Efficient Privacy Preserving Protocols for Visual Computation

Submitted in partial fulfillment of

the requirements for the degree of

Master of Science (by Research)

in

Computer Science

by

Maneesh Upmanyu

200402026

`<upmanyu@research.iiit.ac.in>`

`http://research.iiit.ac.in/∼upmanyu`

Center for Visual Information Technology

International Institute of Information Technology

Hyderabad, INDIA

March, 2010

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY

Hyderabad, India - 500032

CERTIFICATE

It is certified that the work contained in this thesis, titled "*Efficient Privacy Preserving Protocols for Visual Computation*" by Maneesh Upmanyu, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date	Advisor: Dr. C.V. Jawahar
Date	Advisor: Dr. Anoop M. Namboodiri
Date	Advisor: Dr. Kannan Srinathan

Specially dedicated to my late grandfather.

To my beloved parents and dearest siblings.

आचार्यात् पादमादत्ते पादं शिष्यः स्वमे॑धया।
पादं सब्रह्मचारिभ्यः पादम् कालक्रमे॑ण च॥

A student gets
a quarter (knowledge) from his teacher,
a quarter by his own intelligence.
A quarter from his fellow students
and a quarter in due course of time.

अ॒सतो मा स॒द्गमय।
त॒मसो मा ज्यो॒तिर्गमय।
मृत्यो॒र्माऽमृ॒तं गमय॥

From delusion lead me to truth
From darkness lead me to light
From death lead me to immortality.

Abstract

The rapid expansion of the Internet is receiving a great deal of attention world-wide. The technological developments and the increase in online communications have played a vital role in revolutionalizing the Information Age. There is a tremendous growth of online applications to manipulate the user's personal data, which has resulted in the widespread availability of the user's personal data in the digital form. This raises the issue of privacy protection against potential misuse of this data by legitimate service providers or intruders. Without proper countermeasures to thwart the attacks, security problems become a major threat and a serious impediment to further development of business applications on communication systems.

Many of the current solutions provides information security by assuming a level of trust among the parties. The leakage of the critical data to third parties is prevented by applying cryptographic primitives as a secure layer over the standard algorithm. On the other hand, privacy preservation computation is more closely related to Secure Multiparty Computation (SMC). SMC enables two parties; one with the function $f()$ and the other with the input x; to compute $f(x)$ without revealing them to each other. However, the solutions based on the general protocol of SMC requires enormous computational and communication overhead, thus limiting the practical deployment of the secure algorithms.

In this dissertation, we focus on development of 'highly-secure', 'comunication and computationally efficient' algorithms to problems with 'immediate impact' in the domain of computer vision and related areas. Security issues in computer vision primarily originates from the storage, distribution and processing of the personal data, whereas privacy concerns with tracking down of the users activity. The primary challenge is in providing the ability to perform generic computations on the visual data, while ensuring provable security. In this thesis, we propose lightweight encryptions for visual data, such that the server should be able to carry out the computations on the encrypted data and also store the stream if required, without being able to decipher the actual contents of the image. Moreover, the protocols are designed such that the interaction and the data communication among the servers is kept to a minimum.

It has been proven before that the best way to achieve secure computation on a remote server is by using the cryptographic protocol of SMC. Thus, a method that provides provable security, while allowing efficient computations without incurring either significant computation or communication overhead has remained elusive till now. We show that, for designing secure visual algorithms one can exploit certain properties such as scalability, limited range etc, inherent to visual data to break this impenetrable barrier. We study and propose secure solutions for applications such as Blind Authentication, i.e. blindly authenticating a remote-user using his biometric. Subsequently, we present a highly secure framework for carrying out visual surveillance on random looking video streams at remote servers. We then propose a simple and an efficient cloud-computing based solution using the paradigm of secret sharing to privately cluster an arbitrary partitioned data among N users. The solutions we propose are accurate, efficient and scalable and has potential to extend over to even more diverse applications.

In our first work, *blind authentication*, we propose private biometric authentication protocol which is extreamly secure under a variety of attacks and can be used with a wide variety of biometric traits. The protocol is blind in the sense that it reveals only the identity, and no additional information about the user or the biometric to the authenticating server or vice-versa. The primary advantage of the proposed approach is the ability to achieve classification of a strongly encrypted feature vector using generic classifiers such as Neural Networks and SVMs. Our proposed solution addresses the concerns of user's privacy, template protection, and trust issues. And captures the advantages of biometric authentication as well as the security of public key cryptography.

We then present an efficient, practical and highly secure framework for implementing visual surveillance on untrusted remote computers. To achieve this we demonstrate that the properties of visual data can be exploited to break the bottleneck of computational and communication overheads. The issues in practical implementation of certain algorithms including change detection, optical flow, and face detection are addressed. Our method enables distributed secure processing and storage, while retaining the ability to reconstruct the original data in case of a legal requirement. Such an architecture provides us both security as well as computation and communication efficiency.

We next extend our proposed paradigm to achieve the ability to do un-supervised learning using K-means in the encrypted domain. Traditional approaches uses primitives such as SMC or PKC, thus compromising the efficiency of the solutions and in return provide very high level of privacy which is usually an overkill in practice. We use the paradigm of *secret sharing*, which allows the data to be divided into multiple shares and processed separately at different servers. Our method shows that privacy need not be always at the cost of efficiency. Our proposed solution is not only computationally efficient but also secure independent of whether or not $P \neq NP$.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction to Privacy Preserving Methods

In the past few years, the world has become increasingly connected over the Internet. Business over the Internet has become a standard practice, leading to a tremendous growth of online applications to manipulate digital data. The advances in information processing and data sharing over the transmission media such as the Internet, has paved way to new sort of services, whereby an organization lends their processing ability and algorithms to the customers. However, many of these applications work on sensitive (personal) data of the client.

To maintain proprietary rights, the organization may not wish to make their algorithms public, but would still like to maximize their profits. On the other hand, a potential client may not be willing to reveal his private data even to the processing server. Thus, what we want is an ability for the server to obliviously compute his private function $f(x)$, without learning any meaningful information about the user's input x. An example for this would be a search engine that returns related pages without learning anything about the searched query or the related pages. Moreover, to ensure trustworthiness, computing and storage services over untrusted remote servers require security assurances against malicious attacks and faulty behavior.

To achieve this, security and cryptography are important enablers. Many of the current solutions provides information security by assuming a level of trust among the parties. The leakage of the critical data to third parties is prevented by applying cryptographic primitives as a secure layer over the standard algorithm. On the other hand, *privacy preservation computation* is more closely related to *Secure Multiparty Computation (SMC)* [143]. SMC enables two parties; one with the function $f()$ and the other with the input x; to compute $f(x)$ without revealing them to each other. In SMC based solutions every function is represented as a boolean circuit. A secure computation is then performed for each gate of the circuit. However, the solutions based on the general protocol of SMC requires enormous computational and communication overhead. The complexity of the protocol is linear in the size of the circuit, making it theoretically efficient, however for real-world applications this method is not practical and is much slower than the corresponding non secure computation [21].

In this work, we explore methods to design efficient privacy-preserving protocols with provable security and privacy guarantees.

1.1 Broad Objective

This thesis focus on development of secure computational algorithms in computer vision and related areas. Some of the specific sub-tasks associated with this work include:

- To develop **"highly-secure"** solutions; the proposed algorithms should offer a provable and semantically secure privacy rather than one on an ad hoc basis. Defining security based on formal mathematical proofs, allows us to do a theoretical analysis of the privacy achieved by our proposed solutions.

- To develop **"communication and computationally efficient"** solutions; any secure solution requires more processing over an insecure one (eg. http vs https). For a practical acceptance of a secured application, the proposed solution should keep the communication and computation overheads within acceptable limits.

- To develop solutions to problems with **immediate impact**; we study and propose solutions to enhance privacy for various popular web-based applications, wide-spread use of which are currently limited because of the privacy concerns they raise. For example, user authentication using a fingerprint over an untrusted web-server.

1.2 Problem Background

With a rapid development and acceptability of *computer vision based systems* in ones daily life, securing of the visual data has become imperative. **Security issues** in computer vision primarily originates from the *storage*, *distribution* and *processing* of the personal data, whereas **privacy concerns** with *tracking down* of the user's activity.

The ideal solution to overcoming all privacy and security concerns is to apply strong cryptographic encryptions, thus destroying any pattern that would be present in that data. Pattern recognition, which is inherent to computer vision algorithms, however exploits the strong structure (pattern) present in the data. It seems that there exists a contradiction in the objectives of these two disciplines. For example, Figure 1.1(a) shows the data pattern in the original feature space. Applying a strong encryption to this would destroy the structure, thus making any pattern recognition task on the encrypted data difficult (see Figure 1.1(c)). In order to overcome this limitation, solutions have been proposed that make a

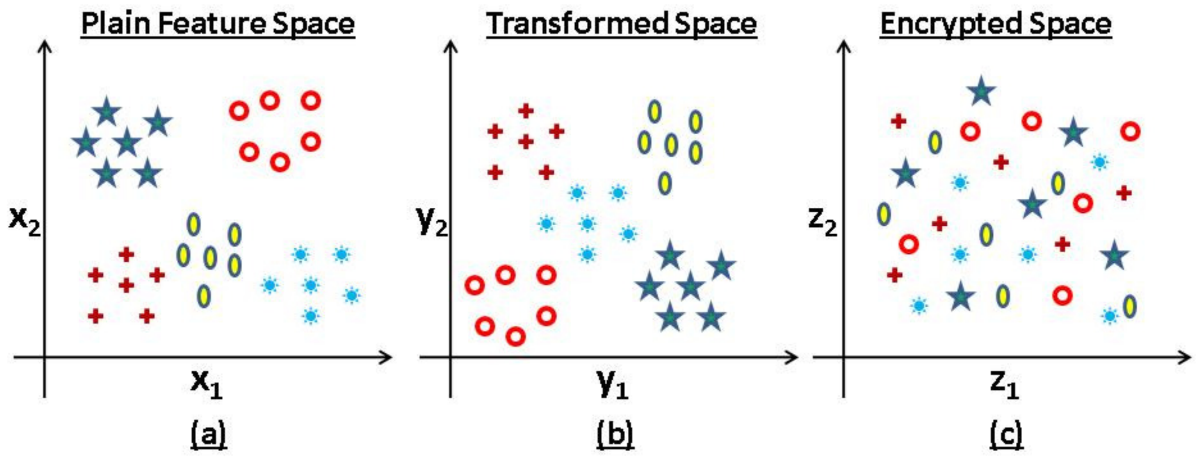


Figure 1.1 Dilemma of Privacy Vs Accuracy: Ideally we would like to adopt transformation functions such that they provide privacy as in (c), while retaining the ability to match patterns as in (a). Functions such as in (b) provide partial privacy.

compromise between privacy and accuracy. Transformation functions are applied to the data, such that they retain the pattern, while providing partial privacy (see Figure 1.1(b)).

The current methods of securing an online protocol is to apply a cryptographic layer on top of an existing processing module, thus securing the data against unauthorized third party access. However, this is often not enough to ensure the complete security of the user's privileged information.

In the world of Internet, a new service sector has emerged, where a service provider gives the user with access to a server running a particular vision algorithm. In some scenarios, the client may be reluctant to reveal the content of the image to the processing server, yet would like to fully utilize the service, while at the same time the service provider would like to protect his own interests, i.e. the algorithm from being made public. How can the service provider and the user achieve these objectives? Over the years, these questions were raised and addressed in two different scenarios as follows:

Avidan and Butman [22] raised and addressed the privacy concerns in a camera surveillance scenario, "A service provider has a proprietary face detection algorithm that he does not want to be made public, and a client wishes to use this algorithm as long as, even the service provider, does not learn anything about the data (images)."

Shashank *et al.* [119] raised and addressed the privacy issues in content based image retrieval systems (CBIR). The proposed system Private-CBIR (PCBIR) deals with the retrieval of similar content *without* revealing the content of the query image to the database.

These problems are similar to the *Secure Multiparty Computation (SMC)* [143] problem in cryptography. A straight forward solution would be to apply the secure multi-party techniques to image related algorithms. However, multi-party techniques are computationally *very* expensive [65]. And, applying the generic SMC protocols to computer vision algorithms, which works on low level data such as pixel values, would make the solution impractical and non-real time.

Through this work we address the security and privacy concerns of the visual data. We propose application specific, computationally efficient and provably secure computer vision algorithms for the encrypted domain. More specifically we address the following issues:

- **Efficacy:** Security should not be at the cost of accuracy. The classification performance of the secured implementation should be similar to that of the non-secured implementation.
- **Efficiency:** Encryption/Decryption is computationally expensive. For an application, the secure algorithms should be practical and keep the computation and communication overheads within acceptable limits.
- **Domain Knowledge:** Domain specific algorithms will be more efficient than generic solutions such as SMC. We exploit the data properties to design application specific, computationally efficient, non-interactive secure solutions.
- **Security:** Algorithms need to be provably-secure and meet futuristic requirements.

1.3 Technical Background

In this section, we discuss the general notations and definitions that are used throughout the dissertation. All chapter specific definitions will be provided at the appropriate place within the subsequent chapters.

1.3.1 What is meant by Privacy?

Privacy in the cryptographic community is defined so as to limit the information that is leaked (learned) by the distributed computations over the information that can be learned from the designated output of the computations [103]. In order to compute the information leaked, we compare the information learned from the result of the actual computations to that learned in an *"ideal"* computations involving a trusted party.

A party is said to be *trusted* if it does not deviate from a pre-defined behavior and does not attempt to cheat. In the ideal scenario, all parties send their respective data to the trusted party, who then computes the functions and sends the appropriate results to the other parties. Thus, a distributed protocol is private, if and only if, an adversary learns no additional meaningful information other than its input and the output it receives from the trusted party. However, finding a trusted third party is in general infeasible [65]. Privacy preserving protocols are introduced to address this specific problem. The objective is to design (efficient) protocols that do not reveal any information except for their designated output [17] [88].

In order to analyze the security and privacy of the system, we will formalize the notion [65] of security as follows:

Let $f : 0, 1^* \times 0, 1^* -> 0, 1^* \times 0, 1^*$ be a function. A two-party protocol is defined by a pair of probabilistic polynomial-time interactive algorithms $\pi = (\pi_A, \pi_B)$. Consider the probability space induced by the computing execution of π on input $x = (a, b)$ (induced by the independent choices of the random input r_A, r_B). Let $view^{\pi}_{A}(x)$ (resp., $view^{\pi}_{B}(x)$) denote the entire view of Alice (resp., Bob) in this execution, including her input, random input, and all messages she has received. Let $output^{\pi}_{A}(x)$ (resp., $output^{\pi}_{B}(x)$) denote Alice's (resp., Bob's) output. We say that π privately computes a function f if there exist probabilistic, polynomial-time algorithms S_A and S_B such that:

$$\{(S_A(a,\ f_A(x)),\ f_B(x))\}_{x=(a,b)\in X} \equiv \{(VIEW^{\pi}_{A}(x),\ OUTPUT^{\pi}_{B}(x))\}_{x\in X} \tag{1.1}$$

$$\{(f_A(x),\ S_B(b,\ f_B(x)))\}_{x=(a,b)\in X} \equiv \{(OUTPUT^{\pi}_{A}(x),\ VIEW^{\pi}_{B}(x)\}_{x\in X} \tag{1.2}$$

where $\equiv$ denotes computationally indistinguishability, which means that there is no probabilistic polynomial algorithm A which can distinguish the probability distribution over two random string.

1.3.2 Adversary Model

Two types of adversaries, *a)* a *semi-honest* adversary, and *b)* a *malicious* adversary are commonly considered in cryptographic community. A *semi-honest* adversary (also known as a *passive*, or *honest but curious* adversary) is a party that correctly follows the protocol specification, yet is curious and attempts to learn additional information by analyzing the messages received during the protocol execution. On the other hand, a *malicious* adversary may arbitrarily deviate from the protocol specifications. For example, consider a step in the protocol where one of the parties is required to choose a random number and broadcast it. If the party is semi-honest then we can assume that this number is indeed random. On the other hand, if the party is malicious, then he might choose the number in a sophisticated way that enables him to gain additional information.

Privacy preserving protocols are designed in order to preserve privacy even in the presence of adversarial participants that attempt to gather information about the inputs of their peers. In practice, a semi-honest adversarial model is a realistic one [103]. Moreover, a protocol that is secure against a semi-honest adversary can be transformed, using cryptographic techniques such as *zero-knowledge proofs* [65], into a protocol that is secure against malicious adversaries. Thus, we design secure protocols for the *semi-honest* case. Note that, we do not consider adversaries that change their inputs in order to gain more information about the inputs of the other parties.

The network as such is assumed to be insecure and susceptible to snooping attacks. The servers are assumed to be non-colluding in nature, that is they are independent and would not share information to extract any additional knowledge.

1.3.3 Our Security Goal

As stated in Section 1.2, we aim to strengthen the security and privacy of the visual algorithms without making a compromise on the efficiency and efficacy of the solutions. The three primary issues in designing the privacy preserving protocols are *i)* security and privacy, *ii)* efficacy, and *iii)* efficiency. Hence, we analyze the secure algorithms for the security, correctness and complexity.

- **Correctness** is measured by comparing the proposed protocol to the ideal protocol where the parties transfer their data to a trusted third party that performs the computations. If the secure protocol is identical to the ideal protocol then the protocol is declared correct.
- In **security** one needs to show what can and cannot be learned from the data exchange between the parties. One often assumes that the parties are honest but curious, meaning that they will follow the agreed upon protocol but will try to learn as much as possible from the data flow between the two parties.
- In **complexity**, one shows the computational and communication complexity of the secure algorithm. For practical applications, the overheads of the proposed solution should be minimal as compared to the ideal solution.

We use the semi-honest adversary model, that is the parties follow the protocol but they want to reveal the other party's privacy. Our goal is to design protocols for preserving the party's privacy against such adversaries during the execution of the protocol. Each party learns nothing about the others data, except the output results. Both privacy and correctness are needed to be preserved.

1.4 Thesis Outline

Private distributed protocols have been considered extensively for data mining, pioneered by Lindell and Pinkas [88], who presented a privacy-preserving data-mining algorithms for ID3 decision-tree learning. The work on privacy-preserving algorithms is motivated by the need both to protect privileged information and to enable its use for research or other purposes. The problem is a specific example of secure multi-party computations and, as such, can be solved using generic protocols. However, for the applications working on massive datasets, applying the generic protocols as such are of no practical use and therefore more efficient protocols are required.

Avidan and Butman raised and addressed the privacy concerns through their work *Blind Vision* [22], which is about applying secure multi-party techniques to vision algorithms. In blind vision there are two parties, A with a private program $\pi(I)$, and B with an input I to that program; the joint goal is to let B know the output of $\pi(I)$ whilst maintaining the privacy of π with respect to B and privacy of I with respect to A. Their solution, which is based on SMC techniques, is found to be extremely expensive in terms of communication overheads. In order to design practical protocols, considerable research effort has been made over the recent years. Modifications have been made to improve the efficiency of the solutions, such as, by restricting the usage of Yao's protocol [143] to only a few limited computations/operations. For example, Avidan *et al.* [23] speed-ed up their blind vision protocol [22] at the cost of a controlled leakage of information. Shashank *et al.* [119] on the other hand, exploited the clustered nature of image databases to improve upon the efficiency of SMC for example-based image retrieval.

In general, the SMC based protocols are found to be inefficient for any practical online application. The reason for this is mainly due to the way SMC works. In SMC, every single trusted CPU instruction is securely simulated via a corresponding network protocol. As of today, communication is the bottleneck. Factually, the round-trip time in a LAN is of the order of a few milliseconds, whereas several floating operations take no more than few nanoseconds. Thus the paradigm of SMC which converts the trusted computation into secure network protocol can not avoid a slowdown by a factor of one million, with the current technology. Moreover, communication is likely to remain a bottleneck in the foreseeable future. The communication overheads can be avoided if we can obtain a system where a server can execute a function on the encrypted data without having to decrypt it. Thus, we want encryptions such that the resulting transformation allows non-interactive computations at remote server(s).

The natural way to overcome the (communication) complexity is to design algorithms for computing in encrypted domain. Solutions with minimal distribution, using the paradigm of *encrypt-communicate-compute-decrypt* require the usage of algebraic homomorpic encryption schemes [93]. However, an efficient implementation of it is not yet known. Our method for *blindly authenticating a user using his biometric* uses multiplicative, additive homomorphism and a specific distribution of work between client and server, coupled with a novel randomization scheme to simulate the algebraic homomorphism in the encrypted domain.

As compared to SMC, we optimally reduce the communication overheads for the solution proposed. However, we find that building completely non-interactive protocols is not feasible using the additive and multiplicative homomorphism. Moreover, these schemes themselves are based on computationally expensive protocols such as public key cryptography (PKC).

In order to achieve computational efficiency the encryption should be lightweight, thus we have to avoid the usage of PKC. The inadequacy of solutions with minimal or no distribution, necessitates *non-minimal distribution*, in other words some sort of SMC. Unconditionally or information theoretic secure multi-party computations are closely related to the problem of *secret sharing* [118]. We next show that visual data has certain desirable properties that allows us to use the paradigm of secret sharing to achieve complete privacy and efficient computation of visual algorithms. We present an efficient, practical and highly secure framework for implementing *visual surveillance* on untrusted remote computers. To achieve this we demonstrate that the properties of visual data can be exploited to break the bottleneck of computational and communication overheads.

The paradigm we have proposed is accurate, efficient and scalable. We next explore as to whether the paradigm can be made generic and be used to address the privacy concerns in more related areas. The algorithms we have proposed so far dealt with pattern matching using a classifier in the encrypted domain. The natural extension to the work is to find methods for doing the classifier training itself in the encrypted domain. With these objectives in mind, we explore methods to privately do un-supervised learning using K-means in the encrypted domain. K-means clustering is one of the most widely used techniques for statistical data analysis. We show that the paradigm of secret sharing is generic and unlike the traditional methods using primitives such as SMC, the privacy need not be always at the cost of efficiency.

1.4.1 Scope of the Thesis

We now give a brief description of the problems we have addressed. A detailed description of these are given in the following chapters.

Blind Authentication using a biometric: Concerns on widespread use of biometric authentication systems are primarily centered around template security, revocability and privacy. We have proposed, for the first time, a completely blind biometric authentication protocol, which takes care of concerns on user privacy, template protection, and trust issues in biometric authentication system. The protocol is blind in the sense that it reveals only the identity, and no additional information about the user or the biometric to the authenticating server or vice-versa. Biometrics are ideal to be deployed in both high security as well as remote authentication applications. However, the assertions on security and non-repudiation are valid only if the integrity of the overall system is maintained. Blind authentication provides a mechanism to do non-repudiable authentication over an insecure network, while ensuring the privacy of the user.

Privacy Preserving Video Surveillance: We have presented an efficient, practical and highly secure framework for implementing visual surveillance on untrusted remote computers. The main contribution of the work is in introducing a paradigm shift in looking at private visual surveillance problems from the traditional SMC based approaches. This change in view allows us to have a simplified capture device, an efficient unidirectional data flow, and surveillance operations performed directly on the shattered streams. The issues in practical implementation of certain algorithms including change detection, optical flow, and face detection are addressed. The framework provides a generic setting to carry out an arbitrary vision task. This work opens up a new avenue for practical and provably secure implementations of vision algorithms, that are based on distribution of data over multiple computers.

Unsupervised learning using K-means: Clustering is one of the fundamental algorithms used in the field of data mining. The simplicity and effectiveness of the algorithm have made its usage conducive in various applications. However, the collected data may contain sensitive or private information about the customers, thus heightening the privacy concerns. We propose a novel cloud computing based solution using the paradigm of secret sharing to privately cluster an arbitrary partitioned data among N users. Our proposed solution is not only computationally efficient but also secure independent of whether or not $P \neq NP$. Our paradigm is generic and has potential to extend over to even more diverse data mining applications.

1.4.2 Blind Authentication: A Crypto-Biometric Verification Protocol

As a first step, we addressed the problem of Blind Authentication i.e., design and implementation of a Secure Crypto-Biometric Verification Protocol.

We have proposed, for the first time, a completely blind biometric authentication protocol, which takes care of concerns on user privacy, template protection, and trust issues. We propose a secure biometric authentication protocol over public networks using asymmetric encryption, which captures the advantages of biometric authentication as well as the security of public key cryptography. Biometric authentication provides a secure, non-repudiable and convenient method for identity verification, while not revealing any additional information about the user to the server or vice versa.

The encryption provides template protection, the ability to revoke enrolled templates, and alleviates the concerns on privacy in widespread use of biometrics. The proposed approach does not make any assumption on the nature of the data and is hence applicable to any biometric. Such a protocol has significant advantages over existing biometric cryptosystems, which use a biometric to secure a secret key, which in turn is used for authentication.

Biometrics are ideal to be deployed in both high security as well as remote authentication applications. However, the assertions on security and non-repudiation are valid only if the integrity of the overall system is maintained. Blind authentication provides a mechanism to do non-repudiable authentication over an insecure network, while ensuring the privacy of the user.

The primary concerns to be addressed for any biometric authentication system are:

1. **Template protection:** As a biometric do not change over time, one cannot revoke an enrolled plain biometric. Hence, critical information could be revealed if the server's biometric template database is compromised.

2. **User's privacy:** *i)* The activities of a person could be tracked, as the biometric is unique to a person, and *ii)* Certain biometrics may reveal personal information about a user (e.g., medical or food habits), in addition to identity.

3. **Trust between user and server:** In widespread use, all authenticating servers may not be competent or trustworthy to securely handle a user's plain biometric, while a remote user cannot be reliably identified without biometric information.

4. **Network security:** As the authentication is done over an insecure network, anyone snooping the network could gain access to the biometric information being transmitted.

The previous work in this area tends to build a classifier in encrypted domain, thus making a compromise in security and accuracy. A.K. Jain *et al* [73] does an extensive literature review and concludes *'a template protection scheme with provable security and acceptable recognition performance has thus far remained elusive.'*

In our proposed method, we build a classifier in the plain feature space, which allows us to maintain the performance of the biometric itself, while carrying out the authentication on data with strong encryption, which provides high security/privacy. We design protocols for simulating *Support Vector Machine and Neural network* in an encrypted domain.

Approach in brief: Let ω be the parameters of the linear classifier. The server accepts the claimed identity of a user, if $\omega \cdot x < \tau$, where τ is a threshold. As we do not want to reveal the parameter vector (ω) or the test sample (x) to the server, we need to carry out the computations in the encrypted domain. Computation of the above equation in encrypted domain would require an algebraic homomorphic encryption, which is not known to exist. Our method uses multiplicative, additive homomorphism and a specific distribution of work between client and server, coupled with a novel randomization scheme to simulate the above equation in the encrypted domain.

Several experiments are performed to evaluate the efficiency and accuracy of the proposed approach. An authentication protocol was implemented based on a client-server model that can perform verification over an insecure channel such as the Internet. Evaluation is carried out on various public domain datasets and biometric modalities to verify for efficiency and applicability. Analysis are also carried out for security/privacy and computational overhead of the proposed method.

Blind authentication addresses all of the concerns mentioned above, and provides the ability to classify any feature vector, and hence is applicable to multiple biometrics. This work opens a new direction of research to look at privacy preserving biometric authentication.

1.4.3 Efficient Privacy Preserving Video Surveillance

In this work we focus on development of secure computational algorithms for computer vision, specifically in the area of surveillance. The generic goals of the work are to develop solutions that are *secure* and *computationally efficient*, leading to practical systems.

Video Surveillance is a critical tool for tasks such as law enforcement, personal safety, traffic control, etc. This raises *privacy concerns* such as, watching you in your private moments, spying on you or even implicitly controlling some of your actions. The *challenge* of introducing privacy and security in such a practical surveillance system has been stifled by the enormous computation and communication overhead required by the solutions. The objective is to allow the general surveillance to continue, without disrupting the privacy of an individual in an efficient and cost-effective way.

Provable security/privacy can be guaranteed if the surveillance algorithms can directly run on (cryptographically strong) encrypted video streams. This ensures that the original video stream is hidden at all times and the observer learns only the final output of the surveillance algorithm.

We use the paradigm of *secret sharing* to achieve private and efficient surveillance. We exploit the properties, such as the scale invariance and a fixed range of the image data to define vision specific secret sharing scheme. Our method enables distributed secure processing and storage, while retaining the ability to reconstruct the original data in case of legal requirement. The computational requirement at the data source (camera) is very limited, enabling inexpensive monitoring equipment, and the only communication between the camera and the surveillance servers is a compact and encrypted video stream. Privacy preserving surveillance address the contrasting needs of confidentiality and utility, making the system practical.

The primary **contributions** of our work are:

- **Circumvent theoretical bounds:** Our method is *extremely efficient* compared to SMC.
- **Provably Secure:** We do not assume any trust or security at the servers.
- **No compromise in accuracy:** Faithful image encoding with PSNR of around 50.
- **Practical system:** It is both *scalable* and *inexpensive*, making privacy preservation affordable.

Approach in brief: In our method, a frame, F, of the surveillance video is transformed into a set of seemingly random images, I_i, on which a surveillance operation is successfully carried out. Our solution utilizes the services of r, ($r > 2$) non-colluding computation servers. Each of the r transformed images I_i, is sent to a different server for processing. This ensures that the original video is not revealed to any of the servers, while together they retain the complete video content. Furthermore, accurate surveillance results are obtained since the servers jointly run the original plain-domain algorithm. The solution is not only provably secure but also computationally efficient. The interaction and the data communication among the servers is kept to a minimum and the only processing required of the camera is to generate the transformed images.

A detailed account of the implementation and analysis of a certain surveillance algorithms including change detection, optical flow and face detection, using the proposed framework are discussed. We describe the mapping of these problems to the framework and show the steps involved in carrying out the computations. The experiments have conducted to understand the computational and communication overheads at each stage, any loss in accuracy incurred by the computation, as well as the effectiveness of the data obfuscation for privacy.

1.4.4 Private Yet Efficient K-Means Clustering

In Section 1.4.3, we demonstrated that the properties of the data can be exploited to break the communication and computation bottlenecks for privacy preserving methods. In this work, we extend the proposed paradigm to address the similar privacy concerns in the related areas. We make the approach generic and propose secure protocols for doing un-supervised learning using K-means on the union of databases held by two or more parties.

Un-supervised learning deals with designing classifiers from a set of unlabeled samples. A common approach for unsupervised learning is to cluster or group unlabeled samples into sets of samples that are 'similar' to each other. K-means clustering is a powerful and frequently used technique in data mining. It is widely used to group data with similar characteristics or features together. In this work, we consider the problem of clustering on the union of confidential data that is not supposed to be revealed even to the party running the algorithm. The main challenge in designing such a protocol is to prevent the intermediate values from being leaked. Due to sheer volume of the inputs that are involved, the algorithms should be efficient, while still providing the privacy to the parties.

In this work we propose an efficient method for distributed K-means clustering with arbitrary partition. This means that there is no assumption on how the attributes of the data are distributed among the parties (and in particular, this subsumes the case of vertically and horizontally partitioned data). While this problem has been addressed before, we propose an approach other than the data perturbation, SMC and TTP. We propose a novel solution based on secret sharing such that the actual clustering is done by non-colluding servers, and the results merged by the participants.

Approach in brief: The idea is to use secret sharing and each party will send one share of its data to one of the several cloud computing servers. Since each server only knows one share of the data, without collusion they can not discover the original data. Then a protocol is proposed to securely compute K-means clustering between these servers, largely using the addition and multiplication properties of secret sharing. We compare and analyze the computation and communication overhead of our protocol against a zero-privacy protocol, under which each user sends his data (in plain) to a third party for clustering.

We use the following techniques to achieve the goals of privacy, efficiency and accuracy.

- Efficiency over SMC by working in modulo domain.
- Add random noise to ensure privacy.
- Scaling the axis (values) before adding noise to ensure accuracy.
- Limiting interaction between servers by making operations independent.

In this work, we achieve the security at the level of SMC while keeping the communication costs extremely low. We achieve this using the paradigm of the *Secret Sharing* [20] over a mesh of processing servers. Our solution is first of its type, and is both efficient and mathematically simple. In the process we also side-step the communication bottlenecks posed by the usage of SMC and asymmetric encryption schemes. Our proposed solution is not only computationally efficient but also secure independent of whether or not $P \neq NP$. The ability to do secure computation using limited interaction has the potential to extend over to more diverse data mining applications.

1.5 Organization of the Thesis

The work and the contributions in this dissertation are subdivided into the following chapters:

We discuss the security preliminaries in Chapter 2. The chapter also provides an overview to the state of the art methods in privacy, security and visual algorithms.

In Chapter 3, we propose, for the first time, a completely blind biometric authentication protocol, which takes care of concerns on user privacy, template protection, and trust issues in biometric authentication system. Our proposed protocol has signicant advantages over existing biometric cryptosystems, which uses a biometric to secure a secret key, which in turn is used for authentication.

A method that provides provable security, while allowing efficient computations for generic vision algorithms have remained elusive till now. In fact, it has been proven that one cannot achieve secure computation on a remote machine without incurring either significant computation or communication overhead. In Chapter 4, we show that, for designing secure visual algorithms one can exploit certain properties such as scalability, limited range etc, inherent to image data to break this seemingly impenetrable barrier. In the process, we develop a generic approach to achieve efficient and secure computation for any data that satisfies these properties, which could have potential applications in domains beyond visual surveillance. In Chapter 5, we propose an efficient method for distributed K-means clustering with arbitrary partition. While this problem has been addressed before, we propose an approach other than the data perturbation, SMC and TTP. Our proposed solution is not only computationally efficient but also secure independent of whether or not P $\neq$ NP. We show that our paradigm is generic and therefore opens up a new direction of research to look at privacy preserving methods. Chapter 6 summarizes the thesis. For completeness, the Appendix provides a brief description of some of the theorems and proofs that are used in the work here.

Chapter 2

Background and the Preliminaries

In early stages of research and development in many fields of computer science security was not an issue from the beginning. This is well true for the development of operating systems, email transmission or wireless network. As the amount of visual data increases, automatic methods for visual analysis are gaining popularity, both for accuracy and economic purpose. It becomes absolutely critical that the visual data be secured before transmission or storage, so that leakage of the data does not pose a risk in security or privacy.

Traditional encryption based methods for securing visual data are good for video storage and transmission. However, they do not allow one to carry out computations on the data, which is essential for online visual applications. Encryption and decryption as such are expensive protocols and results in significant data expansion. The communication and computation overheads become critical when dealing with voluminous data such as images and videos. Thus, what we need is lightweight encryption's, such that the server should be able to carry out the computations on the encrypted data and also store the stream if required, without being able to decipher the actual contents of the image.

The primary challenge is in providing the ability to perform generic computations on the data, while ensuring provable security. As long as one can recreate any coarse approximation of the image from the secured version, which reveals any of its contents, the image can not be considered as secured. Moreover, the neighboring pixels of an image tends to have similar values. This is a challenge to image encryption as it can be used to guess the exact value of a pixel, even if only a few LSBs or MSBs are known. Security based on public key encryption such as RSA is high, but it creates very high computational and communication overheads in order to be able to do general purpose computations. For example, approaches such as *Blind vision* [22] requires significant communication overhead between the two parties to achieve a specific goal of face detection.

