 12, 13

[73] Aggarwal, Charu C., and S. Yu Philip, eds. *"Privacy-preserving data mining: models and algorithms"*. Springer Science & Business Media, 2008. 12

[74] Riazi, M. Sadegh, et al. *"Chameleon: A hybrid secure computation framework for machine learning applications"*. Proceedings of the 2018 on Asia Conference on Computer and Communications Security. ACM, 2018. 12

[75] Hastings, Marcella, et al. *"SoK: General Purpose Compilers for Secure Multi-Party Computation"*. IEEE, 2019. 12

[76] Regev, Oded. *"The learning with errors problem"*. Invited survey in CCC 7 (2010). 30

[77] Gentry, Craig, Amit Sahai, and Brent Waters. *"Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based"*. Annual Cryptology Conference. Springer, Berlin, Heidelberg, 2013. 30

[78] Peter Sykacek and Stephen J. Roberts. *"Bayesian time series classification"*. In Advances in Neural Information Processing Systems, pp. 937-944. 2002. 86

[79] Bilal Esmael, Arghad Arnaout, Rudolf K. Fruhwirth, and Gerhard Thonhauser. *"Improving time series classification using Hidden Markov Models"*. In Hybrid Intelligent Systems (HIS), 2012 12th International Conference on, pp. 502-507. IEEE, 2012 86

[80] Karan Sikka, Abhinav Dhall, and Marian Bartlett. *"Exemplar hidden markov models for classification of facial expressions in videos"*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 18-25. 2015 86

[81] Minyoung Kim. *"Semi-supervised learning of hidden conditional random fields for time-series classification"*. Neurocomputing 119 (2013): 339-349. 86

[82] Myriam Abramson. *"Sequence classification with neural conditional random fields"*. In Machine Learning and Applications (ICMLA), 2015 IEEE 14th International Conference on, pp. 799-804. IEEE, 2015. 86

[83] Bendong Zhao, Huanzhang Lu, Shangfeng Chen, Junliang Liu, and Dongya Wu. *"Convolutional neural networks for time series classification"*. Journal of Systems Engineering and Electronics 28, no. 1 (2017): 162-169. 86

[84] Nima Hatami, Yann Gavet, and Johan Debayle. *"Classification of time-series images using deep convolutional neural networks"*. In Tenth International Conference on Machine Vision (ICMV 2017), vol. 10696, p. 106960Y. International Society for Optics and Photonics, 2018. 86

[85] Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Shun Chen. *"Lstm fully convolutional networks for time series classification"*. IEEE Access 6 (2018): 1662-1669. 86

[86] Yang Guo, Zhenyu Wu, and Yang Ji. *"A Hybrid Deep Representation Learning Model for Time Series Classification and Prediction"*. In Big Data Computing and Communications (BIGCOM), 2017 3rd International Conference on, pp. 226-231. IEEE, 2017. 86

[87] Stefan Ruping. *"Svm kernels for time series analysis"*. In Klinkenberg, R., Ruping, S., Fick, A., Henze, N., Herzog, C., Molitor, R., and Schroder, O., editors, LLWA 01 - Tagungsband der GI-Workshop-Woche Lernen - Lehren - Wissen - Adaptivitat, Forschungsberichte des Fachbereichs Informatik der Universitat Dortmund, pages 43-50, Dortmund, Germany. 2001. 86

[88] Ginés Rubio, Héctor Pomares, Luis J. Herrera and Ignacio Rojas. *"Kernel Methods Applied to Time Series Forecasting"*. Conference Proceedings in Computational and Ambient Intelligence. 2007. 86

[89] Marco Cuturi and Arnaud Doucet. *"Autoregressive Kernels For Time Series"*. arXiv:1101.0673. 2011. 86

[90] Karl Mikalsen, Filippo Bianchi, Cristina Soguero-Ruiz and Robert Jenssen. *"Time series cluster kernel for learning similarities between multivariate time series with missing data"*. Pattern Recognition 76, 569-581. 2018. 86