Most people accept an insecure solution for an application, since security is often associated with significant computation overheads. However, we always prefer a secure solution over an insecure one, if the apparent overhead in resources is not significant, as evidenced by the adoption of the secure http protocol (https). We believe that a practical solution to a secure visual algorithm would have a similar impact if it is provided without any apparent additional cost.

In the following sections we provide a brief introduction to the background details which are used in the subsequent chapters. First, we introduce some basic concepts of cryptography with particular emphasis on their suitability for visual algorithms. We then discuss the approaches traditionally taken to ensure security and privacy in visual data.

2.1 Security Preliminaries

The relevance of carrying out the algorithm directly on encrypted data is entirely dependent on the security requirements of the application scenario under consideration. On the other hand, the particular implementation of the signal processing algorithm will be determined strongly by the possibilities and impossibilities of the cryptosystem employed. Finally, it is very likely that new requirements for cryptosystems will emerge from secure signal processing operations and applications.

In this section we present a brief overview of the cryptographic primitives that have been used for the privacy preserving methods. We discuss the concepts of securing the data using methods such as PKC and perturbation techniques. Protocols for securely computing on private data using techniques such as SMC, homomorphic encryption are also discussed.

2.1.1 Public Key Encryption (PKC)

The process of converting the plaintext (P) to ciphertext (C) using an algorithm is called encryption (E). On the otherhand, restoring the plaintext from the ciphertext is called decryption (D). Public key Encryption (PKC), also known as asymmetric cryptography, is a form of cryptography in which key used to encrypt a message differs from the key used to decrypt it. Private key is kept secret, while the public key can be widely distributed. The message that needs to be conveyed to the recipient is encrypted using his public key. It can only be decrypted by the corresponding private key. These keys are related mathematically, but the private key cannot be practically derived from the public key.

In practice, PKC can be used to ensure confidentiality of the data. The messages encrypted with a recipients public key can only be decrypted using the corresponding private key. The private key of which is known only to the intended receiver. Asymmetric key algorithms are generally found to be computationally expensive. Some of the popular PKC algorithms include RSA [111], Pailliers [102], El-Gamal's [55] etc.

2.1.2 Data Perturbation Techniques

Data perturbation techniques tend to secure the data by adding randomness to it. The idea is to alter the data so that the actual original data values cannot be recovered, while preserving the utility of the data for statistical analysis. Privacy is preserved since the noisy version of the data does not reveal the real data values. However, carrying out analysis on the perturbed data results in approximately correct analysis. Privacy is enhanced by adding more and more noise, however, this leads to a rapid degradation of the results. Visual data is inherently noisy and this must be taken care of. Moreover, certain operations such as comparison of noisy data requires special treatment.

2.1.3 Secure Multi-party Computation (SMC)

Secure multi-party computation (SMC) is a problem in cryptography that was initially suggested by Andrew C. Yao [143]. Yao introduced the *millionaire problem*, in which two millionaire's Alice and Bob want to find out who is richer without revealing the precise amount of their wealth to anyone.

Generalization of the Yao's protocol gave way to secure multi-party computation. In SMC, a given number of participants p_1, p_2, ..., p_N each have a private data, respectively d_1, d_2, ..., d_N. The participants want to compute the value of a public function $F()$ on N variables at the point (d_1, d_2, ..., d_N). An SMC protocol is said to be secure if no participant can learn more than the description of the public function and the result of the global calculation than what s/he can learn from his/her own entry.

The computation and communication complexity of the protocol proposed by Ioannis et al. [71] is $O(d^2)$, where 2^d is the upper bound on their numbers which they want to compare. An important primitive in SMC is *oblivious transfer (OT)*. An OT is a protocol by which a sender sends some information to the receiver, but remains oblivious as to what is received.

2.1.3.1 Oblivious Transfer (OT)

Oblivious Transfer (OT) allows Alice to choose one element from a database of elements that Bob holds without revealing to Bob which element was chosen and without learning anything about the rest of the elements. The notion of OT was suggested by Even, Goldreich and Lempal [57] as a generalization of Rabin's OT [12]. 1-2 oblivious transfer or "1 out of 2 oblivious transfer' is a critical problem in cryptography and is used in building protocols for *secure multi-party computation*. In particular, it is 'complete' for secure multiparty computation: that is given an implementation of oblivious transfer it is possible to securely evaluate any polynomial time computable function without any additional primitive.

1-2 OT was generalized to 1-n OT by Brassard et al. [36], such that Bob has an array of size n and Alice wants to obliviously choose the i^{th} element. The communication complexity of the OT protocol [90] is $O(log^2(n))$ and the computation complexity is $O(n)$.

Formally oblivious transfer can be formulated as follows: Bob privately owns two elements M_0 and M_1 and Alice wants to receive one of them without letting Bob know which one. Bob is willing to let her do so provided that she will not learn anything about the other element. The following protocol (see Algorithm 1), based on RSA encryption can be used to solve the problem in a semi-honest setting. Alice has $\sigma \in \{0, 1\}$, Bob has data M_0, M_1 and Alice learns M_σ.

Algorithm 1 1-2 Oblivious Transfer

```
1: Bob sends Alice two different public encryption keys $K_0$ and $K_1$.
2: Alice generates a key $K$ and encrypts it with $K_0$ or $K_1$. For the sake of argument, let's
   say she chooses $K_0$. She sends Bob $E(K, K_0)$; that is, she encrypts $K$ with one of Bob's
   public keys.
3: Bob does not know which public key Alice used, so he decrypts with both of his private
   keys. He thus obtains both the real key $K$, and a bogus one $K'$.
4: Bob sends Alice $E(M_0, K)$ and $E(M_1, K')$, in the same order he sent the keys $K_0$and $K_1$
   in step 1. Alice decrypts the first of these messages with the key $K$ and obtains $M_0$.
```

2.1.4 Homomorphic Encryption

Homomorphic encryption is a form of encryption where one can perform a specific algebraic operation on the plaintext by performing a corresponding algebraic operation on the ciphertext. That is, it is a way of encoding data x into $E(x)$ such that one can compute a function $f(x,\ y)$ easily knowing only $E(x)$ and $E(y)$. Here $f()$ is a function composed of either addition only or multiplication only but not a combination of both. For example, $E(x * y) = E(x) * E(y)$.

There are several efficient homomorphic cryptosystems such as *RSA cryptosystems* [111], *ElGamal cryptosystem* [55], *Paillier cryptosystem* [102] etc.

In the year 2009, the first fully homomorphic cryptosystem was constructed by Craig Gentry [63]. The scheme can potentially support an unbounded number of additions and multiplications. However, the computation time and ciphertext size increase sharply as one increases the security level. To obtain 2^k security against known attacks, the computation time and ciphertext size are high-degree polynomials in k. This makes the scheme impractical for many applications.

2.1.5 Secret Sharing

Unconditionally or information-theoretically secure SMC is closely related to the problem of secret sharing. Secret sharing refers to method for distributing a secret amongst a group of participants, each of which is allocated a share of the secret. The secret can be reconstructed only when a sufficient number of shares are combined together; individual shares are of no use on their own. Secret sharing schemes was invented by Adi Shamir [118] in the year 1979.

More formally, in a secret sharing scheme there is one dealer and n players. The dealer gives a secret to the players, but only when specific conditions are fulfilled. The dealer accomplishes this by giving each player a share in such a way that any group of t (for threshold) or more players can together reconstruct the secret but no group of fewer than t players can. Such a system is called a $(t,\ n)$-threshold scheme.

There are several variants of secret sharing schemes known in literature. For example Shamir's [118] scheme uses polynomial interpolation, while the Blakley's scheme [31] represents the secrets as the planes in the n-dimensional space.

2.1.6 Pseudo-Random Number Generators

A random number is a number that cannot be predicted by an observer before it is generated. If the number is to be in the range $0,\ \ldots,\ 2^n - 1$, and an observer cannot predict that number with probability any better than $1/2^n$. If an algorithm generates m random numbers, and $m-1$ of these are revealed to an observer. The numbers are said to be truly random, if even with this information an observer cannot predict the m^{th} with any better probability than $1/2^n$.

Pseudo-random generators are fundamental to many theoretical and applied aspects of computing. A *pseudo-random number generator (PRNG)* is an algorithm for generating a sequence of numbers that approximates the properties of random numbers. The sequence even though is not truly random but, is completely determined by a relatively small set of initial values, called the *PRNGs state*.

An arbitrary seed state can initialize a PRNG sequence, whose maximum period is determined by the number of bits in the size of the seed state. Increasing the size of the seed state by a single bit will double the length of the maximum period. For example if a PRNGs internal state contains p bits, its period can be no longer than 2^p results. Therefore, it is easy to build PRNGs with long periods for many practical applications.

2.2 Traditional Methods to Privacy and Security in Visual Data

The problem of introducing privacy and security in visual data processing was addressed with considerable success in different domains. The work in secure algorithms for image analysis and recognition have primarily been in two directions. The first group attempts to provide custom solutions to specific algorithms for biometric authentication [108], video surveillance [99], etc. These approaches take advantage of the properties of the algorithms, the data or the application setting to come up with specific data transformations [144] or even capture the data in a fashion that alleviates security concerns [40]. However these approaches do not provide any guarantee of privacy as they rely on the success of certain computer vision tasks, such as face detection.

A second class of algorithms try to provide more general purpose computation ability along with security, and are closer in spirit to our work. Approaches such as secure multi-party computation and oblivious transfer [65], which are well known in the security community, are often adopted to incorporate security into vision computing tasks. Avidan *et al.* [22] proposed the use of secure multi-party computation to achieve a secure system for face detection. Shansank *et al.* [119] addressed the privacy concerns about the user's query to a content based image retrieval system. Barni *et al.* [101] and Nagia *et al.* [97] on the other hand uses the homomorphic encryption techniques to achieve oblivious neural network computation.

Strong encryption based approaches that rely of multiple round of communication are popular in data mining from textual data. However, image and video data are extremely bulky in comparison to text data, and the computational and communication overhead of these approaches make their application to visual data, prohibitively expensive to be practical. In short, the existing approaches that are efficient are extremely limited in their scope, while algorithms that provide general computation ability rely on public key cryptography or heavy interaction, making them less acceptable to the users.

Hence, solutions based on these cryptographic primitives would be impractical for our desired applications. In this thesis we propose solutions that are not only provably secure but also computationally efficient. Protocols are designed such that the interaction and the data communication among the servers is kept to a minimum.

For the completeness sake, in the following sub-sections, we first give a brief introduction to the traditional methods used for video encryption. We then discuss the light-weight algorithms used for hiding the region of interest in the videos. We also provide an insight into the recent advances in visual data security.

2.2.1 Video Encryption Methods

Multimedia data encryption is used for addressing the Digital Rights Management (DRM) for Multimedia. The methods attempts to prevent unauthorized disclosure of confidential multimedia information in transit or storage. In the past, many algorithms have been proposed by which multimedia data can be protected. The key factors to consider in choosing one method over the other is *i)* suitable security level for an application, and *ii)* cost effectiveness for the specific application.

In general, the technique for multimedia encryption is to treat the video as a traditional digital data, such as text. This data is then secured using a classical encryption scheme such as PKC. At the recievers end, the entire cipher data stream is decrypted and playback can be performed at the client device.

However, applying this method alone is not enough to secure multimedia data that is broadcast on wireless or satellite networks. A variety of constraints like time, security, compression rate, etc restricts the usage of many popular encryption methods for securing the multimedia data.

Multimedia data streams has many different characteristics from traditional digital data streams. They are larger in size, compressed for transmission and storage, and are coded in a different way for different applications. From an end-user's point of view, low decryption and re-encryption cost overhead is critical. So is the additional hardware required, if any. Thus, selecting an application adequate encryption is gaining popularity.

The cryptographic encryption methods such as DES (Data Encryption Standard) [48], RC5 (Rivest Cipher) [112], AES (Advanced Encryption Standard) [110] etc have been used in securing the multimedia stream. However, even on modern hardware, these schemes are computationally very expensive for many real-time video applications.

In order to reduce the large volumes of the data to be encrypted, selective encryption methods have been proposed. For a given video stream, not all parts of the video are important and thus need not be encrypted. Selective encryption [15, 80, 92, 104] intends to encrypt only some parts (region of interest) of the entire data stream, like MPEG headers, thus reducing the overall computational requirements. Partial encryption methods does not strive for maximum security. In general, for a given application it trades off security for computational complexity.

For real-time video applications, selective cum light-weight encryption algorithms are preferred than encryption of the complete video data. In these types of encryption methods, selected data of video are encrypted based on the video properties.

2.2.2 Light-Weight Algorithms

For many practical applications such as video-on-demand, the adversary is not interested in exploring expensive (cost and time) attacks to breach the security. Thus, for such practical applications, encryption schems ensuring even partial privacy are sufficient. Thus, light-weight encryption and decryption methods are suitable for securing certain multimedia data.

There are several light-weight encryption algorithms [18, 19, 34, 87] proposed for protection of the video data. Many of these algorithms are based on the XOR and scrambling based operations. For example Shi *et al.* [120] uses the XOR on the sign bits of the DCT co-efficients to encrypt an MPEG video. Where as, Choon *et al.* [42] proposes an encryption scheme based on the Shannon principle of confusion and diffusion. A scramble based approach is suggested by L. Tang [125], who uses a random permutation of the DCT co-efficients. A major limitation of such schemes is the weak security provided by such schemes and hence can not be used in applications such as military and video conferencing.

Problem specific approaches have also been proposed to address the specific concerns in videos and images. The idea is to identify and obnubilate the region of interest (such as human face) from the video stream before transmitting it over to the network. Senior *et al.* [116] presented a model to define video privacy and re-render the video in a privacy-preserving manner. In today's age of video surveillance,

surveillance cameras are being increasingly used to monitor the public places such as shopping centers, airports etc. This raises the privacy concerns for the individuals, and for many practical scenarios, action identification is sufficient to ensure security. *De-identification* is the technique to protect the privacy of the individuals by hiding/removing all personal identification information from the videos. The objective is to render videos in a privacy preserving manner, while retaining sufficient information about the human activity. For example, face swapping [30] and face de-identification [99, 122] try to modify face images such that they can be automatically detected, but yet cannot be correctly recongnised. These approaches do not provide any gurantee of privacy as they rely on the success of certain computer vision tasks, such as face detection.

Steganography techniques are used for concealing information in other, seemingly innocent media. For example, concealing messages within the lowest bits of noisy images and videos. On the other hand, *digital watermarking* is the process of embedding information into a digital signal in a way that is difficult to remove. Watermarking schemes usually rely on a symmetric key for both embedding and detection, which is critical to both the robustness and security of the watermark and thus needs to be protected. One application of watermarking is in copyright protection systems, which are intended to prevent or deter unauthorized copying of digital media. The framework proposed by Zhang *et al.* [145] stores the privacy information in surveillance video as a watermark and monitors an invalid person in a restricted area while protecting the privacy of the valid persons.

2.2.3 Recent Advances

Treating the digital content as a binary data and securing it using the cryptographic primitives is not realistic and eliminates the possibility of further processing. In several application scenarios, however, it is desirable to carry out signal processing operations directly on encrypted signals. The possibility of processing encrypted data has been advanced several years ago. The peculiarities of the visual data with respect to other classes of data more commonly encountered in the cryptographic literature poses many challanges. The general cryptographic tools that allow to process encrypted signals are SMC and homomorphic cryptosystems. The limitations of these general protocols is that they are infeasible for situations where the parties own huge quantities of data or the functions to be evaluated are complex, as it happens in signal processing applications.

Blind Vision proposed by Shai Avidan and Moshe Butman [22] applies secure multi-party computation techniques to vision algorithms. They propose a method for securely evaluating a Viola-Jones type face detector [134]. In their application scenario, Bob offers a face-detection web service where clients can submit their images for analysis. Alice would very much like to use the service, but is reluctant to reveal the content of her images to Bob. Bob, for his part, is reluctant to release his face detector to anyone. Blind Vision uses the standard cryptographic tools to solve this problem without leaking any information. Unfortunately, these methods are too slow to compute, taking hours to scan a single image.

The authors, improved upon the efficiency by proposing a couple of machine learning techniques that allow the parties to solve the problem while leaking a controlled amount of information [23]. The first method is an information-bottleneck variant of AdaBoost that lets Bob find a subset of features that are enough for classying an image patch, but not enough to actually reconstruct it. The second machine learning technique is active learning that allows Alice to construct an online classifier, based on a small number of calls to Bob's face detector. She can then use her online classifier as a fast rejector before using a cryptographically secure classifier on the remaining image patches.

Blind Vision addresses the problem at the expense of heavy computation. The authors extended their approach for privacy preserving pattern classification [21]. The authors propose SMC based protocols to generic pattern classification for classifiers such as threshold function, polynomial function, gaussian function etc. They adopt a lookup table approach to a kernel function evaluation, where a lookup table approximates the range of values taken by the feature vectors. However, SMC is used as the building block in all these methods. There is a need to accelerate the protocols either by relying on different cryptographic primitives or by taking advantage of domain specific knowledge.

Private Content Based Image Retrieval (PCBIR) deals with retrieving similar images from an image database without revealing the content of the query image, not even to the database server. Shashank *et al.* [119] proposed algorithms for PCBIR, when the database is indexed using hierarchical index structure or hash based indexing scheme. PCBIR is achieved by exchange of messages between the user and the database. These messages collectively help the user in inferring the required information from the database but prohibit the database from knowing the user's interest.

PCBIR is similar to private information retrieval (PIR) schemes [43, 139] that allow a user to obtain the data stored at a specific address in a database whilst keeping the database oblivious of the address. However, in practice, the address corresponding to the correct answer is typically unknown a priori to the user. PIR is concerned about point queries, that is the value at a particular position, while PCBIR deals with a similarity search. PCBIR is also different from blind vision since it requires privacy in only one direction. The whole database is often public while query is private. PCBIR is concerned with efficient retrieval under the privacy constraint without trading the recall and precision.

The general SMC based solution to PCBIR is usually quite inefficient when compared to tailor-made solution for the same. The authors showed that, since image retrieval is fundamentally a similarity search, PCBIR can be more efficiently solved than PIR. They improved upon the efficiency of SMC by exploiting the clustered nature of image databases. They showed that the image databases are amenable to significant faster private retrieval on reasonably large databases using a variety of state of the art indexing schemes.

Chapter 3

Blind Authentication: A Secure-Crypto Biometric Verification Protocol

Concerns on widespread use of biometric authentication systems are primarily centered around template security, revocability and privacy. The use of cryptographic primitives to bolster the authentication process can alleviate some of these concerns as shown by biometric cryptosystems. In this work, we propose a *provably secure* and *blind* biometric authentication protocol, which addresses the concerns of user's privacy, template protection, and trust issues. The protocol is blind in the sense that it reveals only the identity, and no additional information about the user or the biometric to the authenticating server or vice-versa. As the protocol is based on asymmetric encryption of the biometric data, it captures the advantages of biometric authentication as well as the security of public key cryptography. The authentication protocol can run over public networks and provide non-repudiable identity verification. The encryption also provides template protection, the ability to revoke enrolled templates, and alleviates the concerns on privacy in widespread use of biometrics.

The proposed approach makes no restrictive assumptions on the biometric data and is hence applicable to multiple biometrics. Such a protocol has significant advantages over existing biometric cryptosystems, which use a biometric to secure a secret key, which in turn is used for authentication. We analyze the security of the protocol under various attack scenarios. Experimental results on four biometric datasets (face, iris, hand geometry and fingerprint) show that carrying out the authentication in the encrypted domain does not affect the accuracy, while the encryption key acts as an additional layer of security.

3.1 Biometrics-based Authentication Systems

Reliable user authentication is a critical task in the web-enabled world. Surrogate representations of identity such as passwords and ID cards are not sufficient for reliable identity determination, as they can be easily misplaced, shared or stolen. Once an intruder acquires the user ID and the password, the intruder has total access to the user's resources. In addition, there is no way to positively link the usage of the system or service to the actual user. That is, there is no protection against repudiation by the user

ID owner. Thus, in the modern distributed systems environment, traditional authentication protocols based on a simple combination of user ID and password has become inadequate.

Biometric authentication is the task of verifying the claimed identity of someone, by using their anatomical and behavioral traits. A biometric system provides automatic recognition of an individual based on some unique features or characteristics possessed by the individual. Biometric systems have been developed based on common biometric traits such as fingerprint, facial features, iris, hand geometry, voice, handwriting, etc. (see Figure 3.1).

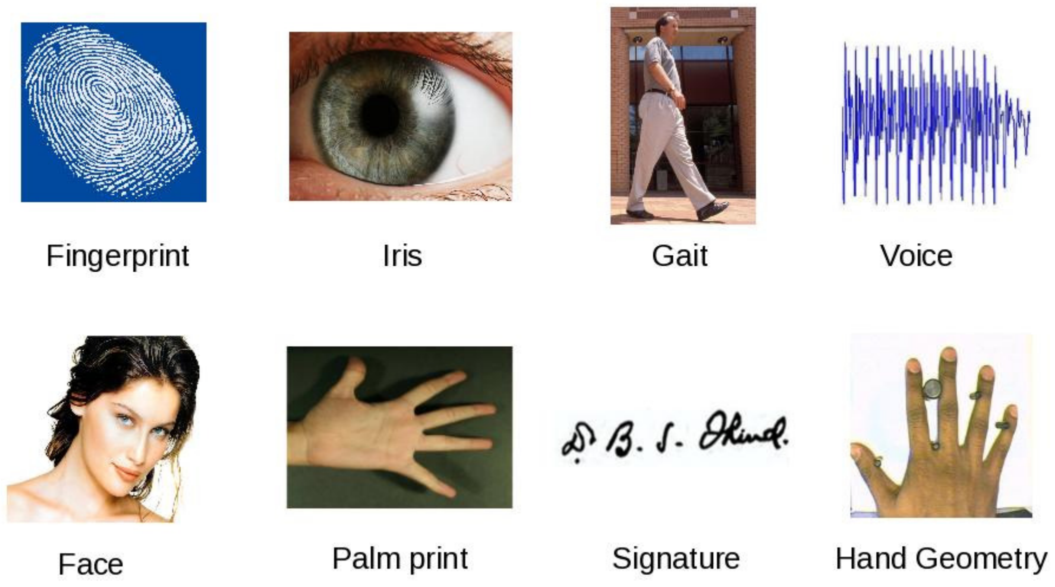


Figure 3.1 Some of the commonly used biometric modalities used for recognition.

A good biometric is characterized by use of a feature that is; *highly unique* - so that the chance of any two people having the same characteristic will be minimal, *stable* - so that the feature does not change over time, and be *easily acquired* - in order to provide convenience to the user, and prevent misrepresentation of the feature.

A biometric authentication system (see Figure: 3.2) consists of two phases, *i)* Enrollment phase and, *ii)* Authentication phase. During the *enrollment phase*, a user (say, Alice) scans her biometric data, from which features template is created and stored, either in a central database, or on a mobile device. The biometric template provides a normalized, efficient and highly discriminating representation of the features. During the *authentication phase*, a user who claims to be Alice would scan his/her biometric data again, and the same feature extraction algorithm is applied to the biometric. The resulting template is then compared with the stored template of the user Alice. If they are sufficiently similar according to some similarity measure, the matching algorithm outputs a yes, which indicates that the user is authentic, or a no when the user is not authentic.

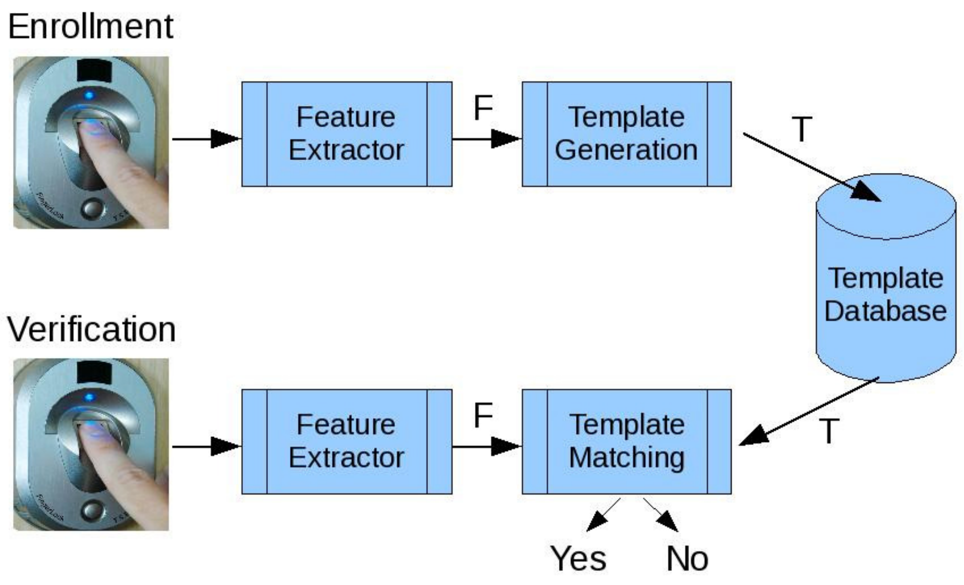


Figure 3.2 Biometric Authentication System.

A typical biometric system is compromised of five integrated components (see Figure 3.3):

1. A **sensor** is used to collect and convert the information to a digital form.
2. **Feature extractor** performs the quality control activities and computes a biometric template.
3. **Template storage** keeps information that new biometric templates will be compared to, along with the user's identity.
4. A **matching algorithm** to compare a new template to one or more templates kept in data storage.
5. A **decision module** uses the result from the matching algorithm to make the authentication decision and initiates a response to the query.

Due to the rapid growth in sensing and computing technologies, biometric systems have become affordable and are easily embedded in a variety of consumer devices (e.g. mobile phones), making this technology vulnerable to the malicious designs of criminals. It is important that such biometrics-based authentication systems be designed to withstand attacks when employed in security-critical applications, especially in unattended remote applications such as e-commerce.

One of the greatest strength of biometrics is that the biometrics does not change over time. This at the same time is its greatest liability. Once a biometric data has been compromised, it is compromised forever. Since it is difficult to replace or revoke biometric data, it it important to securely keep the user's

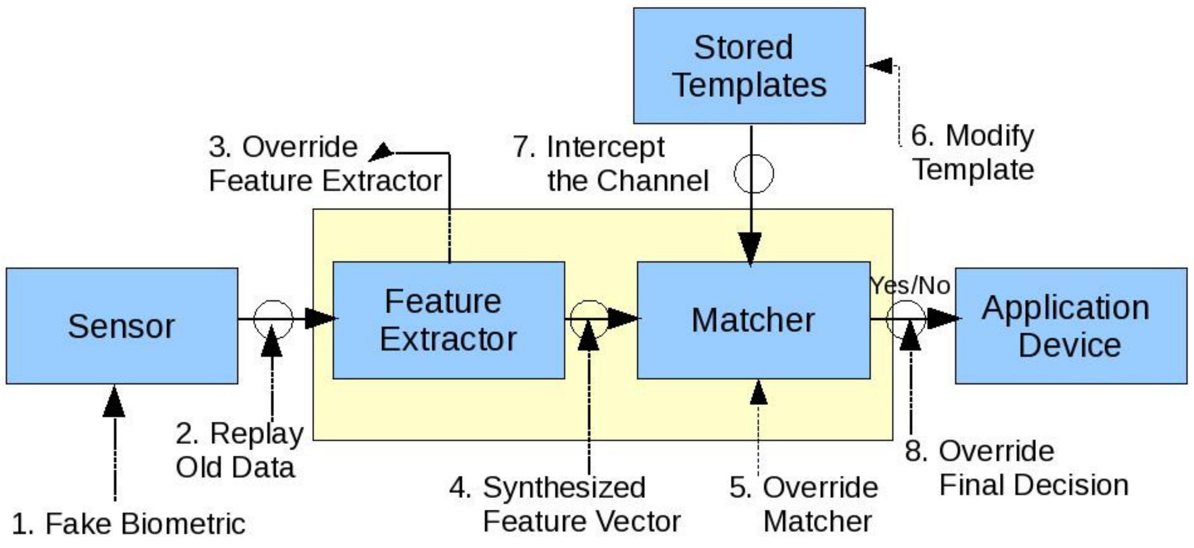


Figure 3.3 Points of Attack in a generic biometric system.

biometric data and template when they are used in authentication systems. Thus, template security is one of the most crucial issues in designing a secure biometric system.

Adversary attacks generally exploit the system vulnerabilities at one or more modules or interfaces. Ratha et al. [108] identified eight points of attack in a biometric system (see Figure 3.3). Among these vulnerabilities, an attack against stored biometric templates is a major concern due to the strong linkage between a user's template and his identity and the irrevocable nature of biometric templates.

One advantage of passwords over biometrics is that they can be re-issued. If a token or a password is lost or stolen, it can be cancelled and replaced by a newer version. This is not naturally available in biometrics. If someone's fingerprint is compromised from a database, they cannot cancel or reissue it. *Cancelable biometrics* is a way in which to incorporate protection and the replacement features into biometrics. It was first proposed by Ratha et al. [108]

3.2 Introduction to Blind Authentication

Biometric authentication systems are gaining wide-spread popularity in recent years due to the advances in sensor technologies as well as improvements in the matching algorithms [75] that make the systems both secure and cost-effective. They are ideally suited for both high security and remote authentication applications due to the non-repudiable nature and user convenience. Most biometric systems assume that the template in the system is secure due to human supervision (e.g., immigration checks and criminal database search) or physical protection (e.g., laptop locks and door locks). However, a variety of applications of authentication need to work over a partially secure or insecure networks such as an ATM networks or the Internet. Authentication over insecure public networks or with untrusted servers raises more concerns in privacy and security. The primary concern is related to the security of the plain biometric templates, which cannot be replaced, once they are compromised [108]. The privacy concerns arise from the fact that the biometric samples reveal more information about its owner (medical, food habits, etc.) in addition to the identity. Widespread use of biometric authentication also raises concerns of tracking a person, as every activity that requires authentication can be uniquely assigned to an individual (see Table 3.1).

To clarify our problem let us consider the following usage scenario: *"Alice wants to create an account in Bobmail, that requires biometrics based authentication. However, she neither trusts Bob to handle her biometric data securely, nor trusts the network to send her plain biometric."*

The primary problem here is that, for Alice, Bob could either be incompetent to secure her biometric or even curious to try and gain access to her biometric data, while the authentication is going on. So Alice does not want to give her biometric data in plain to Bob. On the other hand, Bob does not trust the client as she could be an impostor. She could also repudiate her access to the service at a later time. For both parties, the network is insecure. A biometric system that can work securely and reliably under such circumstances can have a multitude of applications varying from accessing remote servers to e-shopping over the Internet. Table 3.1 summarizes the primary concerns that needs to be addressed for widespread adoption of biometrics. For civilian applications, these concerns are often more serious than the accuracy of the biometric [13].

a) Template protection: As a biometric do not change over time, one cannot revoke an enrolled plain biometric. Hence, critical information could be revealed if the server's biometric template database is compromised.
b) User's privacy: i) The activities of a person could be tracked, as the biometric is unique to a person, and ii) Certain biometrics may reveal personal information about a user (e.g., medical or food habits), in addition to identity.
c) Trust between user and server: In widespread use, all authenticating servers may not be competent or trustworthy to securely handle a user's plain biometric, while a remote user cannot be reliably identified without biometric information.
d) Network security: As the authentication is done over an insecure network, anyone snooping the network could gain access to the biometric information being transmitted.

Table 3.1 Primary concerns in widespread adoption of biometrics for remote authentication.

If the user is able to authenticate himself using a strongly encrypted version of his biometric (say using RSA [111]), then many of the concerns on privacy and security are addressed. However, this would require the server to carry out all the computations in the encrypted domain itself. Unfortunately, encryption algorithms are designed to remove any similarity that exist within the data to defeat attacks, while pattern classification algorithms require the similarity of data to be preserved to achieve high accuracy. In other words, security/privacy and accuracy seems to be opposing objectives. Different secure authentication solutions try to make reasonable trade-offs between the security and accuracy, in addition to making specific assumptions about the representation or biometric being used.

We overcome this seemingly unavoidable compromise by designing the classifier in the plain feature space, which allows us to maintain the performance of the biometric. We would then like to carry out the computations required for authentication using this trained classifier, completely in the encrypted domain. However, such a solution would require an *algebraic homomorphic encryption* scheme [61]. The only known doubly homomorphic scheme has recently been proposed by Craig Gentry [63] and would mostly lead to a computationally intensive theoretical solution. We show that it is possible to achieve a practical solution using distribution of work between the client (sensor) and the server (authenticator), using our proposed randomization scheme.

3.2.1 Previous Work

The previous work in the area of encryption based security of biometric templates tend to model the problem as that of building a classification system that separates the genuine and impostor samples in the encrypted domain [58] [126] [73]. However a strong encryption mechanism destroys any pattern in the data, which adversely affects the accuracy of verification. Hence, any such matching mechanism necessarily makes a compromise between template security (strong encryption) and accuracy (retaining patterns in the data). The primary difference in our approach is that we are able to design the classifier in the plain feature space, which allows us to maintain the performance of the biometric itself, while carrying out the authentication on data with strong encryption, which provides high security/privacy.

Over the years a number of attempts have been made to address the problem of template protection and privacy concerns and despite all efforts, as A.K. Jain *et al.* puts it, *a template protection scheme with provable security and acceptable recognition performance has thus far remained elusive.* [73]. In this section, we will look at the existing work in light of this security-accuracy dilemma, and understand how this can be overcome by communication between the authenticating server and the client. Detailed reviews of the work on template protection can be found in Jain *et al.* [73], Uludag *et al.* [130], and Ratha *et al.* [109]. We will adopt the classification of existing works provided by Jain *et al.* [73] (see Fig 3.4), and show that each class of approaches makes the security-accuracy compromise.

Let us now analyze each of the four category of solutions in terms of their strengths and weaknesses:

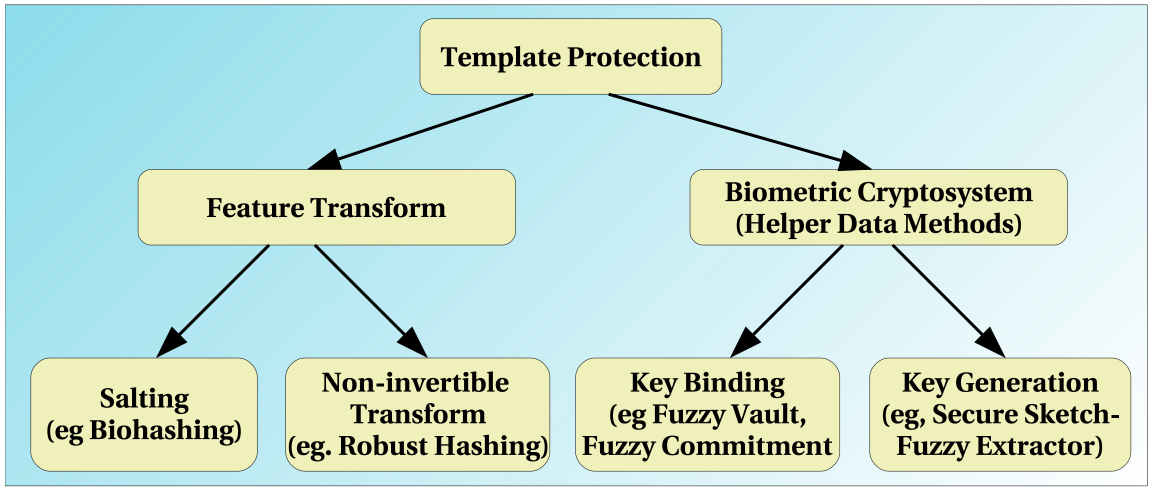

Template Protection
Feature Transform
Biometric Cryptosystem (Helper Data Methods)
Salting (eg Biohashing)
Non-invertible Transform (eg. Robust Hashing)
Key Binding (eg Fuzzy Vault, Fuzzy Commitment
Key Generation (eg, Secure Sketch-Fuzzy Extractor)

Figure 3.4 Categorization of template protection schemes by Jain *et al.* [73].

The first class of feature transformation approaches known as *Salting* offers security using a transformation function seeded by a user specific key. The strength of the approach lies in the strength of the key. A classifier is then designed in the encrypted space. Although the standard cryptographic encryption such as AES or RSA offers secure transformation functions, they cannot be used in this case. The inherent property of dissimilarity between two instances of the biometric trait from the same person, leads to large differences in their encrypted versions. This leads to a restriction on the possible functions that can be used and in salting, resulting in a compromise made between security and the performance. Some of the popular salting based approaches are biohashing [127] [126] and salting for face template protection [114]. Moreover, salting based solutions are usually specific to a biometric trait, and in general do not offer well defined security. Kong *et al.* do a detailed analysis of the current biohashing based biometric approaches [84]. They conclude that the zero EER reported by many papers is obtained in carefully set experimental conditions and unrealistic under assumptions from a practical view point.

The second category of approaches identified as *Non-invertible transform* applies a trait specific non-invertible function on the biometric template so as to secure it. The parameters of the transformation function are defined by a key which must be available at the time of authentication to transform the query feature set. Some of the popular approaches that fall into this category are Robust Hashing and Cancelable Templates. Cancelable templates [45, 109] allows one to replace a leaked template, while reducing the amount of information revealed through the leak, thus addressing some of the privacy concerns. However, such methods are often biometric specific and do not make any guarantees on preservation of privacy [35], especially when the server is not trusted. Methods to detect tampering of the enrolled templates [76] help in improving the security of the overall system.

Boult *et al.* [35] extended the above approach to stronger encryption, and proposed an encrypted minutia representation and matching scheme of fingerprints. The position information of a minutia is divided into a stable integer part and a variable increment. A *Biotoken* consists of the encrypted integer part and the increment information in plain. A specific matching algorithm was proposed to match the biotokens for verification. The approach provides provable template security as a strong encryption is used. Moreover, the matching is efficient, and is shown to even improve the matching accuracy. However, the primary fact that encryption is applied to part of the data, which itself is quantized, may mean some amount of compromise between security and accuracy. An extension to the above work based on re-encoding methodology for revocable biotokens is proposed by the authors in [115]. In this method, the computed biotoken is re-encoded using a series of unique new transformation functions to generate a *Bipartite Biotoken*. For every authentication, the server computes a new bipartite biotoken, which is to be matched by the client against the biotoken generated by him. The method significantly enhances the template security as compared to the original protocol. Moreover, as bipartite biotoken is different for each authentication request, replay attacks are not possible. However, in the current form, the base biotoken is available (in plain) with the server, and if the biotoken database is compromised, a hacker can gain access to all the users' accounts until the biotokens are replaced. The method aims at securing the actual biometric template, which cannot be recovered from a secure biotoken.

The third and fourth classes, shown in Fig 3.4, are both variations of *Biometric cryptosystems*. They try to integrate the advantages of both biometrics and cryptography to enhance the overall security and privacy of an authentication system. Such systems are primarily aimed at using the biometric as a protection for a secret key (Key Binding approach [79]) or use the biometric data to directly generate a secret key (Key Generation approach [50]). The authentication is done using the key, which is unlocked/generated by the biometric. Such systems can operate in two modes in the case of remote authentication. In the first case, the key is unlocked/generated at the client end, which is sent to the server for authentication, which will ensure security of the template, and provide user privacy. However, this would become a key based authentication scheme and would lose the primary advantage of biometric authentication, which is its non-repudiable nature. In the second case, the plain biometric needs to be transmitted from the user to the server, both during enrollment and during authentication. This inherently leaks more information about the user than just the identity, and the users need to trust the server to maintain their privacy (concerns Table 3.1: *b* and *c*). Moreover, authenticating over an insecure network makes the plain biometric vulnerable to spoofing attacks (concerns Table 3.1: *d*).

Biometric cryptosystem based approaches such as Fuzzy Vault and Fuzzy extractor in their true form lack diversity and revocability. According to Jain *et al.* [73], a performance degradation usually takes place as the matching is done using error correction schemes. This precludes the use of sophisticated matchers developed specifically for matching the original biometric template. Biometric cryptosystems, along with salting based approaches introduce diversity and revocability in them. Moreover, Walter *et al.* [135] demonstrated a method for recovering the plain biometric from two or more independent secrets secured using the same biometric. A detailed review of the previous work in this area can be found in Uludag et al. [130] and Jain et al. [73].

Nagai *et al.* [97] proposed the use of client side computation for part of the verification function. Their approach, termed *ZeroBio*, models the verification problem as classification of a biometric feature vector using a 3-layer neural network. The client computes the outputs of the hidden layer, which is transferred to the server. The client then proves to the server that the computation was carried out correctly, using the method of zero-knowledge proofs. The server completes the authentication by computing the output values of the neural network. The method is both efficient and generic as it only requires computation of weighted sums and does not make any assumption on the biometric used. It also provides provable privacy to the user, as the original biometric is never revealed to the server. However, the system requires that the hidden layer weights be transferred to the server without encryption. This allows the server to estimate the weights at the hidden layer from multiple observations over authentications. Once the weights are known, the server can also compute the feature vector of the biometric, thus compromising both security and privacy. The system could also be compromised if an attacker gains access to the client computer, where the weight information is available in plain.

Blind authentication, proposed by us, is able to achieve both strong encryption based security as well as accuracy of a powerful classifiers such as support vector machines (SVM [14]) and Neural Networks [29]. While the proposed approach has similarities to the Blind Vision [22] scheme for image retrieval, it is far more efficient for the verification task.

Blind Authentication addresses all the concerns mentioned in Table 3.1: -

1. The ability to use strong encryption addresses *template protection* as well as privacy concerns.

2. *Non-repudiable* authentication can be carried out even between non-trusting client and server using a trusted third party solution.

3. It provides *provable protection* against replay and client-side attacks even if the keys of the user are compromised.

4. As the enrolled templates are encrypted using a key, one can replace any compromised template, providing *revocability*, while allaying concerns of being tracked.

In addition, the framework is generic in the sense that it can classify any feature vector, making it applicable to multiple biometrics. Moreover, as the authentication process requires someone to send an encrypted version of the biometric, the non-repudiable nature of the authentication is fully preserved, assuming that spoof attacks are prevented. Note that the proposed approach does not fall into any of the categories given in Figure 3.4. This work opens a new direction of research to look at privacy preserving biometric authentication.

3.3 Blind Authentication

We define *Blind Authentication* as "A biometric authentication protocol that does not reveal any information about the biometric samples to the authenticating server. It also does not reveal any information regarding the classifier, employed by the server, to the user or client". Note that such a protocol can satisfy the conditions presented in our initial scenario, where Alice wanted to create an account with Bobmail that required biometric authentication, whom she did not trust. We now present the authentication framework that achieves this goal using any biometric, and prove that the information exchanged between the client and the server does not reveal anything other than the identity of the client.

For the sake of simplicity, we initially assume that authentication is done through a generic *linear classifier*. We later describe, how the protocol can be extended to more generic and powerful classifiers, like the *Support Vector Machine* (SVM [14]) and the *Neural Networks* [69] [29]. One could use any biometric in this framework as long as each test sample is represented using a feature vector x of length n. Note that even for biometrics such as fingerprints, one can define fixed length feature representations [58].

Let ω be the parameters of the linear classifier (perceptron). The server accepts the claimed identity of a user, if $\omega \cdot x < \tau$, where τ is a threshold. As we do not want to reveal the template feature vector (ω) or the test sample (x) to the server, we need to carry out the perceptron function computation directly in the encrypted domain. Computing $\omega \cdot x$ involves both multiplication and addition operations, thus computing it in the encrypted domain requires the usage of a doubly homomorphic encryption scheme [93]. In the absence of a practical doubly homomorphic encryption scheme (both additive and multiplicative homomorphic), our protocol uses a class of encryption that are multiplicative homomorphic, and we simulate addition using a clever randomization scheme over one-round of interaction between the server and the client. An encryption scheme, $E(x)$ is said to be multiplicative homomorphic, if $E(x)E(y) = E(xy)$ for any two numbers x and y. We use the popular RSA encryption scheme [111], which satisfies this property.

An overview of the authentication process is presented in Fig 3.5. We assume that the server has the parameter vector ω in the encrypted form, i.e., $E(\omega)$, which it receives during the enrollment phase. The authentication happens over two rounds of communication between the client and the server.

To perform authentication, the client locks the biometric test sample using her public key and sends the *locked ID* to the server. The server computes the products of the locked ID with the locked classifier parameters and randomizes the results. These *randomized products* are sent back to the client. During the second round, the client unlocks the randomized results and computes the sum of the products. The

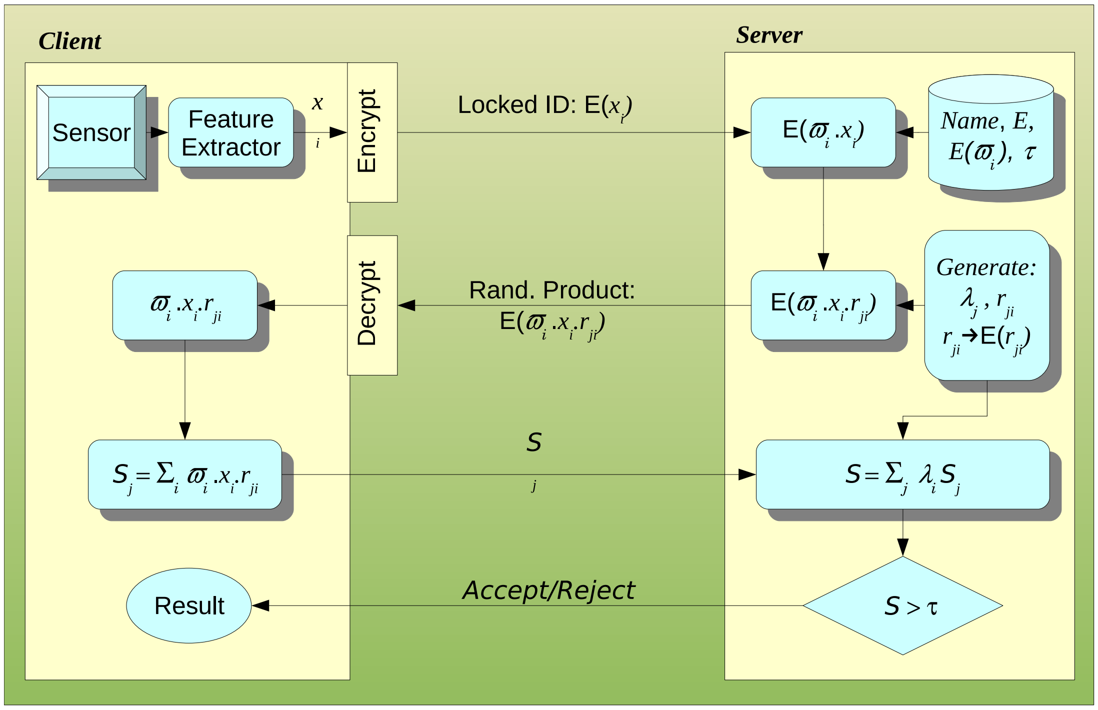


Figure 3.5 Blind Authentication Process: Linear kernel computation for encrypted feature vectors. At no point, the identity vectors x, ω or the intermediate results $x_i \cdot \omega_i$ is revealed to anyone.

resulting *randomized sum* is sent to the server. The server de-randomizes the sum to obtain the final result, which is compared with a threshold for authentication.

As we described before, both the user (or client) and the server do not trust each other with the biometric and the claimed identity. While the enrollment is done by a trusted third party, the authentications can be done between the client and the server directly. The client has a biometric sensor and some amount of computing power. The client also possesses an RSA private-public key pair, E and D. We will now describe the authentication and enrollment protocols in detail.

3.3.1 Authentication

We note that the computation of: $\omega \cdot x$ requires a set of scalar multiplications, followed by a set of additions. As the encryption used (RSA) is homomorphic to multiplication, we can compute, $E(\omega_i x_i) = E(\omega_i)E(x_i)$, at the server side. However, we cannot add the results to compute the authentication function. Unfortunately, sending the products to the client for addition will reveal the classifier parameters to the user, which is not desirable. We use a clever randomization mechanism that achieves this computation without revealing any information to the user. The randomization makes sure that the client can do the summation, while not being able to decipher any information from the products. The randomization is done in such a way that the server can compute the final sum to be compared with the threshold. The overall algorithm of the authentication process is given in Algorithm 2. Note that all the arithmetic operations that we mention in the encrypted domain will be $modulo-$ operations, i.e. all the computations such as (a *op* b) will be done as (a *op* b) mod p, where p is defined by the encryption scheme employed.

Algorithm 2 Authentication

```
1:  Client computes feature vector, $x_{1..n}$, from test data
2:  Each feature $x_i$ is encrypted ($E(x_i)$) and sent to server
3:  Server computes $kn + k$ random numbers, $r_{ji}$ and $\lambda_j$, such that, $\forall_i, \sum_{j=1}^{k} \lambda_j \, r_{ji} = 1$
4:  Server computes $E(\omega_i \, x_i \, r_{ji}) = E(\omega_i) \, E(x_i) \, E(r_{ji})$
5:  The $kn$ products thus generated are sent to the client
6:  The client decrypts the products to obtain: $\omega_i \, x_i \, r_{ji}$
7:  Client returns $S_j = \sum_{i=1}^{n} \omega_i \, x_i \, r_{ji}$ to the server
8:  Server computes $S = \sum_{j=1}^{k} \lambda_j \, S_j$
9:  if $S > \tau$ then
10:     return $Accepted$ to the client
11: else
12:     return $Rejected$ to the client
13: end if
```

In the algorithm, the server carries out all its computation in the encrypted domain, and hence does not get any information about the biometric data (x) or the classifier parameters (ω). A malicious client also cannot guess the classifier parameters from the products returned as they are randomized by multiplication with r_{ji}. The reason why the server is able to compute the final sum S in *Step 8* of Algorithm 2 is because we impose the following condition on r_{ji}s and λ_js during its generation:

$$\forall_i, \sum_{j=1}^{k} \lambda_j \, r_{ji} = 1 \tag{3.1}$$

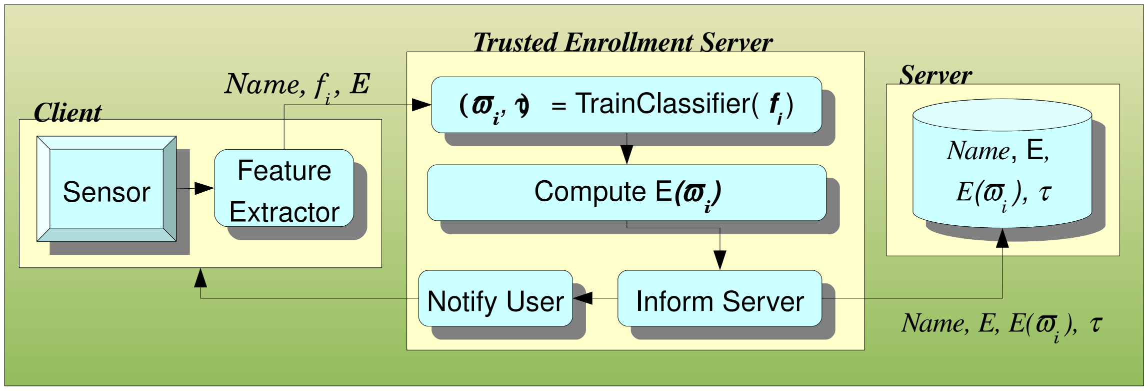
The privacy is based on the ability of the server to generate random numbers using a random number generator (PRNG). The λ_j and r_{ji} are generated using PRNG while ensuring that the Equation: 3.1 holds. This means that all but the last row of the r_{ji} and the corresponding λ_j are truly random. The last row of r_{ji} and λ_j are generated so as to satisfy the Equation: 3.1.

Substituting the above equality in the expansion of the final sum (S) in Algorithm 2, we get:

$$\begin{aligned} S &= \sum_{j=1}^{k} \lambda_j \, S_j = \sum_{j=1}^{k} \lambda_j \, \sum_{i=1}^{n} \omega_i \, x_i \, r_{ji} && (3.2) \\ &= \sum_{i=1}^{n} \sum_{j=1}^{k} \lambda_j \, \omega_i \, x_i \, r_{ji} && (3.3) \\ &= \sum_{i=1}^{n} \omega_i \, x_i \sum_{j=1}^{k} \lambda_j \, r_{ji} \; = \sum_{i=1}^{n} \omega_i \, x_i \end{aligned}$$

We note that the server is unable to decipher any information about the original products, and directly obtains the final sum-of-products expression. This quantity measures the confidence that the test biometric belongs to the claimed identity, and does not reveal any information about the actual biometric itself. The authentication process thus maintains a clear separation of information between the client and the server and hence provides complete privacy to the user, and security to the biometric. Moreover, the clear biometric or parameters are never stored at any place, thus avoiding serious losses if the server or the client computer is compromised. We will take a detailed look at the related security aspects in Section 3.4. The extension of this approach to compute more complex functions such as the kernelized inner products are given in Section 3.5. One can also deal with variable length features and warping based matching techniques using a similar approach. However, a complete treatment of such solutions are beyond the scope of this thesis. We now look at the enrollment phase of the protocol.

3.3.2 Enrollment



Trusted Enrollment Server
Name, f_i, E
Client
Sensor
Feature Extractor
(ϖ_i, ϑ) = TrainClassifier(f_i)
Compute E(ϖ_i)
Notify User
Inform Server
Server
Name, E, E(ϖ_i), τ
Name, E, E(ϖ_i), τ

Figure 3.6 Enrollment based on a trusted third party(TTP): At the time of registering with a website, the encrypted version of the user's biometric template is made available to the website. The one-time classifier training is done on the plain biometrics, and hence requires a trusted server to handle training.

In the previous section, we assumed that server has copies of the clients public key, E, as well as the classifier parameters that are encrypted using that key, $E(\omega_i)$. These were sent during the enrollment phase by a trusted enrollment server. Assuming a third party as the enrollment server gives us a flexible model, where the enrollment could also be done by the client or the server if the trust allows.

During the enrollment, the client sends samples of her biometric to the enrollment server, who trains a classifier for the user. The trained parameters are encrypted and sent to the authentication server, and a notification is sent back to the client. Fig 3.6 gives an overview of the enrollment process. The biometric samples sent by the client to the enrollment server could be digitally signed by the client and encrypted using the servers public key to protect it.

The use of a third party for enrollment also allows for long-term learning by the enrollment server over a large number of enrollments, thus improving the quality of the trained classifier. Algorithm 3 gives a step-by-step description of the enrollment process. Note that the only information that is passed from the enrollment server to the authentication server is the users identity, her public key, the encrypted versions of the parameters, and a threshold value.

Algorithm 3 Enrollment

```
1: Client collects multiple sample of her biometric, $B_{1..k}$
2: Feature vectors, $x_i$, are computed from each sample
3: Client sends $x_i$, along with her identity and public key, $E$, to the enrollment server
4: Enrollment server uses $x_i$ and the information from other users to compute an authenticat-
   ing classifier $(\omega, \tau)$ for the user
5: The classifier parameters are encrypted using the users public key: $E(\omega_i)$
6: $E(\omega_i)s$, along with the user's identity, the encryption key ($E$), and the threshold ($\tau$), are
   sent to the authentication server for registration
7: The client is then notified about success
```

3.3.3 Applicability

We have not made any assumptions on the specific biometric being used in the framework. One could use any biometric as long as the feature vector embeds the samples in a Euclidean space. The classifier itself was assumed to be a linear classifier. However, one can extend it to work with kernel based methods (explained in Section 3.5) and hence any verification problem that can be carried out using a generic SVM-based classifier can be modeled by this protocol. We also sketch an extension of the protocol that works with the Neural Networks in Section 3.5.

3.4 Security, Privacy, and Trust in Blind Authentication

Security of the system refers to the ability of the system to withstand attacks from outside to gain illegal access or deny access to legitimate users. Since we are dealing with insecure networks, we are primarily concerned with the former. Security is hence a function of the specific biometric used as well as the overall design of the system. In terms of information revealed, security is related to the amount of information that is revealed to an attacker that would enable him to gain illegal access.

Privacy on the other hand is related to the amount of user information that is revealed. Ideally, one would like to reveal only the identity and no additional information. Most of the current systems provide very little privacy, and hence demands trust between the user and the server. An ideal biometric system would ensure privacy and hence need not demand any trust, thus making it applicable in a large set of applications. We now take a closer look at the security and privacy aspects of the proposed system.

3.4.1 System Security

Biometric systems are known to be more secure as compared to passwords or tokens, as they are difficult to reproduce. As the authentication process in the proposed system is directly based on biometrics we gain all the advantages of a generic biometric system. The security is further enhanced by the fact that an attacker needs to get access to both the user's biometric as well as her private key to be able to pose as an enrolled user.

3.4.1.1 Server Security

We analyze the security at the server end using two possible attacks on the server:

Case 1: *Hacker gains access to the template database.* In this case, all the templates (or classifier parameters) in the server are encrypted using the public key of the respective clients. Hence gaining access to each template is as hard as cracking the public key encryption algorithm. Moreover, if by any chance a template is suspected to be broken, one could create another one from a new public-private key pair. As the encryption's are different, the templates would also be different. Brute-force cracking is practically impossible if one uses a probabilistic encryption scheme, even for limited-range data.

Case 2: *Hacker is in the database server* during *the authentication.* In such a situation, the hacker can try to extract information from his entire "view" of the protocol. Specifically, the view consists of the following five components:

1. Encrypted values of all ω_i's, that is $E(\omega_i)$, $i \in [1, n]$;

2. Encrypted values of all x_i's, that is $E(x_i)$, $i \in [1, n]$;

3. All the random values used in the protocol, that is all the r_{ji}'s, $i \in [1, n]$ and $j \in [1, k]$;

4. All the λ_j's, $j \in [1, k]$; and

5. All intermediate sums: $S_j = \left(\sum_{i=1}^{n} \omega_i x_i r_{ji}\right) \% N$ for all $j \in [1, k]$.

We ask, what can the hacker learn about the critical data, viz., ω_i's and x_i's? Note that the hacker only obtains k linear congruences over the n variables $y_1, y_2, \ldots, y_n$, namely, $S_j = \left(\sum_{i=1}^{n} r_{ji} y_i\right) \%N$ for all $j \in [1, k]$, where $y_i = \omega_i x_i$. Even though this may reveal some information about y_is, it is impossible to recover the original biometric, as it requires $|\mathbb{Y}|^{n-k}$ authentication trials ($|\mathbb{Y}|$ is domain of y_i's), each involving the help of the client and his private key. We now show that the amount of effort required in doing this is at least as much as randomly guessing the original biometric, and hence no additional information is revealed in principle.

Let $\mathbb{X}$ be the domain of x_i's and let $\mathbb{D}$ be the domain of r_{ji}'s. Without loss of generality, we assume that $\mathbb{D} \supset \mathbb{Y} \supset \mathbb{X}$, and all computations in the authentication protocol are done over the finite domain $\mathbb{D}$.

The number of authentication trials required in a brute-force attack of x_is is $O(|\mathbb{X}|^n)$, which is transformed to $O(|\mathbb{Y}|^{n-k})$ when the k linear congruences are revealed. We want to ensure that $|\mathbb{Y}|^{n-k} \geq |\mathbb{X}|^n$. That is, $ln(|\mathbb{Y}|) \geq \frac{n}{n-k} ln(|\mathbb{X}|)$. Solving this, we get:

$$k \leq n \left(1 - \frac{ln(|\mathbb{X}|)}{ln(|\mathbb{Y}|)}\right), \text{ or } \frac{ln(|\mathbb{X}|)}{ln(|\mathbb{Y}|)} \leq 1 - \frac{k}{n}. \tag{3.4}$$

We note that $|\mathbb{Y}|$ is around $|\mathbb{X}|^2$ as $y_i = x_i \omega_i$, which results in $k \leq n/2$ for complete privacy. As the minimum value of k that is required by the protocol is 2, we find that $2 \leq k \leq n/2$. Choosing a lower value of k will enhance security further, but increase the required $|\mathbb{D}|$.

Case 2.1: *If the hacker is in the server over multiple authentication trials of the same user*, then he will have multiple sets of k linear congruences to infer the values of y_i. However, note that the values of x_i will change slightly over multiple authentications, which gets reflected in the values of y_i. Now the hacker's problem is to compute an approximate estimate of y_i from his view of congruences over noisy y_is, which we call y_i'. Let $\varepsilon_i \in \mathbb{E}$ be the noise between the two instances of x_i. From linear algebra, we know that every additional set of k linear congruences will reduce the brute-force attack complexity by $O|\mathbb{Y}|^k$. Thus, it seems like after a certain number of authentication trials, a hacker will have sufficient congruences to uniquely solve for the n variables. However, we now show that even this is not possible, as during each authentication trial, the hacker not just obtains k additional equations but also ends up adding n new variables.

The hacker obtains k new equations in y_i'. As $y_i' = \omega_i(x_i + \varepsilon_i) = y_i + \omega_i \varepsilon_i$, this can be thought of as k new equations in y_i along with n new unknowns $\omega_i \varepsilon_i$. The domain of these new variables is $|\mathbb{E}|.|\mathbb{X}| \geq |\mathbb{X}|$. To ensure complete privacy, one has to make sure that the information gained by the additional k equations is less than the uncertainty introduced by the new n variables. That is, we need to ensure that $|\mathbb{Y}|^k \leq |\mathbb{X}|^n$. We also know, $|\mathbb{Y}|$ is around $|\mathbb{X}|^2$, thus we have to ensure that $|\mathbb{X}|^{2k} \leq |\mathbb{X}|^n$. This condition holds when $k \leq \frac{n}{2}$, which is true for any choice of k from the previous case. Thus, in spite of the view of multiple authentication trials, the hacker gets no additional information about the biometric.

Our scheme assumes that the server runs the delegated code faithfully. If the server is malicious, it can try to learn additional information about the client's biometric by using a selected vector (say unit vector in a direction) instead of the template for the product. However, the client can detect this using an input, whose result is known. For example, the client can randomly send a vector, which is known to be authentic (not authentic), and check if the the server accepts (rejects) it. Another option would be to use a probabilistic encryption scheme for the template, and keep the randomness in the encryption, a secret, as the server never needs to decrypt any data. In this case, the server will not be able to use any data other than the temple provided for computations.

Case 3: *Impostor trying blind attacks from a remote machine.* It is clear that a brute force attack will have a complexity of the product of that of the plain biometric and the private key. However, note that in the final step, the computed confidence score S is a linear combination, and is compared with a threshold. Hence, if the impostor replaces the partial sums S_js with random numbers, he might be able to pass the confidence test without knowing anything about the biometric or the private key. Also note that the probability of success in this case could be very high. However, a simple modification of the protocol at the server side could thwart this attack. The server could multiply all the sums with a random scale factor, sf, and check if the returned sum is a multiple of sf or not. From his view, the impostor cannot learn sf as GCD is not defined for congruences.

In short, we see that the server is secure against any active or passive attack, and will not reveal any information about the classifier or the user's biometric.

3.4.1.2 Client Security

Case 4: *Hacker gains access to the user's biometric or private key.* Our protocol captures the advantages of both the biometric authentication as well as the security of the PKC. If the attacker gets hold of the user's biometric from external sources, he would also need the private key of the user to be able to use it. If only the private key of a user is revealed, the security for the effected individual falls back to that of using the plain biometric. Note that in practice, the private key is secured by storing it in a smart card, or in the computer using a fuzzy vault. In short, an impostor need to gain access to both the private key and the biometric to pose as a user. Even in this case, only a single user will be affected, and replacing the lost key would prevent any further damages. In practice, periodic replacement of the private key is advisable as in any PKC-based system.

Case 5: *Passive attack at the user's computer.* In this case, the hacker is present in the user's computer during the login process. As the private key can be secured in a hardware which performs the encryption, the hacker will not have direct access to the private key. In other words, he will only learn the intermediate values of the computations. The hackers view will consist of kn quadratic congruences: $y_i r_{ji}, i \in [1, n], j \in [1, k]$ He further knows that there exists k λ_is that satisfy n congruences: $\sum_j \lambda_j r_{ji} \% N = 1$. Thus he has $kn + n$ quadratic congruences in $kn + n + k$ variables. This, as in case 2, results in an effort equivalent to a brute force attack. However if the hacker can stay in the user's computer over multiple authentications, then at some point of time, he will have sufficient number of congruences to solve for y_is (see case 2). Note that y_is does not reveal any useful information about the classifier. Moreover, any partial information gained is of no use as an authentication cannot be performed without access to the private key.

Note that an active attack in this case is identical to that of case 3, and the hacker does not know the private key.

3.4.1.3 Network Security

An insecure network is susceptible to snooping attacks. We consider the following attack scenarios:

Case 6: *Attacker gains access to the network.* An attacker who may have control over the insecure network can watch the traffic on the network, as well as modify it. The confidentiality of the data flow over the network can be ensured using the standard cryptographic methods like symmetric ciphers and digital signatures. Furthermore, all the traffic on the network are encrypted either using the clients public key or using the random numbers generated by the server. Hence, even if successfully snooped upon, the attacker will not be able to decipher any information. A replay attack is also not possible as the data communicated during the second round of communication is dependent on the random numbers generated by the server.

3.4.2 Privacy

Privacy, as noted before deals with the amount of user information that is revealed to the server, during the process of enrollment and authentication. We noted that there are two aspects of privacy to be dealt with:

1. *Concern of revealing personal information:* As the template or test biometric sample is never revealed to the server, the user need not worry that the use of biometrics might divulge any personal information other than her identity.

2. *Concern of being tracked:* One can use different keys for different applications (servers) and hence avoid being tracked across uses. In fact, even the choice biometric or real identity of the user itself is known only to the enrolling server. The authenticating server knows only the user ID communicated by the enrollment server and the biometric is obtained in the form of an encrypted feature vector.

As the user and server need not trust each other, the framework is applicable to a variety of remote and on-site identity verification tasks. Moreover, we note that there is no delegation of trust by the server to a program or hardware at the user's end, thus making it applicable to a variety of usage scenarios.

3.5 Extension to Kernels and other Variations

Even though the linear classifier model can support some of the simple template matching approaches, it does not generalize to other model based classifiers. In the following subsections we will show the extensions for the proposed approach to deal with *a)* the kernel form of the linear classifier, the support vector machine (SVM), *b)* the neural networks, and *c)* the possible usability and the security extensions.

3.5.1 Kernel-based classification:

In the linear case, we described a procedure, $secureProduct$, to compute the inner product of two encrypted vectors without revealing its contents. However, in order to use a kernel based classifier at the server for verification, one needs to compute a discriminating function of the form:

$$S = \sum_{i=1}^{N} \alpha_i d_i \kappa({v_i}^T x) = \alpha \cdot \kappa(v, x), \tag{3.5}$$

where the rows of v are the support vectors and $\kappa()$ is referred to as the kernel function.

We first describe a simple extension of the $secureProduct$ procedure to deal with kernel based classification. We note that the parameter of the kernel function is a set of inner products of vectors. This could be calculated in a similar fashion as the regular blind authentication (using $secureProduct$). Once we obtain the individual inner products, we can compute the kernel functions, κ, at the server side. The discriminant function to be computed is once again the dot product of the vector of κ values and the α vector. This could again be computed, securely using the $secureProduct$ procedure. We note that this procedure allows us to compute any kernel function at the server side.

The above approach is more generic and secure than any of the secure authentication protocols in the literature. Moreover, it does not reveal any information about the classifier to the client. However, as the results of the intermediate inner products are known to the server, this simple extension is not completely blind in the information theoretic sense. This can be solved using another round of communication with the client and define a completely blind kernel-based verification protocol (as explained below).

Let the kernel function be $\kappa(v, x)$. Without loss of generality, we can model $\kappa()$ as an arithmetic circuit consisting of add and multiplication gates over a finite domain. Consider two encryption functions: E^* and E^+, which are multiplicative and additive homomorphic [55, 102, 111], respectively. The client

Algorithm 4 $E^+(\mu)$ to $E^*(\mu)$

```
1: Initial State: The server has $E^+(\mu)$, and client has the corresponding private key.
2: The server chooses a random prime number $r$, and computes $E^+(\mu r)$ using repeated addition. This can be efficiently done in $O(log(r))$ additions using the well-known doubling technique.
3: The server sends $E^+(\mu r)$ to the client, who decrypts it to obtain $\mu r$, which reveals nothing about $\mu$.
4: The client then computes $E^*(\mu r)$ and sends this back to the server.
5: The server computes $E^*(\mu)$ by multiplying $E^*(\mu r)$ with $E^*(r^{-1})$.
```

has the private keys of both, while the public keys are available to the server also. We show that one can securely execute such a circuit using interaction between the server and the client. One can perform *addition* operations using $E^+()$ encrypted operands and *multiplication* operations using $E^*()$ encrypted operands, securely. The only cases of concern are when the operands of *multiplication* are in $E^+()$ and vice-versa. We show that if the server has $E^+(\mu)$ (encrypted using the public key of the client), it can convert it into $E^*(\mu)$ using one round of interaction with the client, without revealing μ to the client or the server. The details of the process are given in Algorithm 4.

Similarly, one may also want to convert $E^*(\mu)$ to $E^+(\mu)$. This is possible as explained in Algorithm 5. The above conversion procedures (described by Algorithms 4, 5) along with the secure product protocol (Algorithm 2) is sufficient for blind computation of any kernel based function such as radial basis function networks(RBFs). The computed confidence score S, is then compared by the server against the threshold τ to authenticate a user.

Algorithm 5 $E^*(\mu)$ to $E^+(\mu)$

```
1: Initial State: The server has $E^*(\mu)$, and client has the corresponding private key.
2: The server chooses a random prime number $r$, and computes $E^*(\mu r)$.
3: The server sends $E^*(\mu r)$ to the client, who decrypts it to obtain $\mu r$, which reveals nothing about $\mu$.
4: The client then computes $E^+(\mu r)$ and sends this back to the server.
5: The server computes $E^+(\mu)$ by repeatedly adding $E^+(\mu r)$, $r^{-1}$ times. This can be efficiently done in O(log($r^{-1}$)) additions using the well known doubling technique.
```

For example, consider a polynomial kernel, $\kappa(v, x) = ({v_i}^T \cdot x)^p$, that is to be securely computed in our setting. Initially, the server has access to the encrypted feature vector $\vec{x}$ and the encrypted support vectors $\vec{s}v_k$. The initial . encryption scheme is assumed to be multiplicative homomorphic. Now, computing the kernel value requires both addition and multiplication operations among the support vectors and the feature vector. Utilizing the switch encryption protocols 4 and 5, the polynomial kernel can be computed by using two rounds of switch operations per support vector. The final confidence score

S is computed using the secure dot product protocol 2. The complete protocol to securely compute a polynomial kernel is shown in Figure 3.7.