[91] Huanhuan Chen, Fengzhen Tang, Peter Tino, and Xin Yao. *"Model-based kernel for efficient time series analysis"*. In Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '13). 2013. 86

[92] Walter Rudin. *"Fourier analysis on groups"*. Courier Dover Publications, 2017. 87

[93] Alexander M. Akhmetshin and I. V. Lyuboshenko. *"The reconstruction of signals and images from the noisy Fourier transform phase by means of the generalized difference principle"*. In Pattern Recognition, 1996., Proceedings of the 13th International Conference on, vol. 2, pp. 370-375. IEEE, 1996. 87

[94] Hiroshi Shimodaira, Ken-ichi Noma, Mitsuru Nakai, and Shigeki Sagayama. *"Dynamic time-alignment kernel in support vector machine"*. In Advances in neural information processing systems, pp. 921-928. 2002 87

[95] Marco Cuturi, Jean-Philippe Vert, Oystein Birkenes and Tomoko Matsui. *"A Kernel for Time Series Based on Global Alignments"*. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '07). 2007. 87

[96] Alan V. Oppenheim. *"Discrete-time signal processing"*. Pearson Education India, 1999. 87, 88

[97] Gabriel Wachman, Roni Khardon, Pavlos Protopapas, and Charles R. Alcock. *"Kernels for Periodic Time Series Arising in Astronomy"*. In Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part II (ECML PKDD '09), 2009. 87

[98] Charles Van Loan. *"Computational frameworks for the fast Fourier transform"*. Vol. 10. Siam, 1992. 88

[99] Yang, Jiyan, Vikas Sindhwani, Quanfu Fan, Haim Avron, and Michael W. Mahoney. *"Random laplace feature maps for semigroup kernels on histograms"*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 971-978. 2014 88

[100] Avrim Blum. *"Random projection, margins, kernels, and feature-selection"*. In Proceedings of the 2005 international conference on Subspace, Latent Structure and Feature Selection (SLSFS'05). 2005. 88, 93

[101] Purushottam Kar and Harish Karnick. *"Random feature maps for dot product kernels"*. In Artificial Intelligence and Statistics, pp. 583-591. 2012 88

[102] Ninh Pham and Rasmus Pagh. *"Fast and scalable polynomial kernels via explicit feature maps"*. In Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 239-247. ACM, 2013. 88

[103] Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen and Gustavo Batista. *"The UCR Time Series Classification Archive"*. 2015. URL `www.cs.ucr.edu/~eamonn/time_series_data/` xi, 90, 92

[104] David Cox and Nicolas Pinto. *"Beyond simple features: A large-scale feature search approach to unconstrained face recognition"*. Automatic Face & Gesture Recognition and Workshops (FG 2011), 2011 IEEE International Conference on. IEEE, 2011. 93

[105] Riazi, M. Sadegh, et al. *"Sub-Linear Privacy-Preserving Near-Neighbor Search with Untrusted Server on Large-Scale Datasets"*. arXiv preprint arXiv:1612.01835 (2016). 35, 39

[106] Boldyreva, Alexandra, et al. *"Order-preserving symmetric encryption"*. Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin, Heidelberg, 2009. 38

[107] Boldyreva, Alexandra, Nathan Chenette, and Adam OâĂŹNeill. *"Order-preserving encryption revisited: Improved security analysis and alternative solutions"*. Annual Cryptology Conference. Springer, Berlin, Heidelberg, 2011. 38

[108] Naveed, Muhammad, Seny Kamara, and Charles V. Wright. *"Inference attacks on property-preserving encrypted databases"*. Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. ACM, 2015. 38

[109] Schaefer, Rafael F., et al., eds. *"Information Theoretic Security and Privacy of Information Systems.* Cambridge University Press, 2017. 21

[110] Jacques, Laurent, et al. *"Robust 1-bit compressive sensing via binary stable embeddings of sparse vectors"*. IEEE Transactions on Information Theory 59.4 (2013): 2082-2102. 85