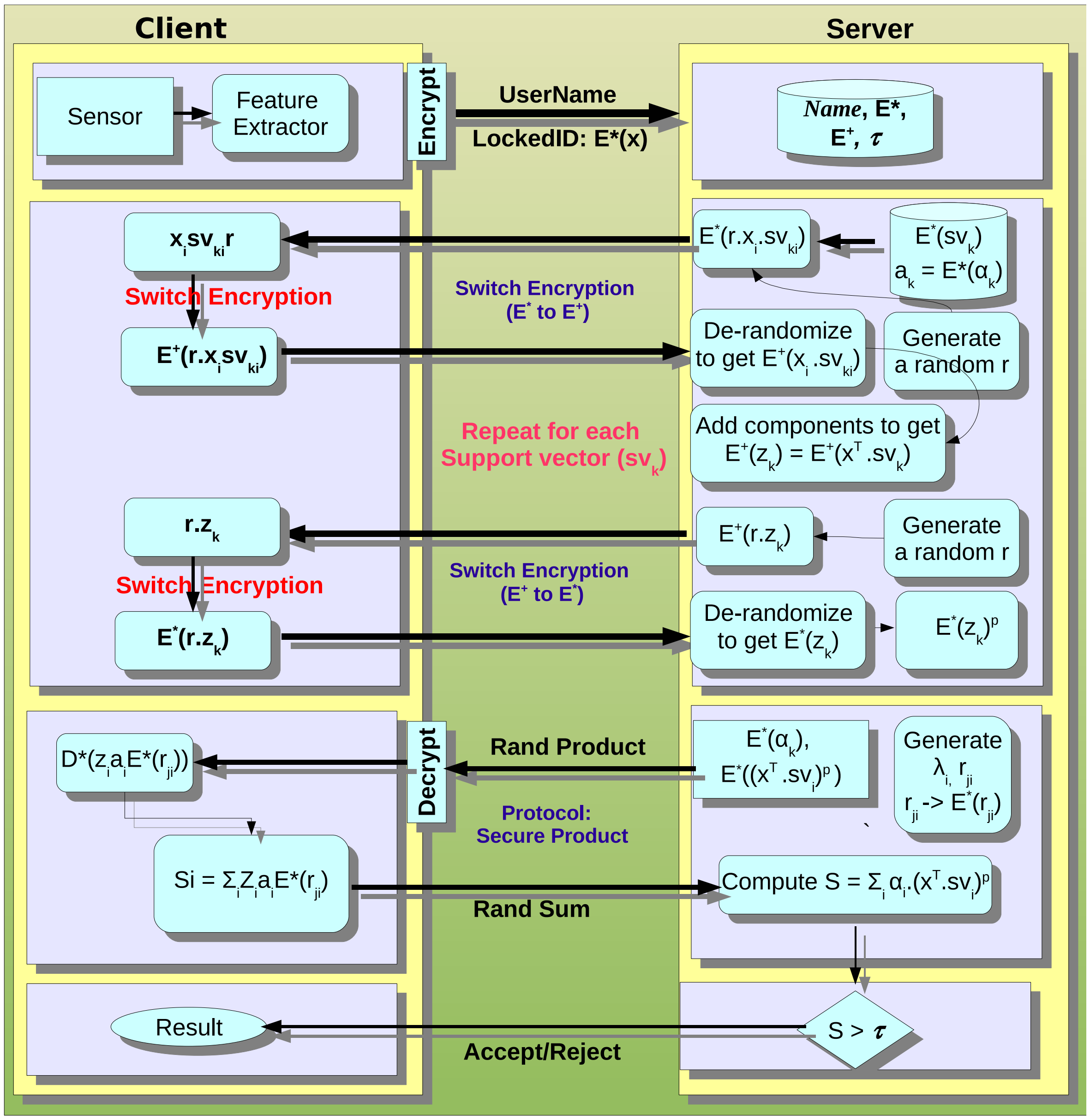


Figure 3.7 Blind authentication process for a polynomial kernel.

In general, the computed confidence score may be considered as an input to a new classifier. For example, in neural networks, the output at one layer is passed as input to the next layer. In such scenarios, one may wish to keep the server oblivious of the computed score S. Thus, we define a *Blind Secure Product Protocol*, Algorithm 6, that computes only the encryption of the score S.

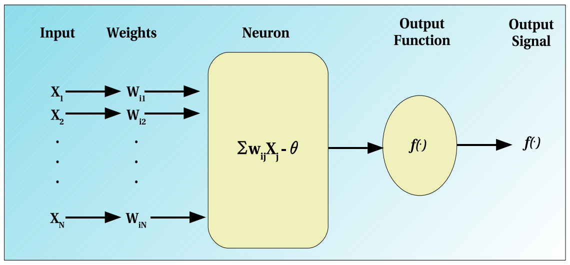
Algorithm 6 Blind Secure Product Protocol

```
1: Initial State: The server has $E^*(\omega)$, $E^*(x)$ received from the client.
2: Server computes $kn + k$ random numbers, $r_{ji}$ and $\lambda_j$, such that, $\forall_i$, $\sum_{j=1}^{k} \lambda_j \, r_{ji} = 1$
3: Server computes $E(\omega_i \, x_i \, r_{ji}) = E(\omega_i) \, E(x_i) \, E(r_{ji})$
4: The $kn$ products thus generated are sent to the client
5: The client decrypts the products to obtain: $\omega_i \, x_i \, r_{ji}$
6: Client computes $S_j = \sum_{i=1}^{n} \omega_i \, x_i \, r_{ji}$
7: $S_j$ is encrypted using $E^+$ and $E^+(S_j)$ is send over to the server.
8: Server computes $E^+(S) = \sum_{j=1}^{k} \sum_{i=1}^{\lambda_j} E^+(S_j)$, this can be efficiently computed using the well known doubling technique.
```

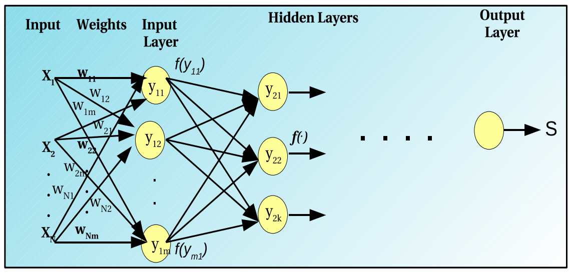
3.5.2 Neural Network based classification

The generalization and approximation provided by Neural Networks have presented them as a practical method for learning real-valued, discrete-valued and vector-valued functions. ANN learning is well-suited to problems in which the training data corresponds to noisy, complex sensor data, such as inputs from cameras [95], thus making them ideal candidate for applications in biometric classification/verification.

Over the years a large number of methods based on Neural Networks has been proposed for biometric verification [39, 54, 59, 97]. In this section, we show how our proposed protocol is generic enough to blindly and securely evaluate a neural network.



(a)



(b)

Figure 3.8 *a)* A typical processing unit used as a node in ANN. A weighted summation of the input is computed, result of which is then used to computed the output function $f()$, *b) A Typical Multilayer Neural Network.*

Threshold and Sigmoid are the two most popular type of basic units used in ANN. A perceptron is same as the linear classifier discussed in Section 3.3. It takes a vector of real-valued inputs, calculates a weighted summation of these inputs and outputs a 1 if result is greater than the threshold and -1 otherwise. Algorithm 7 describes the completely blind perceptron computation.

$$S = sgn(y) = \begin{cases} 1 & \text{if } y \geq 0 \\ -1 & \text{otherwise} \end{cases} \tag{3.6}$$

Algorithm 7 Blind Threshold Function Computation

```
1: Initial State: The server has $E^*(\mu)$, $E^*(x)$ received from the client. Server to compute $E^*(t)$, where t = 0/1 depending on threshold.
2: After a round of *Blind Secure Product Protocol* [Algo: 6], the server obtains $E^+(\mu^T.x - \alpha)$
3: Server generates a random number $r$ and computes $E^*(r(\mu^T.x - \alpha)$ and sends over to the client.
4: Client decrypts the obtained cipher and returns back the encrypted equivalent of sign bit i.e. returns $E^*(d)$ = $E^*(sign(r(\mu^T.x - \alpha)$
5: Server computes $E^*(S)$ = $E^*(d).E^*(sign(r))$
```

Another important/popular basic unit in ANN is the *Sigmoid Unit*. It is based on a smoothed, differential threshold function. The sigmoid unit first computes a linear combination of its inputs, then applies a threshold to the result. The threshold output is a continuous function of its input, Equation 3.7.

$$S = \sigma(y) = \frac{1}{1 + e^{-\alpha.y}} \tag{3.7}$$

The α, in the above equation, is some positive constant that determines the steepness of the threshold. A completely blind Sigmoid function computation is explained in Algorithm 8.

With the solutions already sketched for securely computing both sigmoid and perceptron based neurons, the solution can be easily extended to securely compute multilayer neural networks. A typical multilayer neural network is shown in Fig 3.8 (b).

Every neuron in each of the layers is securely computed using the above algorithms. In the process, the client doesn't learn anything and all that the server gets is the encrypted output of the neuron. This encrypted output of a particular layer of neurons acts as an input to the next layer in the network. The output of the last layer is decrypted and compared against the threshold to authenticate the user.

The above process is completely secure and blind in that at no point does the server or client learns the weights or intermediate results. All computations are done in encrypted domain, and given an encrypted input vector $E^*(x)$ the client learns nothing but the authentication result. A somewhat similar solution was proposed by Orlandi *et al* [101], however, their solution uses only additive homomorphic encryption

Algorithm 8 Blind Sigmoid Function Computation

```
1: Initial State: The server has $E^*(\mu)$, $E^*(x)$ received from the client. Server to compute $E^*(\frac{1}{1+e^{-\alpha.y}})$.
2: After a round of *Blind Secure Product Protocol* [Algo: 6], the server obtains $E^+(y)$
3: $E^+(\alpha.y)$ is computed using repeated additions.
4: Server chooses a random $r$ and sends to client $E^+(r + \alpha.y)$ = $E^+(r).E^+(\alpha.y)$.
5: Client decrypts the obtained cipher to get $r + \alpha.y$, which is used to compute $E^*(e^{r+\alpha.y})$ and is sent back to the server.
6: Server multiplies the obtained result with $E^*(e^{-r})$ to get $E^*(e^{\alpha.y})$.
7: Switch encryption and add $E^+(1)$ to obtain $E^+(1 + e^{\alpha.y})$
8: Server chooses a random $r = \frac{r_1}{r_2}$, such that $r^{-1}$ exists. Use repeated additions to obtain, $E^+(r_1.e^{\alpha.y})$ and $E^+(r_2.e^{\alpha.y+1})$. These are then send over to the client.
9: Client decrypts the received ciphers and computes $r.\frac{e^{\alpha.y}}{1+e^{\alpha.y}}$. This is encrypted using $E^*$ and send over to server.
10: Server obtains $E^*(\frac{1}{1+e^{-\alpha.y}})$ by multiplying $E^*(r.\frac{1}{1+e^{-\alpha.y}})$ and $E^*(r^{-1})$.
```

schemes and is therefore not as generic as the one proposed by us. Moreover, their solution assumes the hidden layer weights are available in plain with the server, thus compromising both the security and privacy of the system.

3.5.3 Usability and Security Extensions

One could extend the proposed protocol in a variety of ways to improve the usability and security.

Client side security: The users client module (computer) contains the public and private keys for encryption and decryption. Moreover the client end also contains the biometric acquisition device. To ensure complete security of the system, one needs to consider the security at the client end also. This is especially true, if one is using a public terminal to access any service. The first step in securing the private key is to move it to a card so that the private key is not lost if the client computer is compromised. As a second step one could carry out the decryption operation, completely in a smart card. Revealing the secret keys to an attacker can reduce the overall security of the system to that of a plain biometric authentication system.

One could also secure the secret keys at the client end using a fuzzy vault [79], either in the client's computer or on a card. The biometric that is provided for authentication can also be used to unlock the vault to get the key. The released private key is used for decryption of results in the protocol. The fuzzy vault construct precisely suits this purpose as one could blindly use the keys generated by unlocking the vault for encryption. If the biometric presented is wrong, the encryption will not match the server's keys and hence the authentication will fail. Hence we have a double layer of security through the biometric provided by the user.

Avoiding client-side computation and communication: Another possible extension to the framework is to use the paradigms from secure computing to package the intermediate operations done at the client side into an applet. This applet can now be run securely on the server itself, thus avoiding the overhead of communication, and reducing the computing requirements of the client.

Using different encryption schemes: Note that the RSA is one of the many homomorphic encryption schemes. We could replace this with any of the other similar encryption mechanisms. One could analyze the computation cost and security issues for each encryption method.

Since the information content in each feature (or weight) is expected to be limited and the public key of the client is known, it may be possible for an attacker to decode the encrypted features (weights) using a direct plain-text attack. Similarly in the blind threshold function computation, output of the neuron is either zero or one. To combat this attack, public key encryption schemes must incorporate an element of randomness, ensuring that each plaintext maps into one of a large number of possible ciphertexts. Thus, the encryption scheme $E()$ has to be a function of both the secret x and a random parameter r. Such a scheme is known as *probabilistic encryption*. However, for our purpose, we also need to carry out the computations in the encrypted space, thus the encryption scheme should also be homomorphic. ElGamal [55] and Pailler Encryption [102] are two popular probabilistic homomorphic encryption schemes.

Improving speed of SVM-based classifiers: As described in Section 3.5, the kernel based classifiers need to compute the discriminating function given by Equation 3.5. As can be noticed, the computational costs of computing this is directly proportional to the number of support vectors used. In practice, the number of support vectors that are returned from the training step could be quite large. However, a variety of approaches to reduce the number of support vectors used (without loss in accuracy) for classification has been proposed [14].

3.6 Implementation and Analysis

We have performed several experiments to evaluate the efficiency and accuracy of the proposed approach. An authentication protocol was implemented based on a client-server model that can perform verification over an insecure channel such as the Internet. A variety of public domain datasets are evaluated using an SVM classifier to demonstrate the effectiveness of our proposed protocol. The following experiments and analysis evaluates the accuracy and performance of our method.

3.6.1 Implementation

For the evaluation purpose ans SVM based verifier based on a client-server architecture was implemented in GNU/C. RSA keys were generated using the implementation available through *XySSL* [10] and keys for the Paillier cryptosystem were generated using the *Paillier Library* [28] . All computations were done using the *GNU Multiple Precision Arithmetic Library (GMP)* [5]. All experiments are conducted on AMD X2 Dual Core 4000+ processor, 750MB DDR2 RAM and 100Mbps Intranet.

Both RSA and Paillier cryptosystem have exponentiation based encryption and decryption. Their implementation assumes that the data consists of positive integers. For the homomorphism to hold, we need to map the floating point numbers to positive integers. Hence we scale the feature vectors and the SVM parameters to retain the precision and round off to the nearest integral value. Efficiently handling negative numbers is important to achieve efficiency. The representation chosen should ensure a single representation of zero, obviating the subtleties associated with negative zero. In our implementation, the mathematical library operates at the binary representation level. We use an implicit sign representation to handle negative numbers. If the range of numbers used is $(0, M)$, then we use the numbers in the range $(0, M/2)$ to represent positive numbers, and for the remaining numbers negative. For example: let $M = 256$, then to represent -95 we store -95 $modulo$ 256 which is equivalent to 161 since: $-95 + 256 = -95 + 255 + 1 = 160 + 1 = 161$

If x_i is to be encrypted, the forward mapping is defined as: $x'_i = fwdMap(\lfloor s.x_i + 0.5 \rfloor)$, where s is a scale factor, depending on the range of values for x_is, and $fwdMap()$ maps the integral numbers to the implicit sign representation. The server does the reverse mapping on the obtained results.

In the following sub-sections, we will validate the generality of the protocol by validating classification of various publicly available datasets. We will also analyze how the various parameters i.e. key-size, precision affect the classification accuracy and the verification time. Finally we'll show the validity of SVM's as a classification model for various biometric problems.

3.6.2 Classification Accuracy

As the protocol implements a generic classifier, without making any simplification assumptions, the accuracy of the classifier should be identical to that of the original classifier. One could expect small variations in accuracy due to the round off errors used in the mapping function described above. To verify the effect we compared the classification results using linear and SVM classifiers of 8 different public domain datasets: the *Iris*, *Liver Disorder*, *Sonar*, *Diabetes*, and *Breast Cancer* datasets from the UCI repository and the *Heart* and *Australian* datasets from the Statlog repository. The datasets were selected to cover a variety of feature types and feature vector lengths. Table 3.2 describes the datasets and the accuracy obtained using a polynomial kernel with precision set as 4. On these datasets, the classification results remained identical even though there were minor variations in the computed discriminant values.

The above accuracies were cross checked by re-classifying the datasets with the same parameters by the well known SVM classification library SVM^{light} [78]. Figure 3.9 shows the verification time for a

Dataset	Number of Features	Number of Instances	Accuracy (%)
Iris [UCI]	4	150	100
Heart [Statlog]	13	270	90
Liver Disorder [UCI]	6	345	68
Sonar [UCI]	60	208	51.47
Australian [Statlog]	14	690	86.49
Diabetes [UCI]	8	768	76.37
FourClass [Tin Kam Ho]	2	862	69.20
Breast Cancer [UCI]	10	683	89.80

Table 3.2 Classification results on various datasets using a SVM classifier. The accuracies were compared to the corresponding plain domain classifier and was found to be identical.

linear classifier w.r.t. various RSA key-sizes and feature vector lengths. A more detailed analysis of the computational time for the protocol is given in Section 3.6.4.

Figure 3.10 shows how the overall accuracy is affected by changing the precision. For the considered datasets, the feature vectors were first normalized to range -1 to 1 and then scaled to retain a certain precision. When precision is set to less than 2, a lot of feature vectors having feature values of the order of 10^{-3} or less, mapped to a value of zero, thus affecting the accuracy. For the above datasets, we note that a precision of 3 or more results in stable results and the accuracies do not change with any further increase in precision. Thus for our experiments we set precision as 4. Note: precision doesn't affect the computational time, as all the numbers are represented using a fixed length bit representation.

The above set of experiments demonstrate the applicability of our protocol to the SVM based classification problems. We showed that one can achieve the accuracies of SVM's even in an encrypted domain and at the same time obtain heightened security at some computational expense.

3.6.3 Biometric Verification

We have presented a protocol to securely classify data using Support Vector Machines and Neural Networks (Section 3.5). The primary limitation of the protocol in its current form is its restriction to fixed length feature vector representation of the data (Section 3.3). This might raise a concern as to how efficient are fixed length feature vector representation with respect to biometric verification problems.

To address the above concern, we conducted a case-study of the state of art results obtained for various biometric modalities using both fixed length and variable length feature vector representations. Table 3.3 summarizes the primary findings of the literature survey. As can be seen from the comparison, the accuracies of the fixed length feature vector based biometric verification approaches are comparable to those using variable length feature vectors and matching techniques such as dynamic warping.

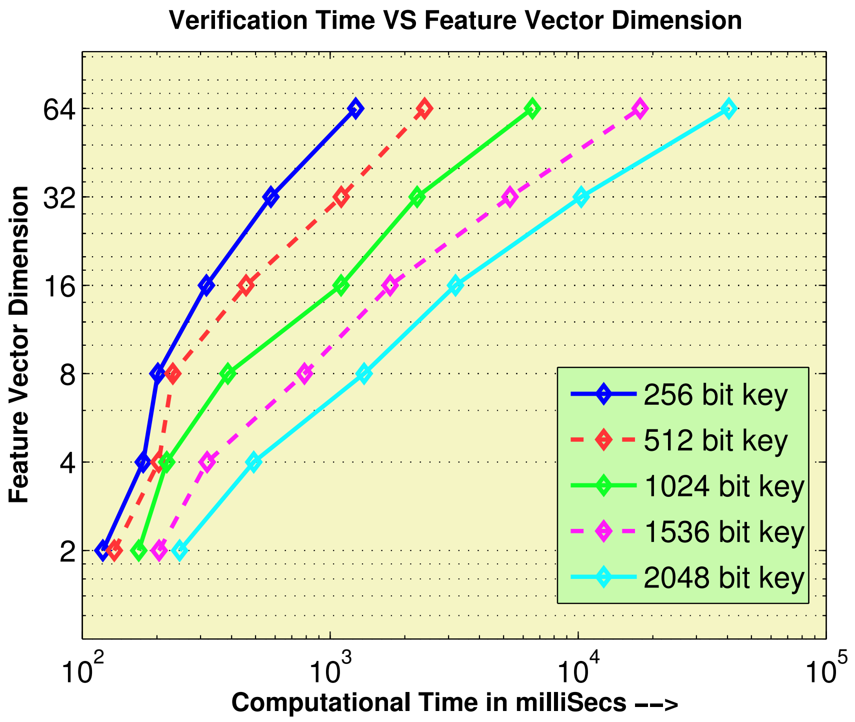


Figure 3.9 Verification time for various key sizes and feature vector lengths.

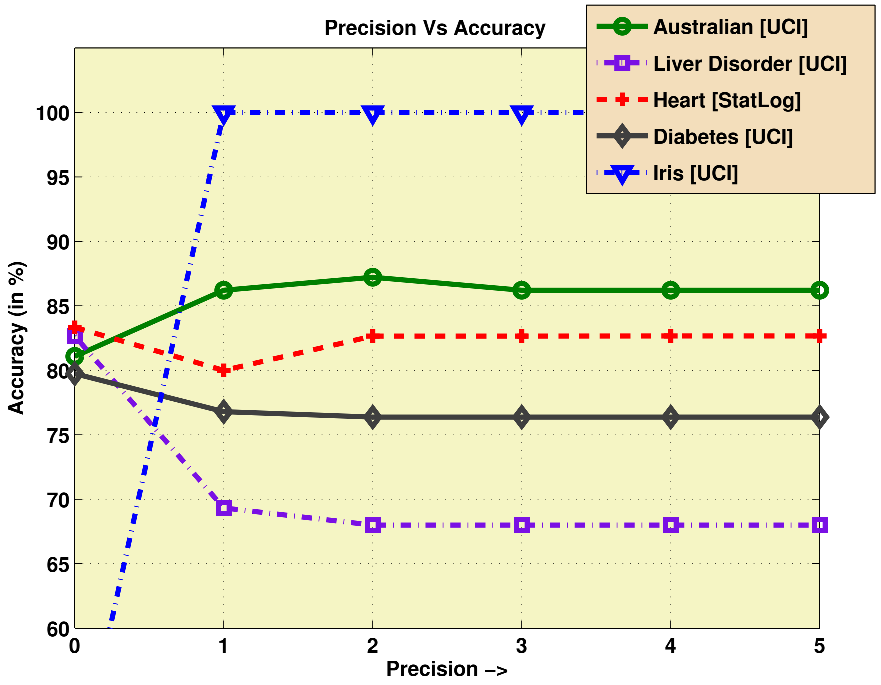


Figure 3.10 Variation of accuracy w.r.t. the precision of representation.

Paper	Feature Set	*Matching Method*	*F.D.*	*DataSet*	*GAR/FAR*	*EER*
Finger Print						
Xu *et al.* [141]	Invariant spectral minutiae set	Spectral minutiae matching	F	MCYT Biometric DB [56]	-	3.21%
Yang *et al.* [142]	7 invariant moment features	LVQ Neural Network	F	FVC2002 DB1 [3]	95.1% / 0.5%	-
Sha *et al.* [117]	Minutiae representation based on ridge pattern	Ridge count matching and minutiae subset combination.	V	NIST-4 [6]	97% / 0.1%	-
Kisel *et al.* [83]	Graph based local structure representation of minutiae	Correspondence set construction and similarity score computation	V	FVC2002 DB1 [3]	96.71% / 0.01%	-
Zsolt [85]	Minutiae set	Triangular matching and DTW.	V	NIST-4 [6]	85% / 0.05%	-
Hand Geometry						
Kumar *et al.* [86]	23 hand-geometry features, discretized using entropy based heuristics	SVM, Neural Network	F	100ppl, 10 images per user	-	1.9%
Marcos *et al.* [**?**]	10 hand-geometry features	Neural Networks	F	50ppl, 10 images per user	99% (avg perf)	1%*
Vit *et al.* [107]	Time series repr of hand geometry	DTW similarity measure.	V	22 ppl, 6-7 images per user	98.25%(TSR)	1.75%*
Face						
Guo *et al.* [66]	Eigenfaces	Support Vector Machine (SVM)	F	ORL face db [7]	-	3.0%
Heisele et al. [67]	Gray values of facial components	Support Vector Machine (SVM)	F	Internal	95% / 5%	-
Wiskott *et al.* [140]	Face Bunch Graph	Elastic Bunch Graph Matching (EBGM)	V	FERET [2]	98% (frontal faces)	2%*
Blanz *et al.* [32]	3-D morphable face model	Similarity measure between model coefficients	F	FERET [2]	87.9% / 1%	-
Iris						
Roy *et al.* [113]	Gabor wavelet technique used to extract features	Support Vector Machine (SVM) s	F	CASIA Iris Dataset [1]	97.34% (acc)	2.66%*
Monro *et al.* [96]	Iris code represented using DCT coefficients	Hamming distance	F	CASIA Iris Dataset [1]	100% (acc)	0%*
Neagoe [98]	Binary templates	Hamming Self organizing map (HSOM)	F	CASIA Iris Dataset [1]	99.08% (acc)	0.9%*

Table 3.3 Biometric Verification: An overview of fixed (F) and variable (V) length representations.

To verify the effectiveness of using SVMs as a classification model for biometric verification problems, we tested it on four different modalities. The verification accuracies after 3-fold cross validation on each of the datasets is presented in Table 3.4.

- The first set of experiments used Eigen face representation as features on the Yale face dataset [11], consisting of 10 users, with 11 samples for each user. For each experiment 4 samples were used for training and the remaining 7 samples were used for testing.
- For the second set of experiments, we used a hand-geometry data-set that was collected in-house. The data-set consisted of 149 users with 10 hand images each. The features consists of the 14 finger length and width features described by Jain *et al.* [74]. For each experiment 4 images per user were used for training purpose and the remaining 6 were used for testing.
- The third were on the CASIA IRIS database [1]. The Version 1 of the data-set consists of 108 users with 7 images per user (the seven images are collected over two separate imaging sessions). The iris code consists of 9600 binary features. 3 samples per user were used for training and 4 sample per user were used for testing purpose in each experiment.
- The forth and the final data-set used was *Fingerprint Verification Contest 2004 (FVC2004* data-set [4]. The DB2_A data-set consists of 100 users with 8 images per user. 7 invariant moment features are used as the feature vector. 3 images per user are used for training purpose and the remaining 5 used for testing for each experiment.

Dataset	# of Features	Avg num of Support Vectors	Accuracy
Hand Geometry	20	310	98.38%
Yale Face	102	88	96.91%
CASIA Iris	9600	127	98.24%
FVC 2004	7	440	84.45%

Table 3.4 Verification accuracy on biometric datasets.

Figure 3.11 shows the *receiver operating characteristic (ROC)* [60] plots for the biometrics using fixed length representation[1]. The primary objective of the experiments is to demonstrate that making the authentication secure *does not decrease* the accuracy. Hence, one can apply the technique to secure any fixed-length representation of a biometric trait, which is classified using an SVM or Neural Network.

[1]* Yang *et al* [142], **Wang *et al.* [137]

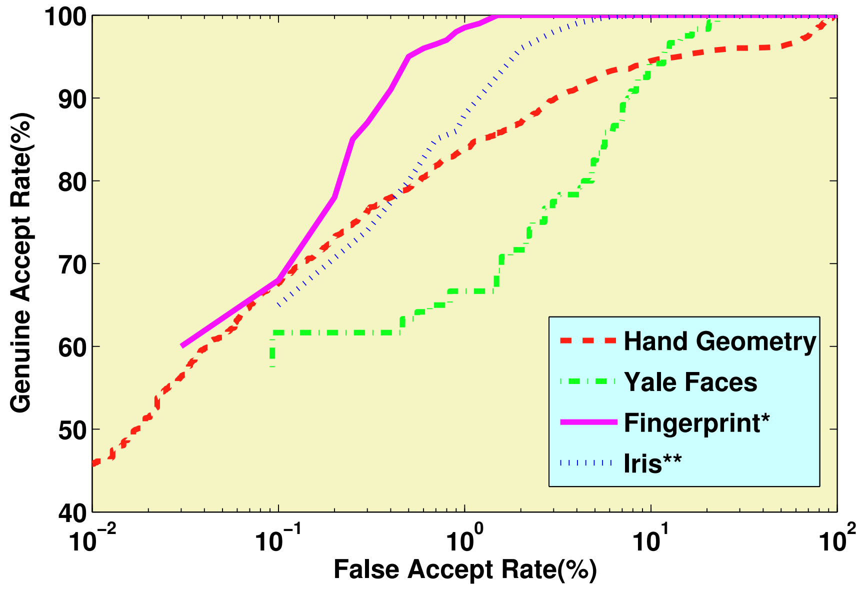


Figure 3.11 ROC Curves for verification

3.6.4 Computation and Communication Overheads

The additional computation that needs to be carried out can be divided into two parts: i) Modulo multiplications to be done for encryption/decryption and inner product, and ii) the additional time spent in the computation of random numbers, products and sums. As the modulo multiplications and encryption decryption operations can be done efficiently using dedicated hardware available [33], we analyze the time required for both, separately. Consider a biometric with feature vector of length n. In the protocol, the client needs to do n encryptions for the test vector x.

For the linear classifier, the server needs to do kn encryptions of the random numbers and $2kn$ multiplications, so as to compute $E(\omega_i x_i r_{ji})$, where k≤n. The client needs to do kn decryptions. Additional computations at the server includes $n + kn$ modulo multiplications of encrypted numbers at the server end, and kn non-encrypted additions at the client end. In addition, the server generates kn random numbers. For most practical biometrics, the total run time required for all these (non-encrypted) computations together on current desktop machines is less than 10 milliseconds. The communication overhead, in addition to regular authentication, includes sending kn numbers from the server to the client and sending k numbers from the client back to the server for evaluation of the final result.

Extending the analysis to a direct kernel based classifier with n_v support vectors (SV), one need to repeat the *secure product* n_v times, once for every SV. Another round of *secure product* computes the final result. Hence the time required will be $n_v + 1$ times that required for the linear classifier. In practice the total time taken (other than those implemented in hardware) is less than one second.

For the completely blind kernel-based protocol, the first phase is the same as the direct kernel extension. However, to achieve complete blindness, we need to do one round of communication to switch encryptions, that will include a kn_v length vector to be sent from the server to the client and back. In the third phase, the computation and communication is identical to that required for a single *secure product*. Hence the total time required will be $n_v + 2$ times that required for the linear classifier.

One could achieve further computational efficiency through support-vector reductions, as well as employing other more computationally fast homomorphic encryption schemes.

3.7 Discussion

The primary advantage of the proposed approach is the ability to achieve classification of a strongly encrypted feature vector using generic classifiers such as Neural Networks and SVMs. In fact, the authentication server need not know the specific biometric trait that is used by a particular user, which can even vary across users. Once a trusted enrollment server encrypts the classifier parameters for a specific biometric of a person, the authentication server is verifying the identity of a user with respect to that encryption. The real identity of the person is hence not revealed to the server, making the protocol, completely blind. This allows one to revoke enrolled templates by changing the encryption key, as well as use multiple keys across different servers to avoid being tracked, thus leading to better privacy.

The proposed blind authentication is extremely secure under a variety of attacks and can be used with a wide variety of biometric traits. Protocols are designed to keep the interaction between the user and the server to a minimum with no resort to computationally expensive protocols such as SMC [143]. As the verification can be done in real-time with the help of available hardware, the approach is practical in many applications. The use of smart cards to hold encryption keys enables applications such as biometric ATMs and access of services from public terminals. Possible extensions to this work includes secure enrollment protocols and encryption methods to reduce computations. Efficient methods to do dynamic warping based matching of variable length feature vectors can further enhance the utility.

Chapter 4

Efficient Privacy Preserving Video Surveillance

Widespread use of surveillance cameras in offices and other business establishments, pose a significant threat to the privacy of the employees and visitors. The challenge of introducing privacy and security in such a practical surveillance system has been stifled by the enormous computational and communication overhead required by the solutions. In this work, we propose an efficient framework to carry out privacy preserving surveillance. In our proposed protocol, we split each frame into a set of random images, each of which is sent to a different non-colluding server for processing. Each image by itself does not convey any meaningful information about the original frame, while collectively, they retain all the information. Our solution is derived from a secret sharing scheme based on the Chinese Remainder Theorem, suitably adapted to image data. Our method enables distributed secure processing and storage, while retaining the ability to reconstruct the original data in case of a legal requirement. The computational requirement at the data source (camera) is very limited, enabling inexpensive monitoring equipment, and the only communication between the camera and the surveillance servers is a compact and encrypted video stream. The system installed in an office or similar environment can effectively detect and track people, or solve similar surveillance tasks. Our proposed paradigm is highly secure and extreamly fast over the traditional SMC, making privacy preserving surveillance practical.

4.1 Introduction

Video surveillance is a critical tool for a variety of tasks such as law enforcement, personal safety, traffic control, resource planning, and security of assets, to name a few. Rapid development/deployment of *closed circuit television (CCTV)* technology is playing a key role in observing suspicious behavior. However, the proliferation in the use of cameras for surveillance has introduced severe concerns of privacy. Everyone is constantly being watched on the roads, offices, supermarkets, parking lots, airports, or any other commercial establishment. This raises concerns such as, watching you in your private moments, locating you at a specific place and time or with a person, spying on your everyday activities, or even implicitly controlling some of your actions.

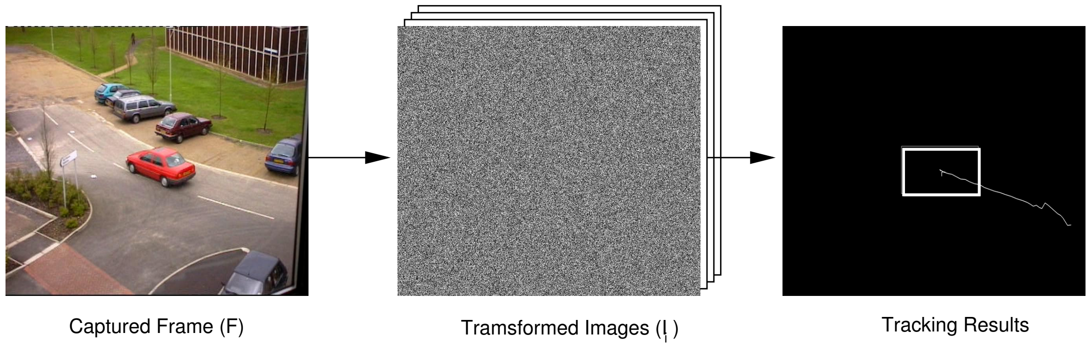


Figure 4.1 Privacy preserving surveillance: Tracking a vehicle in obfuscated surveillance video. Each captured frame *(F)* is uniquely converted into a set of transformed images *I*, each sent to a seperate computational server. Tracking algorithm is now jointly executed by all the servers, such that, at no point, any information of the original video is revealed to any of the servers. At the end of the protocol, all that the observer learns is the final tracking result.

Privacy, therefore, happens to be a serious concern in the Age of Video Surveillance [116]. Widespread usage of surveillance cameras raises the specter of an invasive 'Big Brother' society. In this regards, certain privacy laws have been introduced to guard an individuals privacy/rights. Despite these, video surveillance remains vulnerable to abuse by unscrupulous operators with criminal or voyeuristic aims and to institutional abuse for discriminatory purposes. These legitimate concerns frequently slow the deployment of surveillance systems.

Privacy preserving video surveillance addresses these contrasting requirements of confidentiality and utility. The objective is to allow the general surveillance to continue, without disrupting the privacy of an individual. This novel technology addresses the critical issue of privacy invasion in an efficient and cost-effective way. In practice, for online surveillance to remain meaningful, the protocols are required to be real-time. Furthermore, given the existing and pervasive surveillance infrastruture, the additional costs incurred to accomodate privacy protection in the system needs to be affordable [38].

Ideally, one would like the cameras to generate video streams, which do not convey any useful information by themselves, while providing the ability to run the required surveillance algorithms without any deterioration in performance. For instance, in Figure 4.1 a frame, F, of the surveillance video is transformed into a set of seemingly random images, I_i, on which a tracking operation is successfully carried out. In this work we propose a *"cloud computing"* based solution, that utilizes the services of r, ($r > 2$) non-colluding computational servers. Each of the r transformed images I_i is sent to a different server for processing. This ensures that the original video is not revealed to any of the servers, while together they retain the complete video content. Furthermore, accurate tracking results are obtained since the servers jointly run the original plain-domain tracking algorithm. The solution is not only provably secure but also computationally efficient. The interaction and the data communication among the servers is kept to a minimum and the only processing required of the camera is to generate the transformed images.

A practical and an efficient solution fits into the business model of *'surveillance as a service'*. A company can monitor houses, streets, stores, etc and alert the clients of suspicious incidents, without intruding into their privacy. The client just installs a shatter-cam and sends the feeds for surveillance. Our proposed solution works well with current trends of computing/storing on remote server clouds. This is both economically viable, scalable and provably private, while allowing recovery of original video in case of a crime.

Traditionally, confidentiality of the data is achieved through encryption. However, by definition, encryption destroys any structure present in the data, thus negating the ability to perform any meaningful video processing tasks. Therefore, such solution prevents only eavesdropping, while offering little or no protection against misuse of CCTV video by the authorised personal [38]. Realizing this, recent privacy preserving vision algorithms are build on cryptographic protocols such as *Secure Multiparty Computation (SMC)* [22, 119]. SMC uses interactions between multiple parties to achieve a specific task, while keeping everyone oblivious of other's data.

The problem of introducing privacy and security in visual data processing was addressed with considerable success in different domains. *Smart Cameras* [68] are surveillance cameras equipped with inbuild processing power. In privacy preserving surveillance, smart cameras can be programmed so as to identify and obnubilate the region of interest (such as human face) from the captured video stream before transmitting it over the network (to the observer). Problem specific approaches try to address specific concerns in images and videos. Senior *et al.* [116] presented a model to define video privacy and re-render the video in a privacy-preserving manner. Face swapping [30] and face de-identification [99,122] try to modify face images such that they can be automatically detected, but yet cannot be correctly recognized. Framework proposed by Zhang *et al.* [145] stores the privacy information in surveillance video as a watermark and monitors an invalid person in a restricted area while protecting the privacy of the valid persons. Boult *et al.* [41] has presented a complete surveillance camera with built in processing power that can do tasks such as detection and masking of the regions of interest in the images. Chan *et al.* [40] attempted a related problem of extraction of features from a surveillance video that do not reveal identity of people, while being able to achieve crowd counting and tracking.

In general, smart cameras are designed so as to detect and track suspicious behavior. However these approaches do not provide any guarantee of privacy as they rely on the success of certain computer vision tasks, such as face detection. Furthermore, there are a wide range of possible behavior which may not be part of the suspicious behavior database; in this case smart camera fails to preserve privacy. Moreover, there are also behavioral patterns that may be hard to classify. Alternately smart cameras can be designed to do surveillance in the camera itself. However, this would require expensive programmable cameras and is restricted to single camera algorithms. Changing the algorithms is also tedious and costly.

Provable security/privacy can be guaranteed if the surveillance algorithms can directly run on (cryptographically strong) encrypted video streams. This ensures that the original video stream is hidden at all times and the observer learns only the final output of the surveillance algorithm. The recent research work in this direction uses either of the cryptographic protocols such as SMC [65] or homomorphic encryption schemes [93] [61] or both. The primary goal of Blind Vision [22] [23]; whose security is built upon SMC; is to allow someone to run their classifier on another person's data without revealing the algorithm or gaining knowledge of the data. Shashank *et al.* [119] exploited the clustered nature of image databases to improve the efficiency of SMC for example based image retrieval by processing multiple queries together. As describted in Chapter 3, we utilize the homomorphic encryption schemes to blindly authenticate a biometric sample over the internet.

Perfect privacy can be achieved through approaches based on the generic framework of *SMC* [22] [119] [136] and homomorphic encryptions [101]. However, such solutions would be highly communication intensive. And for the videos, the sheer volume of the data involved makes the protocols infeasible in terms of computation and inter-processor communication costs. For example, computing the dot product of two vectors (length n) using a SMC based protocol would have $O(n^2)$ communication overhead, which requires several milliseconds on a LAN. Clearly, these delays are too high, while dealing with voluminous data like surveillance videos. Hence, solutions based on these cryptographic primitives would be impractical for our desired (real-time) applications.

In this work, we use the paradigm of *Secret Sharing* [118] to achieve privacy and efficient computation of surveillance algorithms. Secret sharing (SS) methods [27] [94] [118] try to split any data into multiple shares such that no share by itself has any meaningful information, but together they retain the complete information of the original data. We exploit certain desirable properties of visual data such as fixed range and insensitivity to data-scale, to achieve distributed, efficient and secure computation of surveillance algorithms in the above framework. Using this approach, one can do change detection on a typical surveillance video at 10 frames/second on generic hardware. Our approach also address the concerns related to video surveillance, presented in Table 4.1.

4.2 Alleviating the Complexity: Our Paradigm

In the last section we gave an overview of the various paradigms explored for privacy preservation. In this section, we reason as to why a paradigm shift is needed to achieve practical privacy preserving surveillance. The traditional methods of securing data is of using strong encryption schemes to secure (lock) the visual data. However, this fails to preserve privacy, since one needs to decrypt (un-lock) the data before processing. In practice, this implies that the server is to be trusted to handle the data

a) Preserves Privacy: Ensure that the person doing surveillance learns nothing but the final output.
b) Computationally Efficient: The encoding process at the sensor should be light weight. And the image representation should allow efficient computations of algorithms in the encrypted domain itself.
c) Efficient to Transmit: The encoding process should not blow up the size of the video data.
d) Secure Storage: One should be able to store the surveillance data in a secure fashion, so that breaking into a storage server do not compromise the privacy of the data.
e) Addresses Legal Issues: For legal or investigative purposes, someone with authorization should be able to recover the plain video without the client's help.
f) Fault Tolerance: Ability to avoid the denial of service(DOS) and single point of failures.

Table 4.1 Mandatory and desirable characteristics of a secure, privacy-preserving surveillance system.

securely, thus making the complete system vulnerable to mis-use by authorized personal in addition to exploitation by hackers.

Selective privacy can be provided with the use of smart cameras. The region of interest (eg: car number plate) can be identified and obfuscated by the camera before streaming the captured CCTV video, over an (un-secure) network, to an observer. However, as already discussed in Section 4.1 these methods do not guarantee complete privacy. Furthermore, defining privacy can be rather subjective, for example, one could de-identify a face but could possibly still identify a person by observing the gait or the characteristics of his/her walk [9].

The primary limitation of the traditional methods was the inablity to carry out meaningful image processing on the encrypted video streams. In order to overcome this, solutions have been proposed using the cryptographic techniques based on secure multiparty computation (SMC) [65]. SMC based solutions allow one to have provable data privacy and at the same time retain the accuracy. SMC originated from the work of Yao [143], who gave a generic solution to privately evaluate 'any' given function (Boolean circuit) that takes private information of 2-parties as input. In other words, SMC facilitates a group of people, each with its own private data, to perform some common computation task on the aggregate of their data without actually revealing, to anyone else, any personal information of their owned data [89].

However, the SMC based protocols are found to be extreamly *computationally expensive* [21]. For video surveillance task, the sheer volume of the data involved makes the protocols infeasible in terms of computation and inter-processor communication costs. In order to design practical protocols, considerable research effort has been made over the recent years. Modifications have been made to improve the efficiency of the solutions, such as, by restricting the usage of Yao's protocol to only a few limited computations/operations. For example, Avidan *et al.* [23] speeded up their blind vision protocol [22] at the cost of a controlled leakage of information. Shashank *et al.* [119] on the otherhand, exploited the clustered nature of image databases to improve upon the efficiency for example-based image retrieval.

Limitation of SMC: In general, for any practical online surveillance systems, the protocols are required to be realtime. However, with the current limits on the bandwidth, the proposed SMC-based protocols are not well suited for the task. In SMC based solutions every function is represented as a boolean circuit. The complexity of the protocol is linear in the size of the circuit, making it theoritically efficient, however for real-world applications this method is not practical and is much slower than the correponding non secure computation [21]. For example, [23] takes few minutes to do face-detection on a single image window, thus limiting the practical deployment.

The reason for this is mainly due to the way SMC works: in SMC, every single trusted CPU instruction is securely simulated via a corresponding network protocol. Most efficient SMC protocol is based on *Oblivious Transfer (OT)* [65]. For instance, a single multiplication is carried out via complex distributed protocol involving OT, which is a highly communication intensive subroutine in SMC. OT is a protocol by which a sender sends some information to the receiver but remains oblivious as to what is received. In fact, OT on its own is a fundamental and important problem in cryptography. In particular, it is 'complete' for SMC: that is given an implementation of OT it is possible to securely evaluate any polynomial time computable function without any additional primitive [82].

Typically a task as elementary as a dot product of two n-vectors involves $O(n)$ multiplications, however if OT is used (as in [22]) the resulting algorithm for dot product will have an $O(n^2)$ communication overhead, even with the best known OT algorithm [65]. Furthermore, the best known OT itself is based on public-key-cryptography (PKC) (eg: RSA) [123] which is not only computationally expensive but also results into data expansion, thus affecting the computation and communication costs. As on today, communication is the bottleneck. Factually, the round-trip time in a LAN is of the order of a few milliseconds, whereas several floating operations take no more than few nanoseconds. Thus the paradigm of SMC which converts the trusted computation into secure network protocol can not avoid a slowdown by a factor of one million, with the current technology. Moreover, communication is likely to remain the bottleneck in the foreseeable future. Hence, a paradigm shift is inexorable.

Natural ways to overcome the (communication) complexity include:

1. Avoid distributed computing, and achieve zero communication overhead,

2. Design algorithms for computing in encrypted domain,

3. Design SMC solutions without OT, or

4. Improve the OT protocol.

Unfortunately, cryptographic literature is not in favor of any of the above. Specifically,

1. Retaining privacy without distribution, necessitates *code obfuscation*. However, obfuscation of a general program is impossible [24]. Barak *et al.* [24] showed that one can always find a non-learnable program such that every obfuscator fails completely for it. Although, certain simple programs such as point functions are known to be obfuscated [138], the current vision algorithms are unlikely to have this property.

2. Solutions with minimal distribution, using the paradigm of encrypt-communicate-compute-decrypt require the existence of an *efficient doubly homomorphic encryption scheme*, which scientists have been searching for over 30 years [93]! Without such a scheme, one can only design secure image processing algorithms using either additive or multiplicative homomorphic scheme which involves either addition or multiplication (*but not both*) respectively. Furthermore, the known (additive/multipicative) homomorphic encryption schemes are themselves based on computationally expensive protocols such as public key cryptography. The only known doubly homomorphic scheme is the one recently proposed by Craig Gentry [63] and would most likely lead to a computationally intensive theoretical solution.

3. The inadequacy of solutions with minimal or no distribution, necessitates *non-minimal distribution*, in other words some sort of SMC. It has been proven that the best way to do SMC is with OT [82], though there are more communication intensive solutions based on information theory [26]. The general research direction now is to find computationally efficient solutions under weaker security assumptions than general SMC (Eg: [23]).

4. The only alternative left out is to *improve upon OT protocols*. Even with the best possible OT protocols [65], a slowdown factor of one million (as mentioned in the previous paragraph) will remain. Further, OT is based on public key cryptography, which is known to be computationally intensive. Therefore the extant approach to secure vision is unlikely to yield practical solutions to visual surveillance, even if communication becomes cheaper than computation.

Thus, we conclude that solutions based on SMC are impractical for our real-time task. And hence a paradigm shift is required to address such critical problems. An efficient solution can be designed using a *trusted third party (TTP)*. Under such a solution, one of the servers is trusted with all the private data (available in plain) to faithfully run a surveillance algorithm. Unfortunately, in practice we don't have the luxury of a trusted entity. In fact, such a trusted entity could become a vulnerability in the system. That is, if the trusted entity is compromised (such as by hackers) then the secret could be disclosed.

Unconditionally or information theoretic secure multi-party computations are closely related to the problem of *secret sharing* [118]. Secret sharing (SS) methods [118] [25] [121] try to split any data into multiple shares such that no share by itself has any information about the data, but they together retain all the information of the original data. However, the standard SS methods do not allow efficient computation (as they require some sort of SMC). We show that visual data has certain desirable properties that allows us to use the paradigm of secret sharing to achieve complete privacy and efficient computation of surveillance algorithms. More specifically, in this work we use a variant of the secret sharing schemes based on the *Chinese Remainder Theorem* (CRT) [94] [27].

4.3 The Proposed Approach

The privacy of our surveillance system is based on splitting the information present in an image into multiple parts or shares. Each share is sent to an independent server for processing. The protocol in a nutshell is as follows:

1. The camera splits each captured frame, F, into k (> 2), shares using a *Shatter function* (defined in Sec 4.5): $\phi(F) = [I_1, I_2, \ldots, I_k]$. Each share is then sent to an independent server for processing. Note that no share by itself reveals any meaningful information about the original image.

2. To carry out a basic operation f on the input image, each computation server blindly carries out the equivalent basic operations (as described in Sec 4.5), f' on its own share. This is equivalent to the corresponding basic operation being carried out on the original image: $f'(I_j) \equiv \phi(f(F)))$.

3. The results of operations on the shares are then integrated by the observer using a *Merge function* (defined in Sec 4.5), to obtain the final result: $f(F) = \mu(f'(I_1), f'(I_2), \ldots, f'(I_k))$.

Figure 4.2 shows a schematic diagram of the complete process. The privacy of the overall system relies on the fact that neither the independent shares, nor the results of computations on them, reveal any information about the original image. The integration of results from the independent servers reveal only the final result of the algorithm to the observer.

In order to analyze the security and privacy of the system, we will formalize the notion of security as follows:

1. *Information Revealed:* We use the term information in the strictest information theoretic sense. That is, an observable quantity I is said not to reveal any information about another quantity F (in our case, the original image), if: $\forall_a Pr(F = a|I)$ is same as $Pr(F = a)$.

2. *Preservation of Privacy:* A surveillance system is said to preserve privacy, if it reveals nothing more that the final output of the surveillance algorithm to any party in the system, outside the camera.

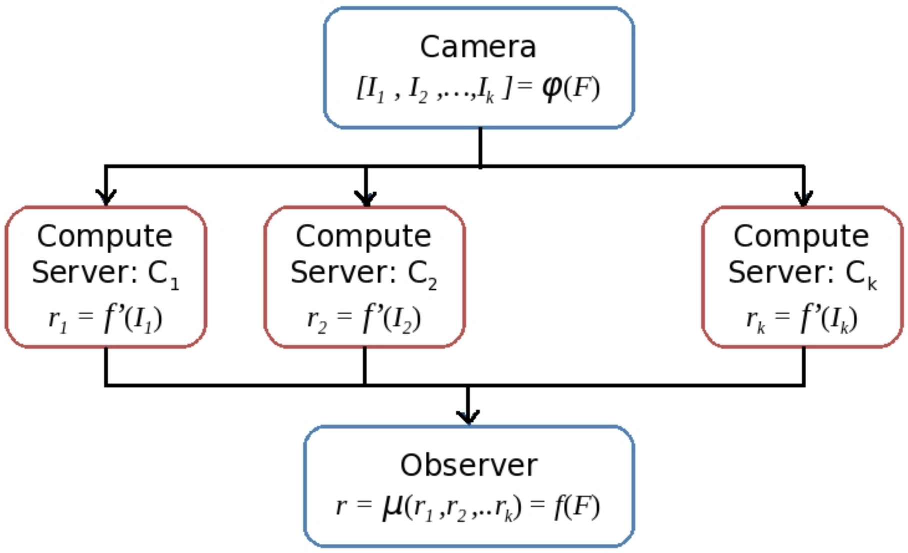


Figure 4.2 Secure computation of $f(F)$ by a set of compute servers. Every frame F is shattered into shares, I_i by the camera. Each server receives and does computation on its (private) share I_i. The observer receives only the 'final' output of the algorithm.

3. *Assumption:* The servers are assumed to be *honest, but curious* in nature. i.e., they will carry out the expected computations faithfully, but may try to learn about the original image from their view of the data. They are independent in the sense that they will not collude to extract any additional information.

The functions $\phi()$ and $\mu()$ that form the basis of our protocol are adapted from the popular secret sharing scheme using the *Chinese Remainder Theorem* (CRT).

4.4 Cryptographic Primitives

This work contributes at the interoperation of security and computer vision in surveillance applications. We now discuss the cryptographic primitives on which the proposed protocol is build upon.

4.4.1 Secret Sharing (SS)

Secret Sharing [25, 118, 121] refers to a method for distributing a secret among a group of servers each of which is allocated a share of the secret. The secret can be reconstructed only when the shares are combined together; on their own, the individual shares gives no information of the secret. Several types of secret sharing schemes have been proposed in literature. Shamir's secret sharing scheme [118] represents the secret as the y-intercept of an n-degree polynomial, and shares correspond to points on the polynomial. In contrast, Blakley's scheme [31] specifies the secret as a point in n-dimensional space, and gives out shares that correspond to hyperplanes that intersect the secret point. The primary motivation behind SS is of securing a secret over multiple servers. However, computing functions on the input secretly shared among n-servers, requires highly communication intensive protocols (which relies on some sort of SMC). Furthermore, such schemes result in huge data expansion, which becomes in-efficient for large secrets, such as live-videos (as in our case). For example, Shamir's shares are each as large as the original secret, whereas Blakley's scheme is even less space-efficient than Shamir's.

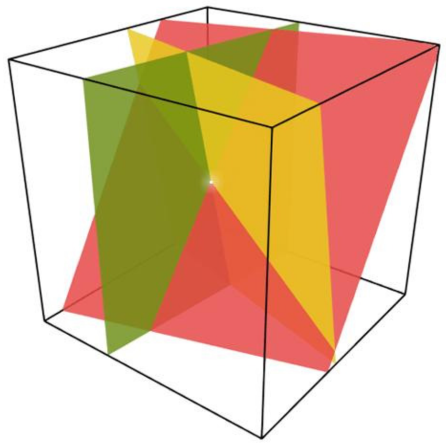


Figure 4.3 Example of Secret Sharing [8]: Each secret share is a plane, and the secret is the point at which three shares intersect. Two shares yield only a line intersection.

The primary limitation of the above schemes was the inability to do efficient computations on secret shares. Asmuth-Bloom [20] overcomes this limitation by working in a *residue number system* [124]. They achieve this by using a special sequence of integers for encoding while CRT is used for reconstruction.

4.4.1.1 SS using the CRT

There are many different versions of the CRT based Secret Sharing Schemes in literature. Mignotte [94] and Asumth *et al.* [20] used co-prime numbers $p_i's$ while Goldreich *et al.* [64] focused only on prime numbers. We now describe the SS using CRT as presented by Goldreich *et al.* [64]. In the remainder of this section, $x \in_R S$ means that x is selected from S with an uniform probability.

4.4.1.1.1 Initialization Let $t + 1 \leq l$, where l is the number of shares and $t + 1$ is the minimum number of shares needed to recover the secret. l primes, each uniquely associated with a server, are selected such that $p_0 < p_1 < p_2 < ... < p_l$. The primes p_i, which play a key role in computing the secret shares, are assumed to be public. The security of the system is based on the non-colluding nature of the servers.

4.4.1.1.2 Sharing Consider a secret $r_0 \equiv s \in \mathbb{Z}_{p_0}$ that is to be securely shared among the l servers. The data owner generates and distributes the secret shares as follows:

1. Choose t uniform random numbers satisfying the constrain, $r_1 \in_R \mathbb{Z}_{p_1}, \cdots, r_t \in_R \mathbb{Z}_{p_t}$
2. Determines $Y \in \mathbb{Z}_P$, where $P \equiv \prod_{i=0}^{t} p_i$ such that $Y \equiv r_i \; mod \, p_i$ for $i = 0, 1, \cdots, t$. This can be efficiently computed using CRT.
3. The secret is encoded in the resulting Y. We now wish to securely share the new secret Y. The secret shares are computed as, $s_i \; = Y \; mod \, p_i$ for $i = 0, 1, \cdots, l$.

The share, s_i, is then sent over to a corresponding, non-colluding server for storage and processing.

4.4.1.1.3 Reconstruction To reconstruct the secret s, a user needs to recollect atleast $t+1$ of the l shares held by different servers. Given a set of $t+1$ shares $\{s_i : i \in I\}$, the secret s is recovered as follows:

1. Compute $X \in \mathbb{Z}_{\prod_{i \in I} p_i}$, such that $X \equiv s_i \; mod \; p_i$ for $i \in I$. This is efficiently computatable using CRT.
2. The actual secret s is given by $s = X \; mod \, p_0$.

In the above reconstruction, note that the recovered X was nothing but the encoded Y. The secret s which was constrained to be $\in \mathbb{Z}_{p_0}$ is correctly (uniquely) recovered by taking $modulo \; p_0$ of the recovered X.

4.4.1.2 Limitation of Standard SS

In the above scheme, each share is of size $|p_i|$ bits. The total share size is therefore given by $\sum_{i=1}^{i<l} |p_i| > l \cdot |p_0|$. Choosing an optimal p_0 becomes curcial since it determines both the range of numbers we can correctly represent as well as the total size of the shares. To understand the data expansion, consider VJ face detector [134] as the algorithm we want to securely execute on a 320*240 input frame. In the input video frame, each pixel is in range [0, 255], as an 8 bit integer. Now, during the execusion of the VJ, the maximum intermediate value we can expect is 320*240*255, a 21 bit integer. Hence our p_0 has to be atleast 21 bits long. Therefore, every 8 bit pixel is expanded into shares with the total size of atleast $21 \cdot l$ bits. This is significant, since the camera needs to communicate the shares over to the servers, thus becoming the bottleneck for the complete system.

The primary motivation is proposing a paradigm shift for our real-time tasks was to reduce communication. We next show that visual-data has certain characteristic properties, which can be exploited to define a tailor made SS scheme exclusively for visual data. Compared to the standard CRT based SS schmes, our scheme significantly reduces the data expansion by atleast a factor of l, where l is the number of servers/shares. Such reduction is prominent for huge data such as live-video feeds, thus making privacy preserving video surveillance practical.

4.4.2 Role of Visual Data

While general purpose secure computation appears to be inherently complex and oftentimes impractical, we show that due to certain "suitable" properties of visual information, *efficiency* and *security* can co-exist in the domain of computer vision! We exploit the following facts, which are valid for most of the computer vision tasks, and in particular for surveillance problem. Meaningful images from real-world have the following properties that are of interest to us:

- **Limited and Fixed Range:** The values that a pixel can take is finite and is from a limited fixed range. And more importantly, the range is known *apriori*. Therefore, algorithms that have multiple possible answers, but only one of them within this range, are as valid as solutions that have only one answer. Such algorithms need not be useful (or correct) for a general purpose (non-vision) tasks. In this work, we exploit this by designing an efficient, secure surveillance solutions that has infinite answers, but only one of them in the valid range. Interestingly, in this process, we also circumvent OT, thereby gaining in efficiency.

- **Scale Invariance:** The information in the image remains practically unchanged even if we change the units of measurement or scale the whole data. This is not true for most non-image information. We exploit this to design a *wrapper* algorithm which converts a partially secure algorithm to a completely secure one. For example, consider a partially secure algorithm that may reveal the LSB of all the pixels. Suppose this algorithm is run on an input which is scaled (at least by a factor of two) with all the LSBs randomized. Then note that with practically no change in the output, the original input (before scaling) is completely secure.

- **Approximate Nature:** The image captured by a camera is an approximate representation of the scene. In practice, the camera sensor that is used to capture an image, is itself noisy. That is, the probability that the camera would generate exactly same images of the scene captured at two different instances, under identical conditions (lighting etc) is negligible. In practice, this implies that the pixel value of $100 \equiv 101$, for the captured image. Thus adding negligible 'hetrogeneous' noise to an image doesn't effect the information retained by the image. Furthermore, as already analysed, chosing an appropriate scale factor can completely overcome any possible accuracy loss incurred due to the added noise.

- **Non-General Operands:** A typical vision algorithm has several operators and operands. However there is no need to expect that all possible pairs could exist in any execution. While it is important to have the ability to implement all operators, it is not practically necessary to allow all possible operands. For example, if an algorithm forbids division by a specific number (say 23), one can still use this scheme as long as you know that your task does not demand division by 23. We exploit this as follows. Traditionally, secure algorithms are designed over finite fields so that the operands as well as operators could be general. Reader may recall from [44] that in the absence of public key cryptography (PKC) and OT, general secure two party solutions are impossible over finite fields. Since we wish to avoid OT and PKC, we design our algorithm over a set which is a finite field with very few exceptions (division by some specific *apriori* known numbers is forbidden). Fortunately, our approach allows one to choose these "forbidden" numbers. Therefore in principle as well as in practice, it is as powerful as having general operands.

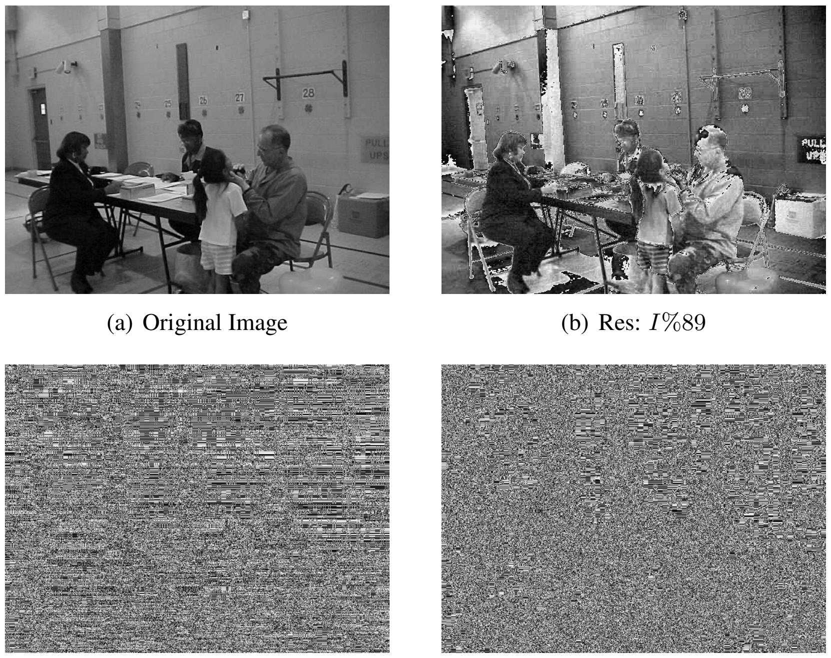
Keeping in mind the above properties, we design a vision specific, distributed framework for surveillance tasks that preserves privacy. The emphasis is on performing this efficiently, thus faciliating proactive surveillance. In our framework, we shatter each video into shares and send each one to a different site in such a way that each site has no information about the scene (data-specific secret sharing). The complete algorithm runs in this distributed setup, such that at the end of the protocol, all that the observer recovers is the final output from the results obtained at each of the site. In summary, our approach does not contradict the theoretical results, but circumvents the problem by exploiting the properties of the data and the problem, without any compromise on the privacy and accuracy.

4.5 Shattering and Merging

An outline of the proposed solution was sketched in Sec 4.3. The three step protocol can be sumarized as 1) Shattering, 2) Computing on the secret shares, 3) Merging the shares to obtain final results. In this section we formalize the subroutines used by our proposed protocol. The *Shatter function* $\phi()$ is used for computing the secret shares, while the *Merge function* $\mu()$ is used for combining the secret shares to retreive the final result.

4.5.1 Shatter Function: $\phi()$

In our problem, we secure each pixel, d of an image, F, independently using a pixel level shatter: $\phi_p()$. The direct CRT based transformation would compute each share as $d \; mod \; p_i$ for different primes



(a) Original Image

(b) Res: $I\%89$

(c) Res: $(I \cdot 44 + \eta)\%89$

(d) Res: $(I \cdot 109 + \eta)\%89$

Figure 4.4 Data Obfuscation: Information retained in the residue image for various scale factors and corresponding η distributions.

(p_i). However, given the correlation between the neighboring pixels in an image, the modulo remainder reveals significant information about the secret (image) (see Figure 4.4(b)). To overcome this, we introduce the following modification to obtain the shatter function $\phi_p()$:

$$d_i = \phi_p(d, p_i) = (d \cdot s + \eta) \bmod p_i, \tag{4.1}$$

where d is a single pixel in the image, s is a constant, positive scale factor, and η is a uniform random number: $U(0, r_{max}), r_{max} \leq s$. Note that the first part of $\phi_p()$, effectively makes the LSBs of the resulting number, random. For example, if $s = 2^k$ and $r_{max} = s$, then k random bits are appended to the right of d. Intuitively, if $p_i < s$, then d_i would essentially be random, and would not reveal any information about d (see Sec 4.6 for analysis).

To shatter an image, we apply the above transformation to each pixel in the image independently, while keeping the set of p_is and s constant. However, we vary the random number, η for every pixel and every image that we encode, and hence the result of modular division is essentially random. Note that without scaling and randomization, the modulo image will reveal considerable amount of information about the original image (see Figure 4.4(b)). Choosing an arbitrary range of randomization, results in partially secure shares (see Figure 4.4(c)). As explained in section 4.6.1, it is possible to choose a range for η that will generate secret shares that are completely obfuscated (see Figure 4.4(d)). In this example, the second-order entropy of image 4.4(d) is identical to that of a pure random noise image. The computed secret shares are then sent over to the respective servers for future storage/processing.

4.5.2 Computing on the Shares

The operations of addition and multiplication are well defined in modular arithmetic, hence making the transformed data appropriate for computations. For example, if f is defined as: $f(x, y) = x+y$, then one can compute $x_i + y_i$ at each compute server and recover $x + y$ at the observer using CRT. In other words, modular arithmetic is homomorphic to both addition and multiplication (doubly homomorphic), within the modulo base. i.e.,

$$\begin{aligned} (a+b) \ \mathbf{mod}\ p &= (\ (a \ \mathbf{mod}\ p) \ + \ (b \ \mathbf{mod}\ p)\ ) \ \mathbf{mod}\ p \\ (a \cdot b) \ \mathbf{mod}\ p &= (\ (a \ \mathbf{mod}\ p) \ \cdot \ (b \ \mathbf{mod}\ p)\ ) \ \mathbf{mod}\ p \end{aligned}$$

As mentioned before, given the value of the rhs of the above equations for multiple values of p, one can exactly recover $(a + b)$ or $(a \cdot b)$ using CRT. Thus, every computational server executes a modular implementation of the plain-domain surveillance algorithm, with its private share as input. The resulting outputs at each of the server is then sent over to an observer, who uses the *Merge function* $\mu()$ to recover the final results.

Question: *What operations are forbidden in RNS?*

Notice that the domain of RNS in not a *finite field*. Recall that any finite field has a cardinality of p^x for some x, where p is a prime, where as we are working in a domain of cardinality $M = p_1 \cdot p_2 \cdots p_k$. Hence in our domain, $\mathbb{Z}_M$, not all divisions are possible. Specifically, division by any number v, where some p_i divides v is not defined in $\mathbb{Z}_M$. However, we can get around this problem in our application due to the following facts:

- The number of forbidden divisors is small. For instance in the example given in line number 426, out of the 1813 number, only 85 numbers are forbidden as divisors (less than 5%).
- The forbidden numbers are known in advance, and in most algorithms the divisors are independent of the data. Hence, we can choose a RNS system that allows all divisors required in the algorithm.

In our implementation, we take a simpler route: assume that all divisions are forbidden, and implement the division operation using yet another untrusted server. The randomization and shuffling as mentioned in Sec 4.7 secures the intermediate results from all parties.

4.5.3 Merge Function: $\mu()$

Given, $d_i = \phi_p(d, p_i)$ for different prime p_is, the secret d can be recovered by *CRT* by solving a system of congruence. The *merge function* $\mu_p()$:

$$d = \mu_p(d_i, p_i) = \frac{CRT(d_i, p_i)}{s} \tag{4.2}$$

CRT recovers $(d \cdot s + \eta)$, which is appropriately scaled down (integer division by the scale factor) to get the actual value of d. The solution is unique if all the intermediate values $(d_i, f'(d_i))$ are less than the product of the primes $(p_i s)$. Note that η, which was randomly chosen for each pixel is not used for recovering the secret. The CRT hence forms our recovery transformation $\mu_p()$, at the pixel level. Thus, provided all the shares are made available, the secret (the original video) can be correctly regenerated by the merge function. In our case, the results of the surveillance algorithm running at each of the servers are made available to an observer, who applies the merge function to retrieve the final result. Note that the only information learned by the observer is the final output and nothing else.

Question: *How is CRT being applied to get output?* RNS is doubly-homomorphic within modulo domain, i.e. both addition and multiplication can be correctly carried out independently on the residues. Therefore, an algorithm that can be computed as a function $f()$ made up of addition and multiplication operations can be implemented in the RNS. During the design, every addition and multiplication operation in $f()$ is executed in the RNS. Thus, at the end of the protocol, what we have is the shattered shares of the final output. Merging the modular outputs will correctly recover the final result of $f()$. To clarify the process, Section 4.5.3 provides some numeric examples of secure computation a set of functions.

Note that:

- We achieve efficiency by working in the modulo domain, thus avoiding communication overheads of solutions such as SMC.
- Working in modular domain does not ensure privacy (since residue is revealed). This is more serious for visual data, since modular image would retain most of the information of the original image. We achieve privacy by adding random numbers to every pixel before taking the mod.
- Evidently, directly adding random noise is a poor solution, since accuracy is traded off. We scale pixel values before adding random numbers to maintain accuracy. Section 4.6.1 shows that we can select the scale factor, the random numbers and the prime divisors to achieve perfect accuracy while maintaining complete privacy.

- The surveillance algorithm runs independently on each of the servers. Interaction is limited and for evaluation of functions which require merging of results, eg: thresholding. This is again secured by randomization/shuffle and use of another untrusted server.

- CRT is used to recover the result from the output of each server.

In this model, the only communication required from the client is that of sending the shattered data to the compute servers. Each compute server can perform most of the computations independently, and hence the communication overhead is very limited. In order to boost the understanding of the complete process, we next consider a few trivial examples.

Examples of Modular Arithmetic

The modular transformation that forms the basis of our protocol is based on RNS and CRT. Therefore the operations that are defined in the RNS are also well defined for us. To verify the computational capacity of our protocol, let us consider the following set of examples.

For the first example, consider the processing of the image patch shown in Figure 4.5. As each pixel is processed independently, consider the pixel with value 68. Let the scale factor s be 33 and the random number η, be 10, the resulting $d \cdot s + \eta$ would be 2254. If the image is shattered into three shares, with primes 19, 29, and 31, the corresponding shattered shares would be: $\{12, 21, 22\}$.

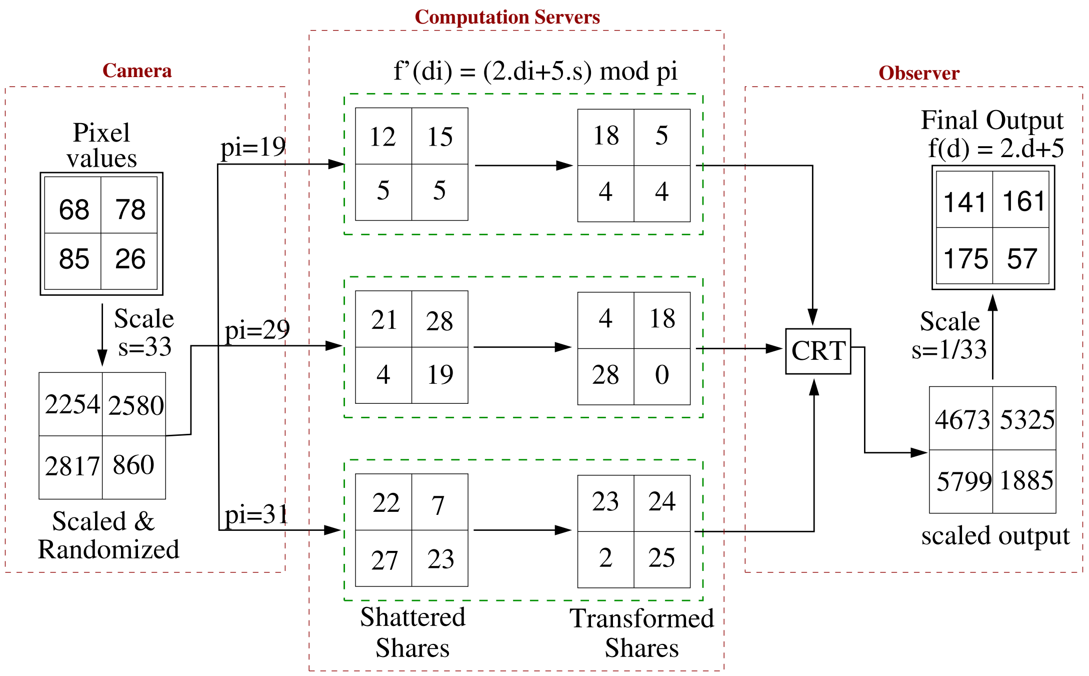


Figure 4.5 Affine intensity transform in modular domain.

If the algorithm applies the affine transformation $f(d) = 2 \cdot d + 5$ to each pixel d, then each compute server will carry out $f'(d_i) = 2 \cdot d_i + 5 \cdot s\%p_i$ independently on its share, and obtain $\{18, 4, 23\}$. The observer will integrate these numbers using CRT to obtain 4673, which when divided by the scale factor 33 gives 141 $(2 \cdot 68 + 5 = 141)$.

The above computations are valid only for integer values of operands and results. As the pixel values of an image has a limited and finite range, these conditions are easily satisfied. Moreover, any non-integer operation can be correctly simulated by appropriately scaling the entire data/algorithm before the shattering operation. One can exactly reconstruct the ideal output by scaling down the results as long as all the intermediate value in the entire computation are correctly represented in the Residue Number System(RNS), i.e. the product of the primes p_i is greater than any intermediate value in the entire computation.

Next we consider an example of *'Overflow'*. Overflow occurs when the numbers being represented are beyond the range of the employed RNS. This leads to errors in the recoverd results using CRT. Let us now consider doing a summation of pixels in a patch. Every pixel in the image has a value in the range [0 - 255]. For a 2×2 patch, the output will be in the range [0-1020]. Figure 4.6 summarises the complete process. As one can notice CRT in this case recovers the sum as 588, whereas the expected correct output is 553.

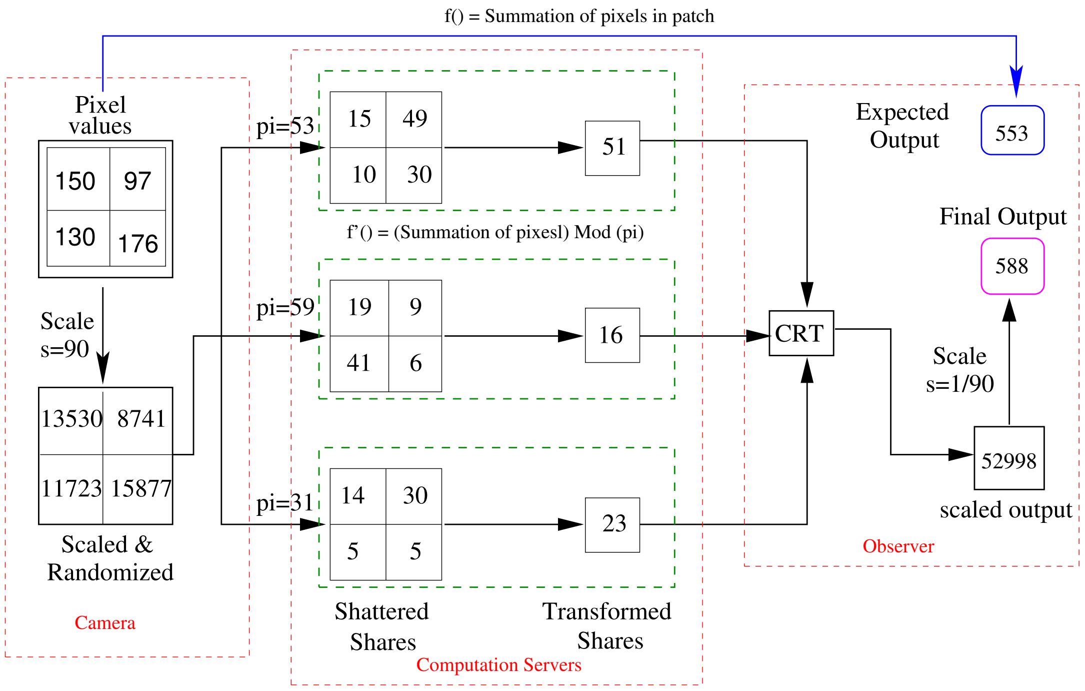


Figure 4.6 Overflow: The correct output is beyond the range of RNS, thus results in error in the recovered result using CRT.

The reason for this is chosing an inappropriate RNS. In the above example, we employ 3 primes, $p_1 = 53$, $p_2 = 59$, $p_3 = 31$ and the scale factor, $s = 90$. This RNS correctly represents numbers in the range $[-(53*59*31)/(90*2), (53*59*31)/(90*2)]$, i.e. $[-538, 538]$. Now since the expected result of 553 is beyond the range of RNS, recovery using CRT leads to errors.

On the otherhand, suppose we just wanted to add the first two pixels. In this case the maximum sum can be ($255*2 = 510$), which is less than upper-bound of the RNS. In this case the pixel values 150 and 97 are shattered and sent to the servers. The servers do the summation and sends over to the observer 11, 28, 13 respectively. CRT now recovers 22271, which when scaled down by 90 gives 247 (i.e. 150+97).

Next example considers the task of computing a polynomial function $f(d) = d \cdot d/100$. The maximum intermediate value we need to represent in this case is $255*255$. A RNS comprising of 4 servers with primes $p_1 = 587$, $p_2 = 593$, $p_3 = 383$ and $p_4 = 193$. The scale factor choosen is $s = 600$. Figure 4.7 shows processing a pixel array using the function $f(d)$. The original image pixels are scaled by s and shattered into 4 components. The shattered components are independently squared at the remote servers. The results recovered by CRT are then scaled down by s^2 to get the expected results. For a polynomial function of order t, the appropriate scaling is given by s^{-t}. The intuitive reason behind this, is that every pixel is scaled by s, therefore, $f(d)$ gets scaled by s^t and we need to scale it down by s^{-t} to get the correct output.

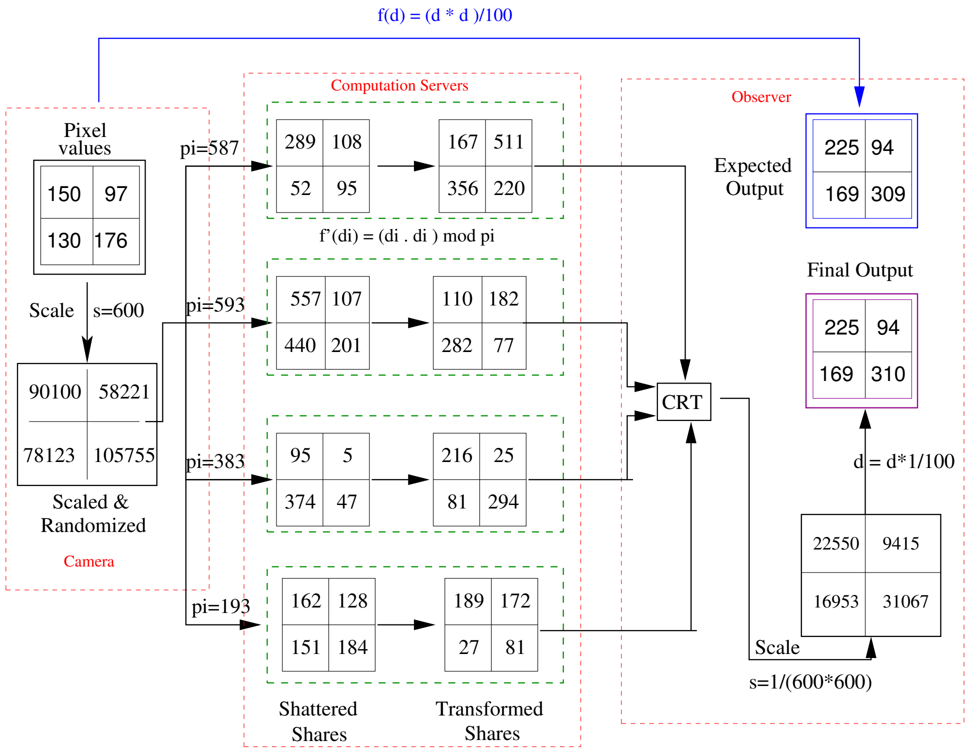


Figure 4.7 Polynomial function: Every pixel is modified using a polynomial $f(d) = (d \cdot d)/100$.

4.6 Theoritical Analysis

Parameter selection is a pivotal step for achieving efficiency and privacy in the proposed protocol. We now take a closer look at the optimal parameter selection and the associated computational and communication overheads. We also take a closer look at the amount and the nature of information revealed by the residue images and the related security and privacy concerns of the proposed system.

4.6.1 Selection of the Primes and the Scaling Factor

Let the number of computation servers employed by the sytem be k. Given the value of k, we select $p_1, \dots, p_k$ such that their product, P, is greater than any number that one would want to represent correctly during the running of the algorithm. Typically, one could just choose the smallest k consecutive primes satisfying the above property. The scaling factor chosen should be higher than the largest prime, so as to obfuscate each share (see Figure 4.4). Note that the range of the intermediate values is a function of the scale factor. So this process might have to be iterated a couple of times during the design phase.

Let M denote the maximum intermediate value that is to be represented in the surveillance algorithm, when run in the plain domain. Assume that we require a scale factor of s to achieve complete privacy. Let c be the constant such that $M.s^c$ is the maximum intermediate value to be represented after scaling. Note that c depends on the algorithm and is usually small (≈ 2) in most vision algorithms, for example if the operations being performed are linear then c is 1, for quadratic functions c would be 2 and so on. In other words, the primes $p_1, \dots, p_k$ are chosen such that:

$$s \geq \max_i p_i, \text{ and } s < \left(\frac{\prod_i p_i}{M} \right)^{\frac{1}{c}} \tag{4.3}$$

Simplifying the above, we find that, if

$$M < \left(\max_i p_i \right)^{k-c} \tag{4.4}$$

then the original image is hidden from the individual servers. At the same time, we can guarantee that the reconstruction by CRT is unique. The above inequality is a sufficient condition and our experiments show that it is usually not necessary.

4.6.2 Computation and Communication Overhead

In our process, the only computation expected of the camera is of generating the secret shares, i.e. applying the shatter function to every pixel of the captured image. For a system with k servers, this involves choosing a random number and k additions and multiplications per pixel. As shown by our experimental results, this can be done on the fly with generic hardware. Each server now independently runs the surveillance algorithm on its secret share. The complexity of the algorithm is same as the original plain-domain implementation. The only computational overhead is for division and thresholding operations, for which the merge function has to be executed. This involves an additional $k + 1$ multiplications per operation. A final merge function has to be executed by the observer to reconstruct the final output.

The communication overheads involves sending the residue images from camera to the servers. Each residue image is made up of pixels of size $|p_i|$ bits. The total shatter size is therefore the summantion of k residue images and is equal to $|P|$ bits per pixel. Thus, for an original 8 bits CCTV stream, an overhead of $|P| - 8$ bits per pixel takes place. All servers now execute the algorithms independently, with the communication limited to division and thresholding operations. For each of these operations, each server, i, sends a $|p_i|$ bits share to a common server, who merges the received shares and sends back the $|p_i|$ bits shares of the result of the operation.

4.6.2.0.1 Comparision with Standard CRT based Secret Sharing In the standard CRT based SS scheme, the size of the each share is atleast $|P|$ bits. Thus for k servers, the total share size will be atleast $k \cdot |P|$ bits. On the otherhand, in our scheme, the total share size is kept to $|P|$ bits, therefore reducing the data expansion, thus communication, by a factor of k. The division and thresholding are performed in a similar manner in both the schemes, however in our scheme the communication overhead is minimized, since each share is of size $|p_i|$ bits, while in standard it is alteast $|P|$ bits, thus reducing the communication costs by a factor of k.

4.6.3 Analysis of Privacy

Privacy of the system refers to the amount of information of the CCTV video that is revealed to the server. An ideal and perfect zero-knowledge scheme requires the shares to be of equal size (as in Shamir scheme). However, in the CRT based SS schemes, the sizes of the share space and secret space are not equal. Quisquater *et al.* [105] showed that CRT based SS schemes are asymptotically optimal both from an information theoretic and complexity theoretic viewpoint when the parameters satisfy a simplified relationship. We now show that our modified scheme is also asymptotically optimal for visual data and thus satisfies the same notion of security as in [105].

Note that the privacy of the system is based on splitting the information present in an image into multiple shares. The parameters used for shattering i.e. the primes p_i and scale factor s are constant for each shattering operation and are in general assumed to be public. The only possible information leakage of the secret is that retained by each share. We now analytically show that with an optimal parameter selection, the information retained by a share is negligible. Consider a pixel with value $d \in [0, 255]$, which is scaled by s, followed by addition of a random number, η. Note that, if η follows the uniform distribution, $U(0, r_{max} - 1)$, the distribution of $r_i = \eta \% p_i$ would be:

$$Pr(r_i = x) = \begin{cases} (k+1)/r_{max}, & x < r_{max}\%p_i \\ k/r_{max}, & otherwise, \end{cases} \tag{4.5}$$

where k is $\lfloor r_{max}/p_i \rfloor$. The above distribution will be uniform, leading to perfect security if r_{max} is a multiple of p_i. On the other hand, if r_{max} is not a multiple of p_i, there is a slight step of size $1/r_{max}$ (see figure 4.8) in the resulting distribution at $x_k = r_{max}\ mod\ p_i$, where r_{max} known only to the camera.

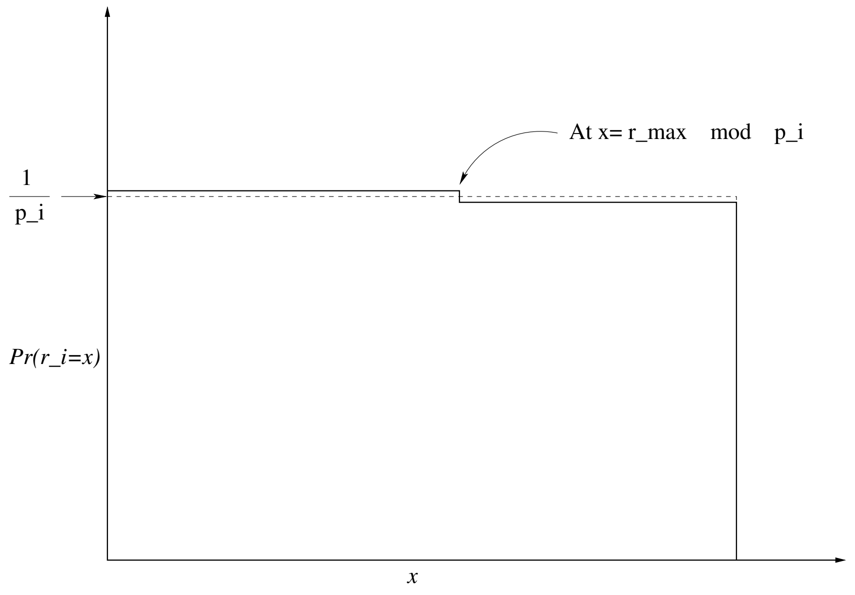


Figure 4.8 Remainder Distribution: Ideal (dotted) and practical (solid) distributions of the shatter for $d = 0$.

If the pixel value is d, and in the worst case remains exactly d across a very large number of frames, the i^{th} server might be able to estimate a distribution with a kink at $s \cdot d + r_{max}$ mod p_i. Even in the worst case, if the server is able to exactly detect the kink, it only knows the value of $s \cdot d + r_{max}$ mod p_i. Therefore, the amount of information about d is exactly same as the amount of information as r_{max}. If we take a naive approach of choosing r_{max}: i.e., choose is randomly from a range $[q, q + R]$, then r_{max} mod p_i will not be uniformly distributed for all p_i – however, it will be very close to uniformly distributed, with a step at R mod p_i. That is, for a particular pixel value d, the share d_i, is marginally more likely to be within a certain range of pixel values.

In short, we see that the method of shattering is statistically impossible to break for images, and unlike encryption based methods, even if it is broken, nothing useful can be learned from the information that is revealed. The advantage of our method is that such a high level of security can be achieved, while allowing computations to be carried out efficiently on the shattered videos.

4.7 Implementation Challenges

As described above, the process of carrying out integer addition and multiplication in the modulo domain is relatively straight forward. We might feel that one can achieve any operation that can be modeled as a combinatorial network of *AND* and *XOR* gates, which makes it equal in power to a general purpose computer. However, there are many practical challenges to be overcome to implement the functionalities.

4.7.0.0.2 Representing Negative Numbers Modulo arithmetic is carried out using positive integers only. Hence one has to map the range of numbers used in an algorithm to an appropriate range of positive integers. Signed numbers in the residue form are represented with an implicit sign. If the range of numbers used is (0, M), we used the numbers in the range (0, M/2) to represent positive numbers, and for the remaining numbers it is negative. The change in sign of $|Z|_M$ is performed by the operation of additive inversion of Z, i.e. -Z = M - Z, which is equivalent to $(m_1 - z_1, m_2 - z_2, ...m_k - z_k)$.

4.7.0.0.3 Overflow and Underflow The employed RNS, correctly represents integers in the range (-M/2, M/2). Use of numbers beyond this range will result in errors in the recovered results using CRT, which we refer to as overflow or underflow. Overflow and underflow are safely avoided by working in a domain large enough to correctly represent all intermediate values encountered. However, the net data size of the shattered video streams is directly proportional to the domain-size that we work with. An efficient domain can be chosen by precomputing the upper bound on the possible intermediate values, and then appropriately deciding on the RNS.

4.7.0.0.4 Integer Division and Thresholding Operations such as divisions and thresholding are difficult to achieve in RNS. Division of an integer A by B is defined as $A/B = (a_i.{b_i}^{-1})$ mod m_i in the RNS. This is valid if B is co-prime with M and B divides A. For this to always hold, one would have to take into account B, in choosing the RNS. Though this looks practical (since the original algorithm is known beforehand), it might not always be efficient since the shattered data size ($\#$ of bits) is directly proportional to the chosen domain-size (consider the case where multiple division operations are to be performed using different divisors, validating the division for every divisor would result in a blowup of the domain size, thus affecting efficiency).

An alternate solution for division and thresholding can be designed using an additional computational server. Every independent computation server (ICS) sends over their respective residues to the additional server, where the merge function is applied and the division/comparison is performed in plain domain. However, simply doing this would end up revealing the intermediate results to the additional server. To secure against such information leakage, every ICS does a reversible randomization (multiplying by a constant) of their respective residues before sending them over to the additional server. The additional server does the division/comparison on the randomized data(post merging) it received. The computed result is shattered by it and sent back to the respective ICS where it is de-randomized to retrieve the actual(expected) modulo result.

4.7.0.0.5 Defining Equivalent operations Finally, the primary challenge remains in defining the implementations over the modular domain for any algorithm. For every function $f(d)$ we need to define $f'(d_i)$ such that merging $f'(d_i)$ would give back $f(d)$. In general, one can imagine building a compiler that converts a given function $f()$ into the equivalent function $f'()$. However, the complete treatment of the generic solution is beyond the scope of this thesis. In the next section we will describe the implementation and analysis of a few standard operations used in video surveillance.

4.8 Implementation and Analysis

We now provide a detailed account of the implementation and analysis of a common surveillance task, tracking of moving objects, using the proposed framework. We describe the mapping of this problem to the framework and show the steps involved in carrying out the computations. We also describe in brief, the results of face detection in the proposed framework.

The process of tracking is carried out in two steps, that of change detection by background subtraction, followed by tracking of points of change.

4.8.1 Background Image Subtraction

We first consider the problem of background removal. Specifically, our problem is: given a static background image (in shattered form), subtract it from each captured image, such that at any point of time, the original image or the background image is not revealed to anyone. At the end of the protocol, all that is learned by the observer is the final output (the difference image). The complete process can be sub-divided into:

1: Deciding the RNS: Every pixel in the image has a value in the range $[0 - 255]$. In background subtraction, the range of numbers in the result is $Y = [-255, 255]$, or $Y = [0, 511]$ (using an implicit sign). Shattering involves scaling every pixel by s. Therefore in the RNS we need to correctly represent the range $R = [0, s.Y]$. The optimal number of servers and the prime numbers defining the RNS is chosen as described in Section 4.6.1. In our example, we have number of servers $k = 3$, the scale factor $s = 33$, and the primes are $19, 29$, and 31. These set of parameters form our RNS, which is made public.

2: Defining the Algorithm: Next step is to map the original algorithm into modulo domain. In our case it is pixel-wise subtraction of a fixed background image (B) from the captured image frame (F), both in the shattered form. As the only operation that needs to be performed is subtraction between two scaled quantities, the equivalent operation in the modulo domain would be the subtraction of the corresponding shattered pixel values as described in Section 4.5.2.

3: Capture Image and Shatter: The image captured by the camera (Figure 4.9(a)) is shattered using the modular transformation described in Section 4.5.1. In our example, the parameters used in the shatter function are the ones as obtained above. Input frame F is scaled by $s = 33$ and its lower order bits randomized to obtain an image I'. The image I' is then shattered using the primes $p_i s$ and the shattered shares are sent to the corresponding servers.

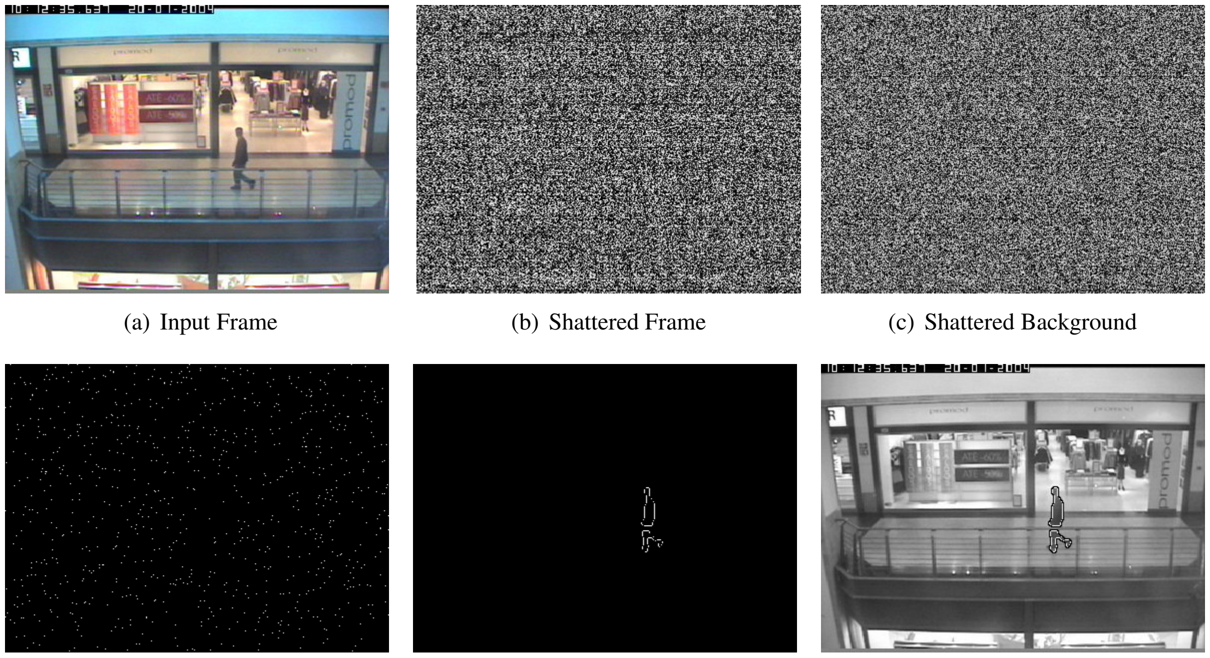
4: Apply the algorithm on shattered components: The shattered components received by the servers are now independently processed at each of the servers. As defined in step 2, the background image B' is subtracted from the each of the input frames. At the server i, the arithmetic operations are done modulo prime p_i. For example if the difference of two pixels is computed as $D = 100$, and the corresponding prime for the server is $p_i = 29$, then the difference is stored as $100\%29 = 13$.

5: Merge the outputs at the observer: The computed results are sent over to the observer who uses the merge function (see Section 4.5.1) to obtain the final output. In our example, the observer obtains the 3 shattered images. Now the observer uses the Chinese remainder theorem(CRT) to reconstruct every pixel of the output image from the corresponding pixel values of the shattered images it receives. For example if the components of a particular pixel after subtraction are $\{12, 0, 11\}$ corresponding to the primes $\{19, 29, 31\}$, CRT would reconstruct -1508 from these values. The result is then scaled down by the initial scale factor 33 to obtain the final result as -45.

4.8.2 Change Detection

The detection of change involves subtraction of a frame from a background frame, which is carried out as explained before. We also update the background image by replacing pixels in the background where change is detected with the corresponding ones from the foreground. This is done directly in the RNS. The difference values are integrated by a thresholding server, using CRT, which then compares the result against the pre-defined threshold to detect motion.

However, sending the shattered differences to a threshold server reveals the difference image. To avoid this, we apply a reversible pixel shuffle that would remove any structure in the image, before sending it to the threshold server as explained in Section 4.7


(a) Input Frame (b) Shattered Frame (c) Shattered Background

(d) Shuffled Difference (e) Change Detection (f) Output+Original Image

Figure 4.9 Change Detection: (b,c) are the shattered shares seen by one of the compute servers, (d) is the obfuscated difference image obtained after a pixel shuffle, (e) is the output as available with the observer, and (f) comparison with original image.

Image Resol.	# of Servers	Comp. Time		Commn.	
		Serv.	Merge	Data	Time
PITS'00 768x576	3	0.367	1.294	324	0.025
	5	0.362	1.433	270	0.017
	7	0.377	1.316	162	0.013
CAVIAR 384x288	3	0.110	0.292	81	0.006
	5	0.122	0.310	67.5	0.005
	7	0.137	0.338	40.5	0.003
Towers 320x240	3	0.071	0.189	56.25	0.004
	5	0.074	0.201	46.87	0.004
	7	0.073	0.217	28.12	0.002

Table 4.2 Average computation and communication times for change detection.

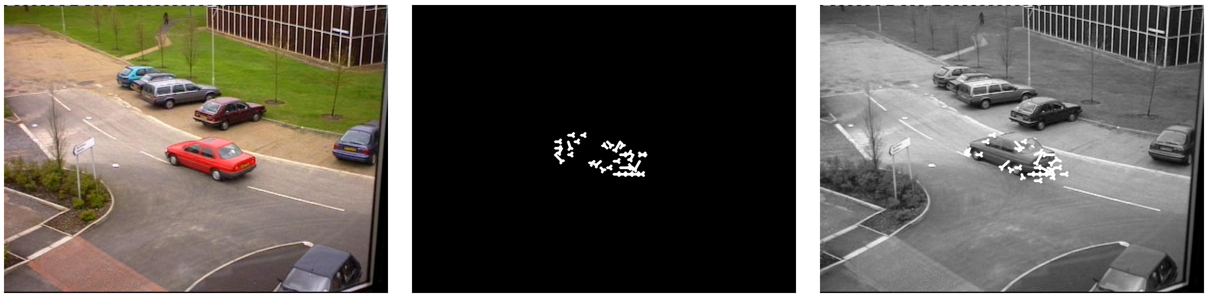
The additional (untrusted) server thresholds the received image to get a shuffled binary image. This is sent back to each of the ICS, where it is de-shuffled to obtained the final binary image. As the result is now in plain domain, one can also apply any post-processing operations such as erosion, merging, etc. to remove any noise. Moreover, the background learning can work in the transformed domain as the pixels with no change are known to the compute servers in each frame. We also note that the accuracy of the algorithm is not affected by the obfuscation process, as indicated by the comparison with plain domain result. Figure 4.9 shows a sample frame that is being processed in the framework.

Table 4.2 shows the exact time (in seconds) spent by the individual compute servers as well as the thresholding server (usually the observer). We note that even with a non-optimized implementation on a desktop class machine, one can achieve a computation speed of upto 14 (QVGA) frames per second at each server. The total data to be transmitted in the process (in Kilobytes) and the corresponding communication time, assuming a 100Mbps connection between the two servers, are also given. We note that most operations are carried out in sub-second times.

4.8.3 Optical Flow and Tracking

The above algorithm can be further extended to compute the optical flow. We use the change detection results as guidelines, and compare a patch around each motion pixel against its neighbors. The comparison is done using correlation, which is similar to the affine transformation operation described in example in Figure 4.5. The optical flow estimates thus derived, along with the motion segmentation results from the previous section can be used to build a complete system that detects and tracks people/objects.

Figure 4.10 shows the results of optical flow being computed using the secret shares and the results superimposed on original frame.



(a) Original Frame (b) Tracking (c) Tracking in Plain

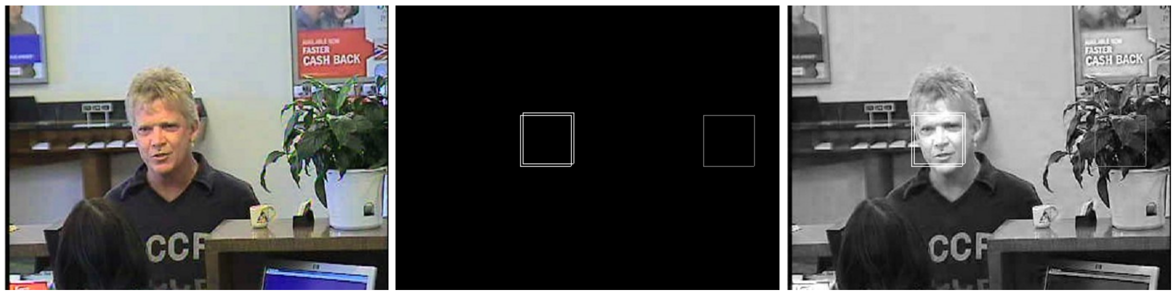
Figure 4.10 Computation of Optical Flow: (a) Frame from input sequence, (b) optical flow computed, and (d) results superimposed on original frame. Note that the shattered images are omitted here.

4.8.4 Face Detection

For the next experiment we implement a more complex classifier, the popular face detection algorithm by Viola and Jones(VJ) [134] that uses a cascade of classifiers. Each classifier is dependent on a set of Haar-like features. We note that all the features can be computed by addition or subtraction of pixel values within a rectangular neighborhood. As the pixel additions and subtractions can be implemented directly in the RNS, the mapping of feature computation into our framework is straight-forward.

Note that we do not attempt to learn the classifiers from the shattered images, but use those which are learned from plain domain images. In a nutshell, VJ adopts a rejection cascade, where every image-window is passed through the cascade to detect faces. An integral image representation(which is summation of pixels) is computed for the input image. For every stage of the cascade, the rectangular features are computed from the integral image (this involves addition and subtraction operations). The computed feature values are securely merged and compared against the cascade threshold to decide upon the acceptance/rejection of the window.

Considering that the only operations involved in VJ are addition, subtraction and thresholding, it is fairly straight forward to define the equivalent functions for modular domain. Each computation server computes the integral representation of its own secret share (this is equivalent to 'shatter' of the integral image computed in plain domain). The cascade (which is trained in plain domain) is then applied independently at each server. Every window of the image is then passed through the cascade, for which the feature value is computed for every weak classifier. In our setup, every server can independently compute its share of the feature value (computed from the integral representation on its secret share). Thresholding is done (as described before) with the help of a pixel shuffle and an additional server. The location of the windows that pass through the complete cascade are made known to the observer as final output. Figure 4.8.4 shows the result of face detection on an input image as obtained by the observer as well as the plain domain result for comparison. Once again we note that the outputs are identical. One



(a) (b) (c)

Figure 4.11 Face detection (a): Captured input image, (b): Result as received by observer, and (c) Detection result, if run on the plain image. The detected faces are shown in white boxes and the current window being processed is shown in gray.

can also notice that the plain image as well as the feature values computed are hidden from all parties involved, thus securing against any possible information leakage, and in the process only knowledge gained by the observer is the final output.

Image Resolution	Parallelization				
	3	5	7	10	12
200x 200	463.0	289.4	231.5	173.6	173.6
320x240	994.8	621.7	497.4	373.0	373.0
400x320	1777.1	1110.7	888.5	666.4	666.4
512x512	3908.4	2442.7	1954.2	1465.6	1465.6

Table 4.3 Data transferred between thresholder and servers (in KB).

Table 4.3 shows the variations in the amount of communication between the compute servers and the observer as the size of the image and the number of servers vary. The amount of data is shown in Kilobytes, and can be communicated over high speed connections to the observer for thresholding.

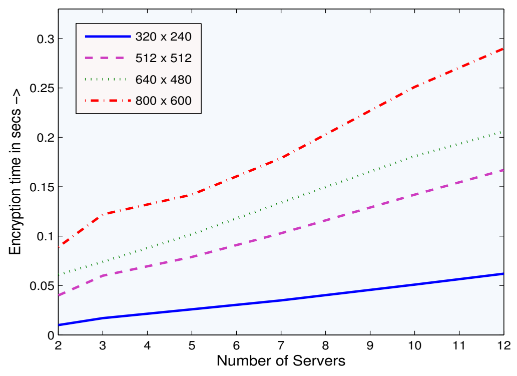
4.8.5 Overheads of Parallelization

To estimate the effects of encoding in terms of computation and communication overheads as well as accuracy, we study three different aspects. In the first experiment, we compute the average number of bits in an encoded frame and the total image size. An interesting observation from Table 4.4 is that as the number of compute servers (or primes) increases, the average bits in the resultant image first decreases, and then increases. The increase in the later part is due to the need of using larger primes, which drives up the size of the resulting image. One can always choose an optimal number of servers, as already explained in Section 4.6.1.

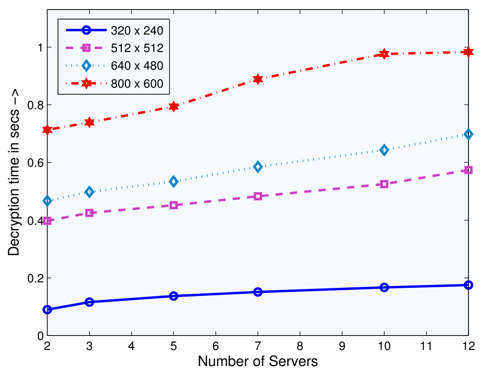
			Avg Data Size	
# primes	Scale	Avg bits	Size/Frame	Total Size
3	17	6	56.25	168.75
4	31	5	46.87	187.48
5	19	4	37.50	187.50
10	13	3	28.13	281.30
20	11	5	46.87	937.40
50	31	6	56.25	2812.5
100	37	7	65.23	6523

Table 4.4 Average data size (without compression) vs. amount of parallelization

Figure 4.12 shows the time required for shattering and merging a frame. We note that the time required for merging is considerably higher due to the use of large-number arithmetic when dealing with scaled numbers. Even then, the system is able to do these operations in well under a second.



(a) Shattering Time



(b) Merging Time

Figure 4.12 Time required to shatter/merge a frame with increasing number of servers.

4.8.6 Comparision with SMC

Clearly, any secure solution needs more processing over an insecure one (eg. http vs https). For an application, if overheads are within acceptable limits, a provably secure method is always preferred. Our method compares with crypto solutions (SMC based) in terms of privacy, which are extreamly inefficient. Our main achievement is an approach that achieves information theoretic privacy, while being extreamly efficient over SMC. Our cameras can be inexpensive as the in-camera operations are simple and fixed. For example, VJ based FD on a QVGA frame requires evaluation of 92636 windows ($W_o = 24$, $H_o = 24$, $S = 1.25$). As per [22], each window takes a few seconds for a non-face and several minutes for a face. Even at 2 secs per window, this translates to 185272 secs (51 Hrs) per frame, which we reduce to 2-3 secs per frame. The processing time can further be improved by processing several windows simultaneously.

4.8.7 Data Transformation and Accuracy

An image is encoded as the residue images computed from a scaled noisy version of the input image. To study the quality of the restored image, we conducted an experiment to encode a set of varied images over a range of parameters and computed the PSNR scores (Table 4.5 of the recomputed image. We see that all the images have PSNR in the fifties. Note that for image compression purposes, a PSNR value of above 35 is considered very good, and our transformation is practically loss-less.

As the images are represented faithfully by the transformation, and the algorithms are exactly mapped from the plain domain, the performance of the algorithms in the proposed framework would be the same as that of their plain domain equivalents. Furthermore, we note that the noise η, that we add to a scaled pixel is conditioned to be always less than the scale factor s. This is equivalent to adding a noise of less

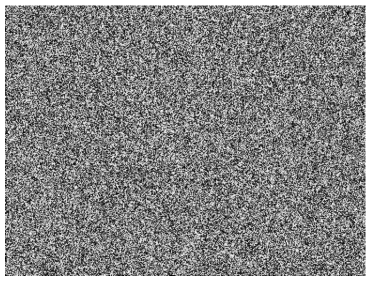
Image	Scaling Factor			
Resolution	11	31	80	120
320×240	51.552	51.309	51.138	51.134
512×512	51.598	51.345	51.176	51.161
640×480	51.568	51.301	51.141	51.138
800×600	51.567	51.307	51.142	51.134

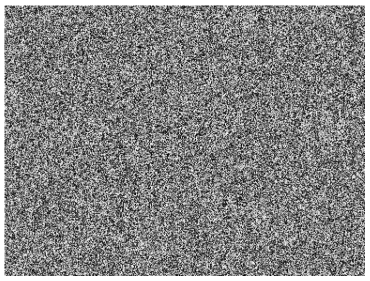
Table 4.5 Peak Signal-to-Noise Ratio(PSNR), for k = 5

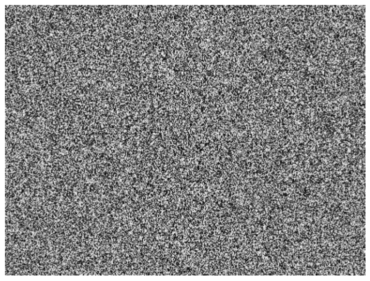
than 1 unit to the original image. This noise is often far less than that present naturally in surveillance videos and does not affect the results of the algorithms.

We have presented experimental results on a range of tasks and data (change detection, optical flow tracking, face detection). The results presented were aimed at understanding the computational and communication overheads at each stage, any loss of accuracy incurred by computation in the proposed framework, as well as the effectiveness of data obfuscation for privacy. We consolidate the experimental evaluation by providing a few more visual results.

(a) Input Frame


(b) Shattered Share 1


(c) Shattered Share 2


(d) Shattered Share 3

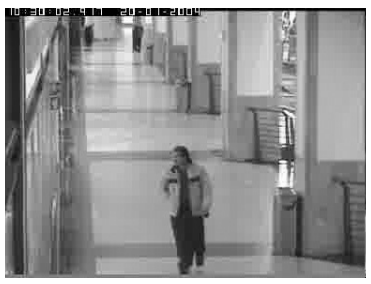
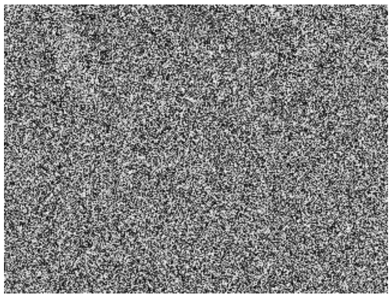
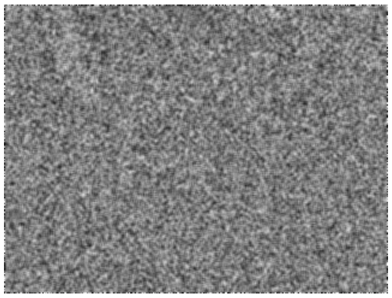
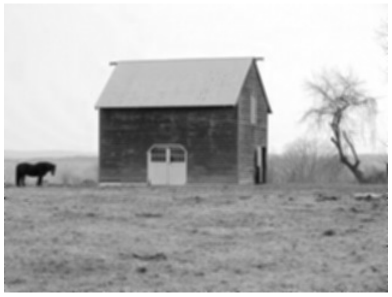

(e) Result of Merging

Figure 4.13 Shatter and Merge: The input Frame is shattered into 3 components. The reconstruction is done by the merge operation. The high quality reconstruction is validated by the PSNR score of 51.34.

(a) Input Frame


(b) Shattered Share


(c) Convolution of Shattered Share


(d) Merged Image

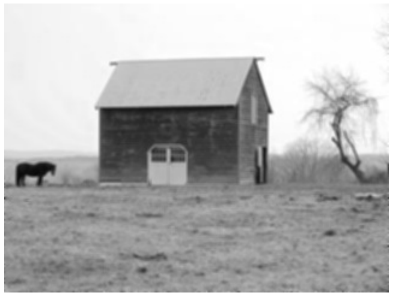
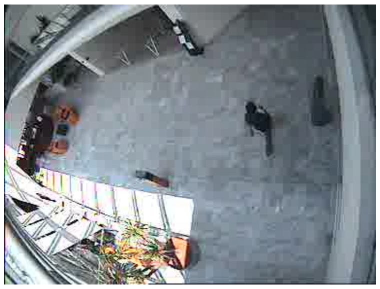
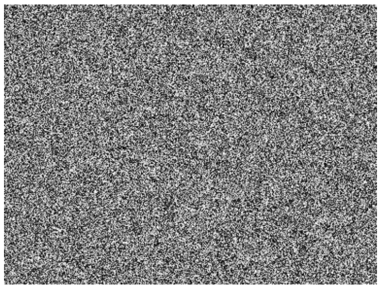
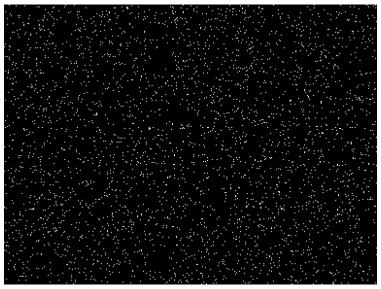
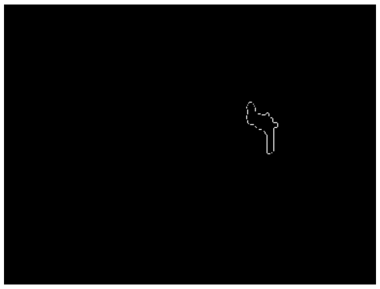

(e) Convolution in Plain

Figure 4.14 Image Convolution: (a) Input Image, (b) Shattered Frame obtained by a server, (c) Processed shattered Frame, (d) Final Result obtained after merging, (e) Equivalent result obtained by processing in plain domain.


(a) Input Frame


(b) Shattered Frame


(c) Shuffled Result


(d) Final Result Obtained

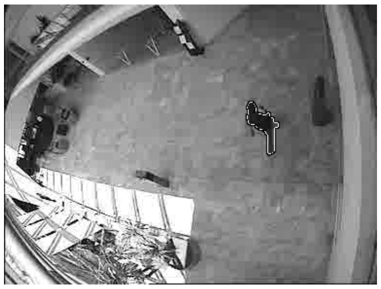
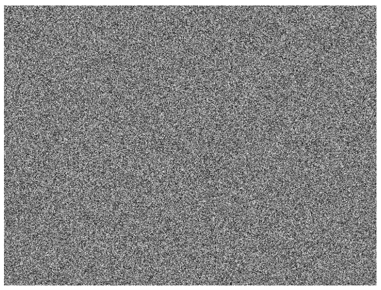
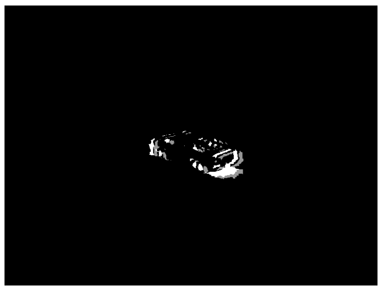
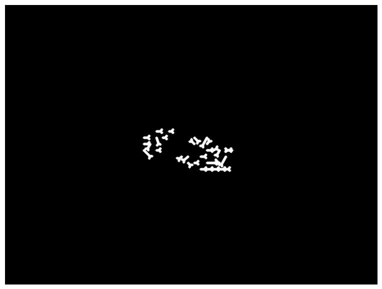

(e) Expected Result

Figure 4.15 Change Detection: (a) Image Frame as captured by Camera, (b) Secret share, as received by one of the servers, (c) Shuffled result available with the thresholding server, (d) Detection Result, post merging, as obtained by observer, (e) Corresponding detection results as computed in plain domain.

(a) Input Frame


(b) Shattered Share


(c) Motion Detection


(d) Merged Result

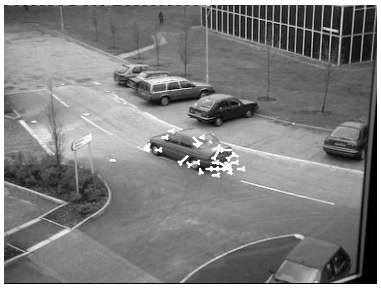
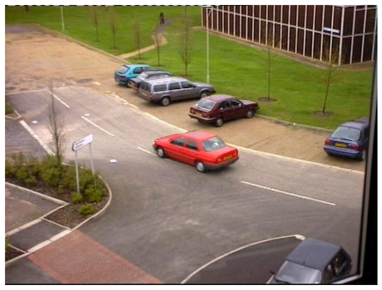
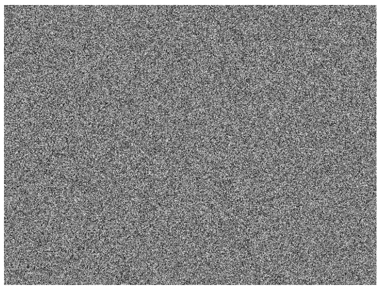
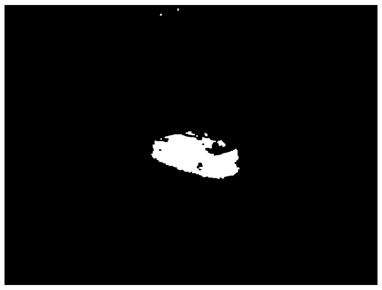

(e) Expected Result

Figure 4.16 Optical Flow. (a) Input Frame captured from camera, (b) Shattered share, (c) Change detection history, (d) Results obtained post merging by observer, (e) Result on computing in plain domain.


(a) Input Image


(b) Shattered Share


(c) Change Detection

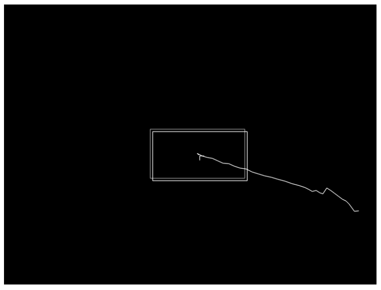
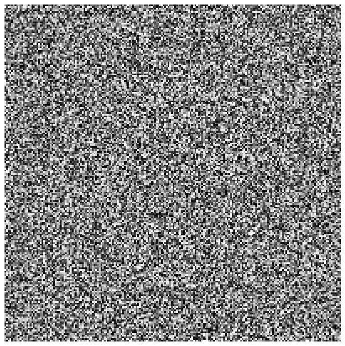
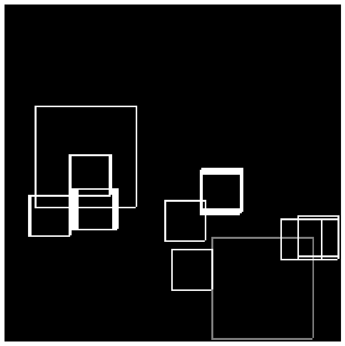

(d) Tracking Result

Figure 4.17 Object Tracking: (a) Input image as captured by camera (b) shattered share available at one of the servers, (c) privately performing change detection performed, (d) tracking result as obtained by the observer. Change detection is performed as described before. The patch matching is done by finding maximal correlation.

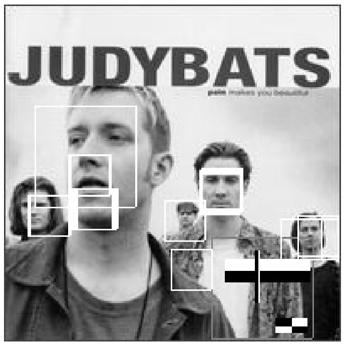
(a) Input Image



(b) Shattered Share



(c) Merged Image



(d) Expected Result

Figure 4.18 Face Detection using VJ: (a) Input image, shattered and sent to servers, (b) shattered share at one of the servers, (c) result as available with the observer, (d) result in plain domain. Detected faces are shown in white, while the current window being processed is given in gray. Note: The exact detection results are computed, (same positive/negative window detection takes place)

4.9 Discussion

The main contribution of the work is in introducing a paradigm shift in looking at private visual surveillance problems from the traditional SMC based approaches. This change in view allows us to have a simplified capture device, an efficient unidirectional data flow, and surveillance operations performed directly on the shattered streams. One could imagine a simple modification to the surveillance camera with modulo division implemented in hardware, producing, say 3 randomized video streams. The streams will be connected to three different service providers, ensuring that none of the servers gain any information about the image. Whenever a thresholding operation needs to be performed, they can communicate with a common server, through high speed networks to achieve the functionality. Such an architecture provides us both security as well as computation and communication efficiency.

Note that only the surveillance results will be available to the observer (law enforcement, government, or security provider). The plain video stream will be revealed only if all the service providers collude, which is easy to avoid. The fact that each video stream, arriving at the individual servers, lack any information of the original video, provides us with a simple mechanism for secure archival of surveillance data. Essentially, each of the servers can store their streams independently. Note that if one needs access to the plain data, one can make the service providers release their respective data under a court order for legal or investigative purposes. The overall architecture is thus designed to address all the concerns mentioned in Table 4.1. One could also build dedicated hardware based solutions for each in a much more efficient manner, essentially making private video surveillance, practical.

As noted from Figure 4.12, the shattering and merging operations are extremely efficient. Using an approach that avoids encryption or cryptographic protocols such as SMC enable us to achieve real-time performance on video surveillance tasks. For example, one can carry out change detection at each of the servers at 15 frames per second even with a non-optimized implementation on a desktop class machine. One could build dedicated hardware based solutions for each in a much more efficient manner, essentially making private video surveillance, practical.

In addition to the above advantages, our approach is also easily scalable, as the computations carried out be each server is identical. One could design a single specialized hardware for a compute server, and replicate it to achieve larger scale operations. This also provides fault-tolerant properties, when combined with the use of CRT. One can recover the results of computations even if one server stops working, as long as the product of primes from the remaining servers are more than the largest number used in computation.

Above all, the framework provides a generic setting to carry out an arbitrary vision task. As the basic operations of addition and multiplication can be realized in the transformed domain, one could imagine building a compiler that transforms any existing non-secure implementation of a surveillance algorithm into a secure/privacy preserving one. However, one might need to define more generic procedures for operations such as sorting and memory indexing to achieve this. In short, the approach has the potential to extend to other areas of visual data processing as well.

One of the disadvantages of the shattering algorithm is the reduction in our ability to compress the resulting video schemes. One might note again that ability to compress, and ability to hide information are opposing goals for a representation. Possible approaches to overcome this include working with compressed video sequences as input, or the use of number theoretic compression schemes.

In short, we have presented an efficient, practical and highly secure framework for implementing visual surveillance on untrusted remote computers. To achieve this we demonstrate that the properties of visual data can be exploited to break the bottleneck of computational and communication overheads. The issues in practical implementation of certain algorithms including change detection, optical flow, and face detection are addressed. This work opens up a new avenue for practical and provably secure implementations of vision algorithms, that are based on distribution of data over multiple computers.

Chapter 5

Private Yet Efficient K-Means Clustering

In this work we introduce an efficient privacy-preserving protocol for distributed K-means clustering over an arbitrary partitioned data, shared among N parties. Clustering is one of the fundamental algorithms used in the field of data mining. Advances in data acquisition methodologies have resulted in collection and storage of vast quantities of user's personal data. For mutual benefit, organizations tend to share with each other, their data for analysis purposes, thus raising privacy concerns for the users.

Over the years, numerous attempts have been made to introduce privacy and security at the expense of massive additional communication costs. The approaches suggested in the literature make use of the cryptographic protocols such as *Secure Multiparty Computation (SMC)* and/or *homomorphic encryption schemes* like Paillier's encryption. Methods using such schemes have proven communication overheads. And in practice are found to be slower by a factor of more than 10^6.

In light of the practical limitations posed by privacy using the traditional approaches, we extend the idea of computing on secret shares to enable unsupervised learning algorithms such as clustering. *Secret sharing* allows the data to be divided into multiple shares and processed separately at different servers. Using the paradigm of secret sharing, allows us to design a provably-secure, cloud computing based solution which has negligible communication overhead compared to SMC and is hence over a million times faster than similar SMC based protocols.

5.1 Introduction

K-means clustering [51, 62] is one of the most widely used techniques for statistical data analysis. The simplicity and effectiveness of the algorithm have made its usage conducive in various applications ranging from machine learning, pattern recognition and data mining. Researchers use cluster analysis to partition the general population of consumers into market segments and to better understand the relationships between different groups of consumers/potential customers. However, the collected data may contain sensitive or private information about the customers, thus heightening the related privacy concerns [47, 128].

To clarify our problem setting, let us consider a scenario where several companies have collected certain information of their clients. Having such large collections of data provides them with an ideal opportunity to gather knowledge that could improve the performance of the organizations. However, privacy and secrecy considerations can prohibit them from sharing their sensitive data with each other. The widespread applicability of the problem setting makes it imperative to find a secure and an efficient solution to such a significant problem. Addressing the problem requires many practical challenges to be overcome before a possible wide-scale deployment. The solution should not just be provably secure i.e. it leaks no additional useful information, but should also minimize the additional overheads in terms of communication and computation costs required to introduce privacy. Vaidya *et al.* [131] summarize the state of art methods available for privacy preserving data mining. They adequately answer the questions of 'when', 'why', and 'how' privacy preserving data mining solutions becomes the need of the hour. More detailed reviews of the previous work can be found in Verykios *et al.* [133].

A simple solution to gain maximum knowledge is to make all the organizations to share their data. However, this results in zero privacy. On the other hand, a *trusted third party (TTP)* based solution, alleviates all privacy concerns. Under these solutions, the aggregate data is made available to a TTP, who runs the clustering algorithm and answers to clients queries [49, 52, 53]. However, finding a trusted third party is in general infeasible in real world [65]. Moreover, as the data is available in plain, TTP is susceptible to be compromised. Privacy preserving data mining was introduced to address this specific problem. Solutions were sketched to extract knowledge by making the participating parties to compute common functions, without having to actually reveal their individual data to any other party [17, 88].

Previous solutions can be primarily categorized as, *i)* those using *Data Perturbation techniques*, and *ii)* those employing *Secure Multiparty Computation (SMC)*. The first category of approaches introduces noise and data transformations to achieve partial privacy [81] [37] [91]. The clustering is then done of the noisy version of the data, resulting in approximately correct clusters [17] [100]. Such approaches compromise privacy for practicality, however the key advantage is the negligible communication overhead needed by such approaches.

The second category of approaches aims to achieve complete privacy. This is done using the well known cryptographic protocol of SMC [65]. SMC facilitates a group of people, each with its own private data, to perform some common computation task on the aggregate of their data. SMC ensures that, in the process, no any personal information of data is revealed to any one [89]. However, the SMC based protocols are found to be extremely computationally expensive [65]. In other words, an operation which requires a single round of communication in a non-secure implementation, would require hundreds of thousands of rounds of communication (depending on the domain size) to achieve the same operation in a secure implementation using SMC. For data mining applications, the sheer volume of the data involved makes the protocol infeasible in terms of the communication cost. For example, Vaidya *et* *al.* [132], İnan *et al.* [70] and Wright *et al.* [72] use SMC as a subroutine to propose privacy preserving clustering. However, the huge computational costs makes these solution impractical. Considerable modifications have been made to improve the computational efficiency, such as, by restricting the usage of Yao's protocol to only a few limited computations/operations [16].

Another set of proposed approaches uses the semantically secure additive or multiplicative homomorphic encryption schemes [101]. In such a protocol, one party encrypts its data using its public key, and share the encrypted data with the other party for computation. Interactive protocols are then designed to carry out the clustering algorithm [77] [37]. The overheads of encryption and the communication costs needed to carry out clustering limits the scope of such algorithms. Interaction can be reduced with the usage of a doubly homomorphic scheme [106]. However, the only known doubly homomorphic scheme is the one recently proposed by Craig Gentry [63] and would most likely lead to a computationally intensive theoretical solution.

Most of the recent works specific to privacy preserving K-means clustering have concentrated on building interactive protocols. Vaidya *et al* [132] proposed a solution to K-means clustering over vertically partitioned data. They used a clever randomization and permutation algorithm to privately achieve cooperation among the parties. Thresholding is done using a secure circuit evaluation. However, the computational costs makes the solution impractical for large data-sets. Jha *et al* [77] proposed solutions for horizontally partitioned data. The first of their protocol is based on oblivious polynomial evaluation, and the second one uses homomorphic encryption. The proposed protocols provide provable security of the private data, however in the current form reveals the intermediate cluster locations to each party and is restricted to horizontally partitioned data. İnan *et al* [70] proposed another protocol for horizontally partitioned data. Their provably secure solution is again based on the secure multiparty computation of dissimilarity matrix over the data. Wright *et al* [72] introduced the concept of arbitrarily partitioned data. Their solution too uses SMC as a subroutine. As Bunn *et al* [37] points out, in [72] the interpretation of division as multiplication by the inverse does not satisfy correctness. Bunn *et al* [37] proposed solution is also based on homomorphic encryption schemes.

In this work, we achieve the security at the level of SMC while keeping the communication costs to a level similar to that of the first category. We achieve this using the paradigm of the *Secret Sharing* [20] over a mesh of processing servers. Our solution is first of its type, and is both efficient and mathematically simple. In the process we also side-step the communication bottlenecks posed by the usage of SMC and asymmetric encryption schemes. Our proposed solution is not only computationally efficient but also secure independent of whether or not $P \neq NP$. We however do assume the servers to be non-colluding and having the ability to generate random numbers.

5.2 The Problem Setting

We address scenarios of N parties, each with its own private data, wishing to privately collaborate for doing cluster analysis on their aggregate data. Practicality of the solutions to this critical problem faces many challenges. The solution needs to not only be efficient and provably secure, but should also avoid any trade-offs between accuracy and privacy. Table 5.2 summarizes the primary concerns that needs to be addressed for widespread adoption of the protocol. We now look at the architecture of our proposed solution.

We propose a 'cloud computing' based solution that utilizes the services of R, $(R > 2)$, non-colluding servers. Each of the N users, is required to compute the R secret shares of its private data using a *shatter function* (see the algorithm, defined in Section 5.3). Each share is then sent over to a specific server for processing. Note that the shatter function ensures that the computed secret shares on its own reveal no information about the original private data. The cloud of employed servers, now runs the K-means algorithm using just the secret shares. The protocol ensures that none of the users/servers have sufficient information to reconstruct the original data, thus ensuring privacy.

a) Data Security: Secure storage of sensitive data is important. Every organization wants to secure itself against a possible data theft by either an insider or a hacker.

b) Accuracy: Introduction of privacy should not compromise the accuracy or the results of the algorithm.

c) Privacy Preservation: The participating parties would like to keep their sensitive data private during computations. No information, other than non-sensitive data mining results are allowed to be learned by others.

d) Efficient Computation & Communication: Large overheads are to be avoided. Obfuscation should not blow up the data-size. Moreover, every user wishes to minimize the communication costs at its end.

e) Facile deployment: Collaboration between two parties should not be hindered by practical issues in deploying PPDM. A dedicated mesh of computation servers implementing the PPDM protocol makes it viable.

f) Reconstruction of data: Organization should have an efficient method to reconstruct the private data. Obfuscated data, with each server, should not reveal any information of the original data.

g) Fault tolerance: Ability to avoid the denial of service (DOS) attacks and single point of failures.

Table 5.1 Primary challenges in practical privacy preserving data mining (PPDM)

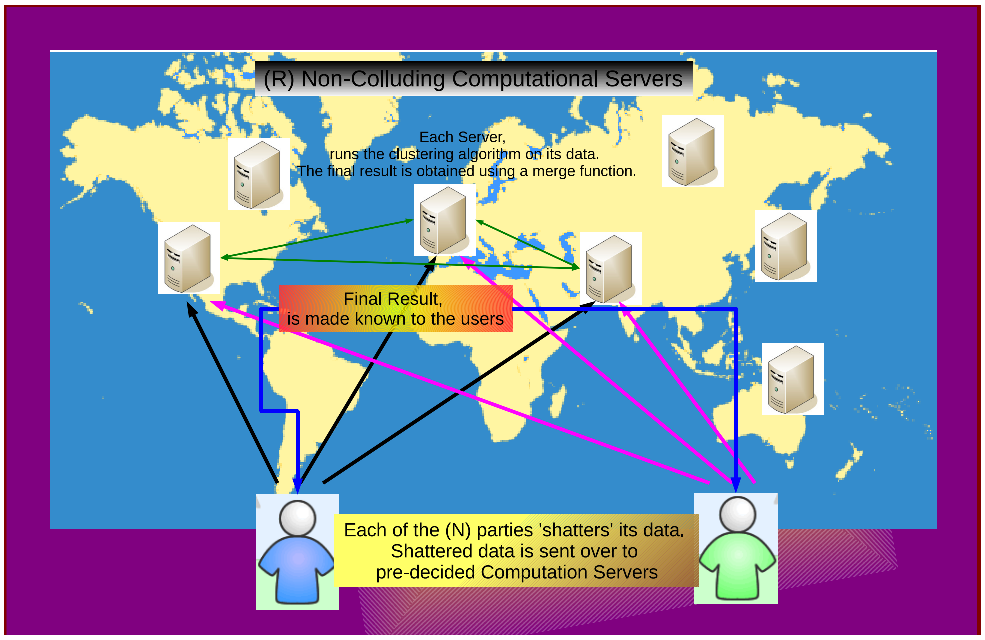

(R) Non-Colluding Computational Servers
Each Server,
runs the clustering algorithm on its data.
The final result is obtained using a merge function.
Final Result,
is made known to the users
Each of the (N) parties 'shatters' its data.
Shattered data is sent over to
pre-decided Computation Servers

Figure 5.1 Sample Mesh of servers. Each of the N users shatters their private data (Sec: 5.3) and sends over the components to the pre-selected R servers for computation. The final result is obtained by merging (Sec: 5.3) the outputs of the computational servers. In above example, N is 2 and R is 3.

The shatter function that we choose allows efficient computations using just the shares. That is, unlike SMC, the number of rounds of communication to implement an operation on secret shares is equivalent to that required in a non-secure implementation of the same operation. The advantage of this is that it significantly reduces the communication costs over the similar SMC based protocols, thus making privacy preserving clustering practical. Figure 5.1 shows a pictorial description of the proposed architecture, while the algorithm is discussed in detail in Section 5.4.

Figure 5.2 shows the notion of arbitrary partitioned data as described by Wright *et al* [72]. *Note:* it is a generalization of both horizontally and vertically partitioned data. We borrow the same notion and extend it to N users. In our setting, the attribute names form the public information. Each of the entity is either completely owned by one of the users, or the attributes are shared among the N users, where the share of some users can also be ϕ. If a record is 'completely' owned by anyone, then its existence remains hidden from other users. If any of the attributes for an entity are with more than one user, in that case a weighted average of the attribute values is considered for the computational purpose. Entities are indexed using a mutually agreed upon indexing scheme. The indexing scheme addresses the two concerns of *i)* hiding the entity's identity from the servers, and *ii)* a common index for accessing the vertically partitioned data.

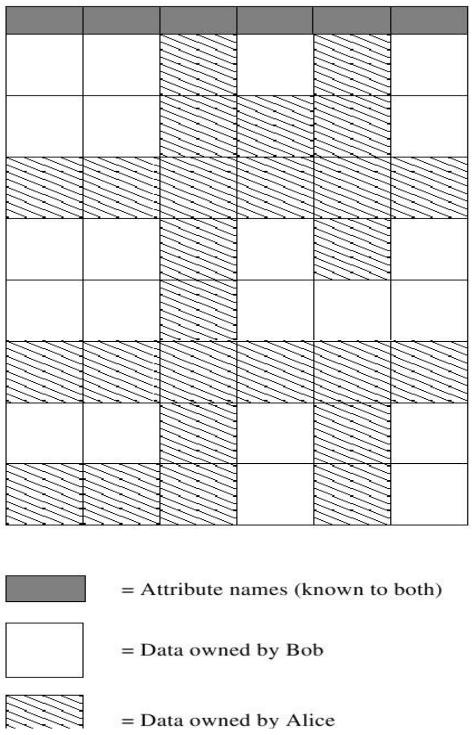


Figure 5.2 Arbitrarily partitioned data (among 2 parties) as described by Wright *et al*

The only computation required of the users is to *shatter* their respective data and *merge* to get the final result. Communication costs for them are of transferring the shattered data to the servers and downloading the outputs. The actual algorithm is being run on the cloud of servers owned and operated by (untrusted) service providers. The expected outcome of the protocol is to correctly classify the data points without revealing any information. That is, the cluster assignment should be identical to when no privacy protection is employed. Every user should only learn the cluster assignment of its own data. The actual cluster locations can be made known to the users, if agreed upon.

Our proposed framework provides the ability to efficiently cluster the private data partitioned among various users. We compare and analyze the computation and communication overhead of our protocol against a zero-privacy protocol, under which each user sends his data (in plain) to a third party for clustering. In our process, all concerns mentioned in Table 5.2 are addressed. Note that our proposed approach does not fall into any of the three categories discussed in Section 5.1. This work therefore opens up a new direction of research to look at privacy preserving data mining.

5.3 The Building Blocks of Security

We use the paradigm of Secret Sharing (SS) to achieve privacy and efficiency. Secret Sharing (SS) [118] [25] [20] refers to the methods for distributing a secret among a group of servers, each of which is allocated a share of the secret. The secret can be reconstructed only when the shares are combined together; on their own, they have no meaningful information. In our problem setting, we ask each of the collaborating users to compute the secret shares of their private data, and send them over to the processing servers. The processing servers then privately collaborate (without reconstructing the actual data) to run the K-means algorithm over the secret shares. *Note that*, not all SS methods allows computation on the secret shares. In order to achieve this, we adopt the Chinese Remainder Theorem (CRT) based secret sharing schemes [20] [64].

However, in the SS schemes of Asmuth *et al.* [20], and Goldreich *et al.* [64], the size (the number of bits) to represent each share is greater than the size of the original data. In other words, for R servers, using these schemes results in a minimum of R fold storage increase. Data expansion is important since it results in cost overheads in terms of storage and interaction among the servers. It becomes even more critical for applications such as data mining that deals with voluminous data.

Understanding the similar limitations, in Chapter 4 we proposed an efficient method to do privacy preserving surveillance on videos (voluminous data). In this work, we extend the method and propose secure protocols to privately carry our collaborative clustering. The data to be clustered using K-means can be thought of as points in a D dimensional Cartesian space. The data is bounded, i.e. it has a fixed range, and its scale invariant, i.e. even if we scale the axis, the cluster assignment will still be the same. These two are the required desirable properties of the data, that are sufficient for one to adopt the secret sharing scheme as proposed by us in Chapter 4. We therefore, adopt the *Shatter* (to compute the secret shares) and *Merge* (to reconstruct the secret) functions for the Cartesian data and design a communication and computationally efficient solution to achieve privacy preserving K-means clustering.

Our proposed solution can be summarized as a three step protocol, 1) each user computes the secret shares of his private data, 2) shares are then sent over to a cloud of servers and clustering is privately carried out over the shares, and 3) the users reconstructs the cluster assignment and the cluster centers using the *Merge* function. Before we jump into describing the K-means protocol in Sec: 5.4, for the sake of completeness we briefly describe the *Shatter* and *Merge* functions as defined in Chapter 4. The *Shatter* and *Merge* functions as defined in Chapter 4 are as follows:

Shatter Function $\phi(x)$ **-** ***Compute and store the secret shares of the private data :*** is defined as the one that splits the data x into R parts, x_1, x_2, ..., x_R, such that each share, x_i, by itself does not reveal any information about x. The participating users pre-decide a set of R primes $P_1, \cdots, P_R$ and a scale factor S. The *Shatter function* is defined as:

$$x_i = \phi(x, P_i) = (x \cdot S + \eta) \bmod P_i, \tag{5.1}$$

where x_i is i^{th} secret share, and η is an independent random number for each secret x, such that $0 \leq \eta \leq S/2$. The secret share x_i is stored with the i^{th} server and on its own gives little meaningful information of x.

In our scenario, each user can shatter his data (each attribute of a record is shattered independently, η is random for each attribute) and sends over the shares to the specific servers for storage. The size of each share is given by $log(P_i)$ per attribute.

Merge Function $\mu()$ **-** *Reconstruct the secret :* given, $x_i = \phi(x, P_i)$ for different prime P_is, the secret x can be recovered using *CRT* [46] by solving a system of congruence. The *merge function* $\mu()$ is defined as:

$$x = \mu(x_i, P_i) = \frac{CRT(x_i, P_i)}{S} \tag{5.2}$$

CRT recovers $(x \cdot S + \eta)$, which is appropriately scaled down (integer division by the scale factor) to get the actual value of x. Note that η, which was randomly chosen for each attribute value is not used for recovering the secret. The CRT hence forms our recovery transformation $\mu()$. In our scenario, $\mu()$ is used for reconstructing the cluster centers as computed by the clustering algorithm.

5.4 The Proposed Algorithm

Following notations are used for describing the protocol. Let L be the number of entities, each made up of D attributes. K be the number of clusters required, and $\vec{C_i}$, $1 \leq i \leq K$, denotes the cluster locations. The data is arbitrary partitioned among N users. R $(R > 2)$ is the number of computation servers employed. Each server is associated with a unique prime P_i, therefore the number of primes is also R. Each entity is represented in a D dimensional space. The common distance metrics; such a Euclidean, Manhattan or Minkowski; are used for finding the distances. To explain the algorithm we will consider a Euclidean space. As the final output of the privacy-preserving K-means (PPKM) algorithm, each user learns the cluster assignment of the entities owned by them, i.e. which of their entities belong to each clusters. If agreed upon, the location of the K-clusters is also revealed to the users.

The complete protocol can be divided into two phases. The *first phase* deals with *i)* choosing the appropriate primes and the scale factor, *ii)* shattering the data, and *iii)* secure aggregation of the data at the servers. The *second phase* of the protocol deals with the clustering algorithm on the aggregate of the shattered data available with the R computational servers. The basic algorithm follows directly from the standard K-means algorithm [95], which consists of three steps, *i)* Initialization, *ii)* Lloyd Step, and *iii)* Stopping Criterion. The complete protocol is as follows:

5.4.1 Phase One: Secure Storage

The first step is the selection of an appropriate residue number system (RNS) for secure storage. We extend the *analytical method* to compute the parameters required for ensuring the security and privacy in our problem setting. For a value of R we select $P_1, \cdots, P_R$, such that their product, P, is larger than any intermediate value we have to represent in our algorithm. This range can be easily computed from the range of values we expect in the computations. Scaling the axis and translating the origin of an Euclidean space does not change the final cluster assignment. Hence we represent negative numbers with an implicit sign [129], i.e. $-x \equiv 2M - x$. Floating point data is taken care of by appropriately scaling the dataset to retain a certain decimal precision.

Let $[-U, U]$ be the range of numbers we expect in the computations on secret shares. We choose P_j's such that $P = \prod_{j=1}^{R} P_j \geq 2U$. Typically, one could just choose the smallest of the R consecutive primes satisfying the above property. For complete obfuscation of the data, the scaling factor chosen should be higher than the largest prime. We now analytically choose the optimal set of parameters for our problem setting.

5.4.1.0.6 Parameter Selection: Let $[-M, M]$ be the attributes domain. Then the points can be represented in a $D - dimensional\ Euclidean\ space$, $\mathbb{R}^D_{2M}$. Let W_1 be the square of the maximum possible Euclidean distance between two points, i.e. the distance between the two extreme points, thus we get $W_1 = 4M^2D$. Also let W_2 be the maximum sum of the coordinates we can get for a cluster (needed for computing the cluster's mean). This is easily computable as $W_2 = 2ML$ (entire database belong to a single cluster). Let W be the upper range of number we expect in K-means, therefore we have $W = max(W_1, W_2)$.

Let us now assume, S to be the required scale factor to get complete privacy. The input data is scaled using this factor. This can be viewed as scaling the axis of the Euclidean space by S, i.e. a point x in the old coordinate system is mapped to $S \cdot x$ in the new scaled space. Therefore, we get $U = max(W_1 \cdot S^2, W_2 \cdot S)$. The primes now need to be chosen such that:

$$S \geq \max_j P_j, \text{ and } P \geq 2U. \tag{5.3}$$

Simplifying the above, we find that if:

$$S \approx (2W)^{\frac{1}{R-2}} \tag{5.4}$$

then the individual servers will have little meaningful information.

Each of the N-parties uses the *shatter function* (Eqn: 5.1), to compute secret shares of their respective data. The shares are then sent over to the servers for processing. Note that we make no assumptions on how the attributes of various data points are partitioned among the N-parties. If $\mathcal{D}$ is the (virtual) database arbitrarily shared among the N parties. Each server j basically then stores the shatter of $\mathcal{D}$ w.r.t. P_j.

5.4.1.0.7 Privacy: Each server stores only the shattered share of the data. As long as the servers do not collude, little meaningful information of the entities is learned by any of the servers. This follows directly from the security of the shattering scheme. In this entire phase the only information learned is of how the data is actually being partitioned among the users, i.e., for each entity which all attributes are being held by which user. However we note that, in practice this information gain is not significant, and known a prior [132]. The indexing scheme employed ensures that the identity of the entity remains unknown to the servers.

5.4.2 Phase Two: Secure K-means

At the end of the phase one, each computation server stores the secret shares (w.r.t. prime P_j) of the database $\mathcal{D}$. Since the scaling factor S was kept positive, the distance comparison in the original space will be equivalent to distance comparison in the new scaled space. Thus, the cluster assignment of the entities in the scaled space would be identical to what we would have expected in the original space. The final cluster locations are obtained from the cluster centers that are learned in the transformed space after appropriately scaling down and removing the introduced randomness.

Our algorithm will follow the same iterative structure as that of the standard K-means algorithm [95]. The objective is to cluster the data (available as secret shares), without leaking any information to any of the servers. RNS being doubly homomorphic, the operations of addition and multiplication can be independently carried out at each server. However division and comparison (both used in K-means) are difficult to do privately in the RNS. We overcome these difficulties by designing communicationlly efficient, privacy preserving protocols for them over one round of communication.

We now give a step by step description of the protocol used for phase two. *Note here*, that the N users are oblivious of algorithm and the data involved in phase two. The contribution of this work is not to improve upon the K-means algorithm as such but to propose an efficient protocol to privately carry out the clustering.

5.4.2.1 Step one: Initialization

Let $\vec{C}_1, \vec{C}_2, \cdots, \vec{C}_K$ be the K cluster centers, where each C_k is a D dimensional vector. The clusters are initialized as the K entities from the database $\mathcal{D}$ chosen in a pseudo-random fashion. Since, we want to keep the actual cluster locations also private, we thus store only their secret share components. i.e. for a cluster location $\vec{C}_k$, $1 \le k \le K$, the computational server j, $1 \le j \le R$, stores the vector $\vec{C}_{kj}$, where, $\vec{C}_{kj}$ is the secret share of $\vec{C}_k$ w.r.t. P_j.

The servers commonly choose the indices of K entities as the initial cluster centers. The secret shares of the chosen K entities, present with the servers, are used as the secret shares of the initial cluster centers C_k. That is, at server j, C_{kj} initialized to the secret share of the chosen entity. The pseudo-code of the algorithm is given in Algorithm 9.

Algorithm 9 PPKM: Initialization

```
1: for each cluster, $k$ = 1 to K do
2:     Choose a random entity index $l$, $l \le L$
3:     We want to initialize $\vec{C}_k = \vec{X}_l$, where $\vec{X}_l$ be the $D$ dimensional vector of entity $l$.
4:     for each server, $j$ = 1 to R do
5:         Let $\vec{X}_{lj}$ be the data corresponding to entity $l$ available with the server. We know $\vec{X}_{lj}$
           is shatter of $\vec{X}_l$ with mod $P_j$, and was stored with the server during phase one.
6:         Initialize, $\vec{C}_{kj}$ to $\vec{X}_{lj}$, where $C_{kj}$ is the shatter share of $C_k$ with mod $P_j$.
7:     end for
8: end for
```

Privacy: Servers do not learn any additional information of the data. The initialization is done, directly using the secret shares. This is done independently at each server, thus resulting in zero computation and communication overheads over TTP.

5.4.2.2 Step two: Lloyd Step

In an attempt to minimize the objective function, each iteration reclassifies and recomputes the new cluster locations. The algorithm terminates when it detects *'no change'* (defined by the termination criterion) in the cluster locations. Every iteration can be represented as a sequence of three steps as described below.

5.4.2.2.1 Finding Closest Cluster Centers: As stated before, since the scaling factor was set to a positive number, finding the closest point is equivalent to finding the one with the minimum of the distances squared in the scaled space. Thus, for every data entity $\vec{X}_l$, $1 \le l \le L$, we find the square of the Euclidean distance to each of the cluster centers $\vec{C}_k$. The distance square between two D dimensional vectors $\vec{X}$ and $\vec{Y}$, is defined as

$$\sum_{d=1}^{D}(X_d^2 + Y_d^2 + 2.X_d.Y_d) \tag{5.5}$$

which is a set of additions and multiplications. Now, RNS being doubly homomorphic, the above equation can be directly computed using the secret shares. Hence, every server can independently compute the respective secret shares of the distances between the L data points and the K cluster locations. For every data point $\vec{X_l}$, let $\vec{T_l}$ be the K length vector, whose share T_{lk} denotes the distance square between data point $\vec{X_l}$ and cluster center $\vec{C_k}$. The task is to, without actually reconstructing, compute T_{lk} from the shatter shares of $\vec{X_l}$ and to assign the point $\vec{X_l}$ to a closest cluster k.

T_{lk} is represented in the RNS such that T_{lkj} denotes the secret share of T_{lk} (w.r.t. P_j) available at server j. Now, each of the server j can use the Equation 5.5 to compute the share (T_{lkj}) using its locally available secret shares of $\vec{X}_{lj}$ and $\vec{C}_{kj}$.

Next, for each data point l, we need to find the cluster k such that T_{lk} is minimum. This would require reconstructing and comparing T_{lk}'s. However, to maintain privacy, the actual distances, T_{lk}'s should be kept private. We overcome this dilemma by applying a clever permutation and randomization scheme. $\vec{T}_{lk}$ is secured by applying another layer of randomization on the secret shares before sending them over for comparison to another untrusted server (thresholder). Finding the minimum of the K numbers is an $O(K)$ algorithm, i.e. the current minimum has to be compared against the next potential candidate. We next describe the protocol to find the minimum of two numbers, Z_1 and Z_2. This can then be repeated $K-1$ times to find the minimum of K numbers.

Finding the minimum: $(Z_1 - Z_2) \leq 0$ implies $Z_1 \leq Z_2$ else otherwise. In-order to check for this, at each server, we can compute the difference $Z_{1j} - Z_{2j}$ and send over the difference shares to an untrusted server for reconstruction and comparison. However, this naive approach reveals to the thresholder the distance between the two data points. We secure this by randomizing the secret shares of the differences before sending it over for comparison. We can even keep the random number itself unknown to any of the servers by the following protocol.

Each of the R servers chooses a random number r_i and sends over $r_i \ mod \ P_j$ to server j. Thus, each server j, has $\sum_{i=1}^{i=R} r_i \ \% \ P_j$ or $r \ \% \ P_j$, where $r = \sum_1^R r_i$ (Algorithm 10: steps 5-12). The servers uses this to randomize its share of difference. The randomized difference shares are then sent over to an un-trusted server who reconstructs the randomized difference and returns the comparison against zero for finding the minimum of the two. The smaller number is then compared against the next potential candidate. After a series of $K-1$ comparisons a data point is confidently and privately assigned to a nearest cluster center. Note that the communication costs can further be reduced by choosing the random numbers offline, i.e. when the systems are idle. Each server maintains the list of the secret shares of the random numbers, r's used in the final protocol.

Algorithm 10 Find Minimum of K Numbers Protocol

```
1: Let $Z_1$, $Z_2$, ... $Z_K$ be the $K$ numbers we want to find minimum of
2: R is the number of computational servers, each knowing $Z_{kj}$, for $1 \leq k \leq K$ and $1 \leq j \leq R$, where $Z_{kj}$ is the shatter share of $Z_k$ with mod $P_j$. Note that the actual value of $Z_k$ is kept secret from all the servers.
3: Initialize $minIndex$ = 1
4: for every index, k = 2 to K do
5:     for every server, j = 1 to R do
6:         Select a positive random number $r_j$ and share the modulo of $r_j$ with every other server (step 7).
7:         for every other server: i = 1 to R do
8:             Send $r_{ji}$ = $r_j$ mod $P_i$ to the server $i$.
9:         end for
10:    end for
11:    for every server, j = 1 to R do
12:        Let $r'_j$ be the summation of the $R$ random numbers received at each server $j$.
13:        Compute the difference of the secret shares of $Z_{minIndex}$ and $Z_k$. Randomize the difference by multiplying with $r'_j$.
14:        The randomized difference share is sent over to the thresholder.
15:    end for
16:    Thresholder applies the merge function to obtain $R'.(Z_{minIndex} - Z_k)$, where $R'$ is the summation of R positive random numbers $r_j$. The randomized difference is compared with 0 and the result sent back to the servers.
17:    if Threshold Result $> 0$ then
18:        minIndex = k
19:    end if
20:    For next iteration, the role of the thresholder is switched to another pseudo-randomly chosen server.
21: end for
22: Return $min\_index$
```

Correctness: Consider a point $\vec{X}$, for which we want to find which is closer $\vec{Y}$ or $\vec{Z}$. Let the points be *shattered* with scale S and randomization $\vec{a}$, $\vec{b}$ and $\vec{c}$ respectively. Thus, we have:

$$(X_1, X_2, \cdots, X_D) \rightarrow (S \cdot X_1 + a_1, \cdots, S \cdot X_D + a_D) \tag{5.6}$$

$$(Y_1, Y_2, \cdots, Y_D) \rightarrow (S \cdot Y_1 + b_1, \cdots, S \cdot Y_D + b_D) \tag{5.7}$$

$$(Z_1, Z_2, \cdots, Z_D) \rightarrow (S \cdot Z_1 + c_1, \cdots, S \cdot Z_D + c_D) \tag{5.8}$$

Let us assume Y is closer than Z, then following holds:

$$\sum(X_i - Y_i)^2 \leq \sum(X_i - Z_i)^2 \tag{5.9}$$

Using the secret shares, the corresponding distances in the scaled space are computed as:

$$Dist_1 = \sum(S(X_i - Y_i) + (a_i - b_i))^2 \tag{5.10}$$

$$Dist_2 = \sum(S(X_i - Z_i) + (a_i - c_i))^2 \tag{5.11}$$

Given that Equation 5.9 holds, the protocol is correct if $Dist_1 \leq Dist_2$. From the constraints given in Section 5.4.1, we know $0 \leq a_i, b_i, c_i \leq S/2$, thus we get $-S/2 \leq (a_i - b_i) \leq S/2$.

$$\sum(S(X_i - Y_i - 1/2))^2 \leq Dist_1 \leq \sum(S(X_i - Y_i + 1/2))^2 \tag{5.12}$$

$$\sum(S(X_i - Z_i - 1/2))^2 \leq Dist_2 \leq \sum(S(X_i - Z_i + 1/2))^2 \tag{5.13}$$

Thus, the protocol satisfies correctness if Equation 5.14 is true whenever Equation 5.9 is true.

$$\sum(S(X_i - Y_i + 1/2))^2 \leq \sum(S(X_i - Z_i - 1/2))^2 \tag{5.14}$$

This will hold if the Cartesian System is designed so as to nullify the effect of the additional $\pm 1/2$ in Equation 5.14. This is achieved by having the step-size in the Cartesian system as 2, i.e. the data is scaled by 2 before choosing the parameters (Section 5.4.1).

Privacy: The protocol is secure against both the GCD and factorization based attacks. The servers are made to jointly choose the randomization, which is different for every threshold operation. This ensures security against the factorization based attacks. The role of the thresholder is also switched among the R servers in an random order, thus ensuring security against the GCD based attacks.

5.4.2.2.2 Updating Cluster Locations: Once each of the L data points has been assigned to one of the K clusters, the next step is to recompute the cluster locations. For every cluster k, the cluster center is updated to the center of mass of the newly assigned points to the cluster. Thus, the new coordinate of the cluster k is a (weighted) mean of the corresponding coordinates of the n_k points assigned to the cluster k. Let n_k be the number of data points assigned to cluster k. For any cluster k, each server stores the secret shares of the data points. Each server j, can thus independently compute the sum (Sum_{kdj}) using the secret shares of the n_k data points. The updated cluster location is then obtained by dividing the sum of co-ordinates by n_k. However as we know that the generic division is not defined in the RNS, therefore we cannot directly divide the sum's shares. Furthermore, so as to maintain complete privacy, we will like to keep the updated cluster locations unknown from all the servers. Therefore, an interactive protocol, similar to the one used for thresholding is employed for the job. We now describe the *privacy-preserving division protocol (PPDP).*

PPDP: Consider a number X, secret shares of which are stored at the R servers. The task is to privately divide X by n, such that the secret X and the quotient $q = \lfloor \frac{X}{n} \rfloor$ is kept private from all of the servers. At the end of the protocol, all that the server j gets is the secret share of q w.r.t. P_j. PPDP is achieved through a single round of interaction, and the secret data, X, is secured using a permutation and a randomization method.

Just as in previous protocol (Algorithm 10, steps 5-12), the R servers jointly computes two random numbers r and r', such that server j knows only the shares of them. Each server now randomizes its share of X according to Equation 5.15, before sending it over to an un-trusted server. As in the previous protocol, this server is switched among the R servers in a pseudo-permutation fashion. The randomized shares are then reconstructed using the *merge* function to compute X' (Equation 5.15).

Division is then performed to compute the randomized quotient q', as given by Equation 5.17, where q is the actual quotient that we wish to compute (Equation 5.16). We next compute the secret shares of q' and sends them over to the specific servers for de-randomization. Each server computes its share of quotient, q_j, from q'_j using Equation 5.18. The secret share of the cluster center is then updated to the computed share of the quotient. The pseudo-code of the protocol is given in Algorithm 11.

Algorithm 11 Privacy Preserving Division Protocol (PPDP)

```
1:  $R$ computational servers, stores i) $X_j$ = shatter of $X$ with mod $P_j$, ii) n
2:  Randomly select $r$, $r'$, in the manner similar to as described in steps(5-12) of algorithm ??.
3:  Let at each server $j$, $r_j$, $r'_j$ be the shatter shares of the two chosen random numbers $r$ and
    $r'$.
4:  for each server, j = 1 to R do
5:      Compute $X'_j = r_j \cdot (X_j + r'_j \cdot n)\ mod\ P_j$
6:      Send $X'_j$ to the thresholder (switched among servers in a pseudo random order).
7:  end for
8:  Thresholder uses the merge function to compute $X'$
9:  Compute $q' = \lfloor \frac{X'}{n} \rfloor$
10: Send over the $q'_j$ to server $j$, where $q'$ is the shatter share of $q'$ with mod $P_j$.
11: for each server, j = 1 to R do
12:     De-randomize the received quotient to get $q_j = (q'_j * r_j^{-1} - r'_j)\ mod\ P_j$
13: end for
14: Now, $q_j$ is the required shatter share of the quotient, $q$, with prime $P_j$.
```

$$X \rightarrow X' = r \cdot (X + r' \cdot n) \tag{5.15}$$

$$q = \frac{X}{n} \tag{5.16}$$

$$q' = \frac{X'}{n} = r \cdot (q + r') \tag{5.17}$$

$$q_j = (q'_j * r_j^{-1} \ - r'_j) \ mod \ P_j \tag{5.18}$$

Privacy: The PPDP method provides high level of privacy for the secret data. The randomization parameters r and r' are jointly chosen and remains unknown to all. The randomization of the secret data, X, is itself done using the secret shares. The randomization function (Equation: 5.15) is designed so as to safeguard against the potential attacks such as factorization and GCD based. In the entire process, no additional meaningful information is leaked to any one. The method not only provides provable privacy but is also efficient with communication cost limited to one round of interaction.

5.4.2.2.3 Checking Termination Criterion: At the end of every iteration, we check for the closeness of the new clusters. The 'closeness' is defined as *i)* minimizing the total energy of the clusters, the energy of a cluster k is given as E_k = $\sum_1^{n_k}(\|\vec{x}_l - \vec{c}_l\|)$, *ii)* the new clusters locations are close to the old ones. i.e $\sum_1^{K}(\|\vec{c}_k - \vec{c}'_k\|)$, or iii) the number of points making transition across clusters is small.

If the closeness is below the threshold, then we go to step three otherwise continue with next iteration. Any of these definitions can be privately implemented using the approaches like already described.

5.4.2.3 Step three: Knowledge Revelation

At the termination of the Lloyd step, the cluster centers are stored as the secret shares at the R serves. The cluster assignment of the anonymized entities is also available. To learn the cluster locations, the servers are made to collude under legal agreements. The identity of the entities is known only to the data owner, and hence he is the only one who learns the final cluster assignment. The cluster locations can be revealed, only if agreed upon.

5.5 Cost Analysis

We analyze the overheads of one iteration of the algorithm. The total cost depends on the number of iterations required to converge, which is dependent on the termination criterion. The overheads are computed against the naive TTP based protocol(i.e. sending the data in plain to a trusted server). A comparison is also drawn against those using the primitives such as homomorphic encryption or SMC.

5.5.1 Communication Cost

The overheads incurred are a result of the interaction among the servers needed for the operations of division and comparison during the Lloyd step. In our solution, every comparison and division requires just one round of communication. For L entities in a D dimensional space and K clusters this translates to $(K-1) \cdot L$ comparisons and $K \cdot D$ division operations per iteration. Thus requiring $(K-1) \cdot L + K \cdot D$ rounds of communication per iteration.

We now *compare against the traditional protocols*. The approaches suggested in literature uses computationally intensive interactive protocols to implement secure multiplication and comparison. The common basic tool used is Oblivious Transfer (OT), which in turn is used for secure circuit evaluation. The communication cost is linear in the number of multiplication gates in the circuit. In K-means, the number of gates for an operation of multiplication, division and comparison is linear in number of bits. Moreover, each round of OT is also computationally expensive as it involves $O(log(W))$ PKC encryption/decryption subroutines.

Compared to this, our shattering based solution, which is defined in RNS, is doubly homomorphic. Thus, enabling secure multiplication without any communication overhead. In practice this is a huge gain over SMC. Further, the interaction is limited to just one round of communication for both division and comparison operations. Thus, introducing the paradigm of shattering and merging significantly reduces the overhead costs over the traditional privacy preserving clustering solutions.

5.5.2 Data Expansion

Securing the data as secret shares results in a data expansion. An optimal selection of the parameters is discussed in Section 5.4.1. Each attribute which requires $log(W)$ bits of storage in the plain domain is shattered to a total size of $\frac{R}{R-2} \cdot log(W)$ bits. On the other-hand, using the standard SS scheme would lead to a total size of $R \cdot log(W)$ bits. Thus, shattering operation gains by a factor of $R-2$ over the standard scheme. The data expansion is critical not only due to the storage costs but also because it determines the number of bits required per round of communication. For example, a 32 bits data shattered into 5 shares requires 54 bits, while using the standard scheme would requires 160 bits of storage. Another advantage is in the performance gain as faster computations are possible for attributes represented using less number of bits.

5.6 Discussion

We propose a novel ‘cloud computing’ based solution using the paradigm of Secret Sharing to privately cluster an arbitrary partitioned data among N users. Traditional approaches uses primitives such as SMC or PKC, thus compromising the efficiency of the solutions and in return provide very high level of privacy which is usually an overkill in practice. This work contributes at ways of looking at things differently. We show that privacy need not be always at the cost of efficiency.

We exploit the properties of the data and the problem to circumvent the limitations faced by traditional methods (that are general-purpose). Our solution does not demand any trust among the servers or users. Security is based on the standard assumptions of honest-but-curious, non-colluding servers having ability to generate random numbers. As expected, the protocol is costly compared to the one with zero-security. However, the additional costs are kept to a minimum and are negligible compared to those of SMC. Unlike SMC, in our method interaction is limited to one round per division and comparision and is reasonable for a practical deployment.

With the RNS being doubly homomorphic, the paradigm of shattering and merging is generic and has potential to extend over to even more diverse data mining applications.

Chapter 6

Conclusions

In this thesis, we introduced efficient privacy preserving protocols for processing visual data. Traditionally, generic cryptographic primitives such as TTP, PKC, SMC etc., have been employed for ensuring the security and privacy of sensitive data. However, we show that the associated computation and communication overhead are significantly high, making such approaches of limited practical interest. In light of this, a few solutions have recently been proposed to improve the efficiency by making a tradeoff in privacy and accuracy.

The work in this thesis opens up a new avenue for practical and provable secure implementations of vision algorithms, that rely on distribution of data over multiple computers. Broadly, we address the scenarios, where a service provider Bob, lends the processing power and algorithms to clients. However, for many practical applications, Bob may not wish to make his proprietary algorithms public, while a client himself may not be willing to reveal his private data to anyone, including the processing server. This is closely related to secure multi-party computation (SMC) problem in cryptography. In this work, we propose application specific, computationally efficient and provably secure computer vision algorithms for the encrypted domain. In designing the algorithms, we addresses the issues of *efficacy* and *efficiency* by utilizing the domain specific knowledge.

In our first work, *blind authentication*, we propose private biometric authentication protocol which is extremely secure under a variety of attacks and can be used with a wide variety of biometric traits. The primary advantage of the proposed approach is the ability to achieve classification of a strongly encrypted feature vector using generic classifiers such as Neural Networks and SVMs. In fact, the authentication server need not know the specific biometric trait that is used by a particular user, which can even vary across users. Once a trusted enrollment server encrypts the classifier parameters for a specific biometric of a person, the authentication server is verifying the identity of a user with respect to that encryption. The real identity of the person is hence not revealed to the server, making the protocol, completely blind. This allows one to revoke enrolled templates by changing the encryption key, as well as use multiple keys across different servers to avoid being tracked, thus leading to better privacy.

We then present an efficient, practical and highly secure framework for implementing visual surveillance on untrusted remote computers. The challenge of introducing privacy and security in such a practical surveillance system has been stifled by the enormous computational and communication overhead required by the solutions. To achieve this, we demonstrate that the properties of visual data can be exploited to break the bottleneck of computational and communication overheads. This change in view allows us to have a simplified capture device, an efficient unidirectional data flow, and surveillance operations performed directly on the shattered streams. Only the surveillance results will be available to the observer. Our method enables distributed secure processing and storage, while retaining the ability to reconstruct the original data in case of a legal requirement. Such an architecture provides us both security as well as computation and communication efficiency.

We next extend our proposed paradigm to achieve the ability to do un-supervised learning using K-means in the encrypted domain. We use the paradigm of *secret sharing*, which allows the data to be divided into multiple shares and processed separately at different servers. Using the paradigm of secret sharing, allows us to design a provably-secure, cloud computing based solution, which has negligible communication overhead compared to SMC and is hence over a million times faster than similar SMC based protocols. Our proposed solution is not only computationally efficient but also secure independent of whether or not $P \neq NP$. Our paradigm is generic and has the potential to extend over to even more diverse data mining applications.

In future, one could further extent the approach to make it capable of implementing generic vision algorithms. We need to explore the possible extensions of the work to other domains and design solutions addressing issues mentioned in Section 1.1.

Related Publications

Journal Papers

- Maneesh Upmanyu, Anoop M. Namboodiri, K. Srinathan, and C. V. Jawahar.
  Blind authetication: A secure crypto-biometric verication protocol,
  In *IEEE-Transactions on Information Forensics and Security (TIFS)*, 2010.
- Maneesh Upmanyu, Anoop M. Namboodiri, K. Srinathan, and C. V. Jawahar.
  Efficient privacy preserving video surveillance,
  In *IEEE-Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, (under-submission).

Conference Papers

- Maneesh Upmanyu, Anoop M. Namboodiri, K. Srinathan, and C. V. Jawahar.
  Efficient privacy preserving K-Means Clustering,
  In *Pacific Asia Workshop on Intelligence and Security Informatics (PAISI)*, 2010.
- Maneesh Upmanyu, Anoop M. Namboodiri, K. Srinathan, and C. V. Jawahar.
  Efficient privacy preserving video surveillance,
  In *Twelfth International Conference on Computer Vision (ICCV)*, 2009.
- Maneesh Upmanyu, Anoop M. Namboodiri, K. Srinathan, and C. V. Jawahar.
  Efficient biometric verication in the encrypted domain,
  In *Third International Conference on Biometrics (ICB)*, 2009.

Bibliography

[1] Casia iris dataset. http://www.cbsr.ia.ac.cn/english/Databases.asp.

[2] Feret database. http://www.itl.nist.gov/iad/humanid/feret/feret_master.html.

[3] Fvc2002 dataset. http://bias.csr.unibo.it/fvc2002/databases.asp.

[4] Fvc2004 dataset. http://bias.csr.unibo.it/fvc2004/databases.asp.

[5] Gnu multiple precision arithmetic library. http://gmplib.org/.

[6] Nist special database 4. http://www.nist.gov/srd/nistsd4.htm.

[7] The orl database of faces. http://www.cl.cam.ac.uk/research/dtg/ attarchive/facedatabase.html.

[8] Secret sharing. http://en.wikipedia.org/wiki/Secretsharing.

[9] Walk the walk: Gait recognition technology could identify humans at a distance. http://gtresearchnews.gatech.edu/newsrelease/GAIT.htm.

[10] Xyssl. http://linux.softpedia.com/get/Security/XySSL-19360.shtml.

[11] Yale face database. http://cvc.yale.edu/projects/yalefaces/yalefaces.html.

[12] How to exchange secrets with oblivious transfer, 1981. Aiken Computation Laboratory.

[13] Proceedings of Worshop on Biometrics (CVPR), 2006,07.

[14] Shigeo Abe. *Support Vector Machines For Pattern Classification*. Springer, 2005.

[15] I. Agi and L. Gong. An empirical study of MPEG video transmission. In *Proceedings of the Internet Society Symposium on Network and Distributed Systems Security*, pages 137–144, 1996.

[16] Dakshi Agrawal and Charu C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *PODS*, pages 247–255. ACM, 2001.

[17] Rakesh Agrawal and Ramakrishnan Srikant. Privacy-preserving data mining. *SIGMOD*, 29(2):439–450, 2000.

[18] A.M. Alattar and G.I. Al-Regib. Evaluation of selective encryption techniques for secure transmission of mpeg-compressed bit-streams. volume 4, pages 340–343, Jul 1999.

[19] A.M. Alattar, G.I. Al-Regib, and S.A. Al-Semari. Improved selective encryption techniques for secure transmission of mpeg video bit-streams. volume 4, pages 256–260, 1999.

[20] C.A. Asmuth and J. Bloom. A modular approach to key safeguarding. *IEEE Transactions on Information Theory*, 29:208–210, 1983.

[21] S. Avidan, A. Elbaz, and T. Malkin. Privacy preserving pattern classification. *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, pages 1684–1687, 2008.

[22] Shai Avidan and Moshe Butman. Blind vision. In *European Conference on Computer Vision (ECCV)*, pages 1–13, 2006.

[23] Shai Avidan and Moshe Butman. Efficient methods for privacy preserving face detection. In *NIPS*, pages 57–64, 2006.

[24] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, 2001.

[25] Amos Beimel and Benny Chor. Universally ideal secret sharing schemes (preliminary version). In *CRYPTO*, pages 183–195, 1993.

[26] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *STOC*, pages 1–10. ACM, 1988.

[27] J. Cohen Benaloh. Secret sharing homomorphisms: keeping shares of a secret secret. *CRYPTO*, 283:251–260, 1986.

[28] John Bethencourt. Paillier library. http://acsc.csl.sri.com/libpaillier/.

[29] C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[30] Dmitri Bitouk, Neeraj Kumar, Samreen Dhillon, Peter Belhumeur, and Shree K. Nayar. Face swapping: automatically replacing faces in photographs. *ACM Trans. Graph.*, 27(3):1–8, 2008.

[31] G. R. Blakley. Safeguarding cryptographic keys. *Proceedings of the National Computer Conference*, pages 313–317, 1979.

[32] V. Blanz and T. Vetter. Face recognition based on fitting a 3d morphable model. *IEEE Transactions on Pattern Analysis & Machine Intelligence (PAMI)*, 25-9:1063–1074, 2003.

[33] Thomas Blum and Christof Paar. High-radix montgomery modular exponentiation on reconfigurable hardware. *IEEE Transactions on Computers*, 50(7):759–764, 2001.

[34] M.N. Bojnordi, M.R. Hashemi, and S.O. Fatemi. Implementing an efficient encryption block for mpeg video streams. pages 127–130, June 2005.

[35] T.E. Boult, W.J. Scheirer, and R. Woodworth. Revocable fingerprint biotokens: Accuracy and security analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, June 2007.

[36] Gilles Brassard, Claude Crépeau, and Jean-Marc Robert. All-or-nothing disclosure of secrets. In *Proceedings on Advances in cryptology—CRYPTO '86*, pages 234–238, 1987.

[37] Paul Bunn and Rafail Ostrovsky. Secure two-party k-means clustering. In *CCS*, pages 486–497. ACM, 2007.

[38] A. Cavallaro. Privacy in video surveillance [in the spotlight]. *Signal Processing Magazine, IEEE*, 24(2):168–166, March 2007.

[39] Anna Ceguerra and Irena Koprinska. Automatic fingerprint verification using neural networks. In *International Conference on Artificial Neural Networks (ICANN)*, pages 1281–1286, 2002.

[40] Antoni B. Chan, Zhang-Sheng John Liang, and Nuno Vasconcelos. Privacy preserving crowd monitoring: Counting people without people models or tracking. In *CVPR*, pages 1–7, June 2008.

[41] A. Chattopadhyay and T.E. Boult. Privacycam: a privacy preserving camera using uclinux on the blackfin dsp. *CVPR*, pages 1–8, June 2007.

[42] L. S. Choon, A. Samsudin, and R. Budiarto. Lightweight and cost-effective MPEG video encryption. In *Proceedings of Information and Communication Technologies: From Theory to Applications*, pages 525–526, 2004.

[43] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *FOCS '95: Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, page 41, 1995.

[44] B. Chor and E. Kushilevitz. A zero-one law for boolean privacy. In *STOC*, pages 62–72, 1989.

[45] Tee Connie, Andrew Teoh, Michael Goh, and David Ngo. PalmHashing: a novel approach for cancelable biometrics. *Information Processing Letters*, 93(1):1–5, January 2005.

[46] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, , and Clifford Stein. The chinese remainder theorem. In *Introduction to Algorithms*, pages 873–876. MIT Press, McGraw-Hill, 2001.

[47] Lorrie Faith Cranor. Internet privacy. *Commun. ACM*, 42(2):28–38, 1999.

[48] R. Davis. The data encryption standard in perspective. *Communications Magazine, IEEE*, 16:5–9, 1978.

[49] Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *PODS*, pages 202–210. ACM, 2003.

[50] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *Eurocrypt*, pages 523–540, 2004.

[51] R. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, 1973.

[52] Cynthia Dwork, Frank Mcsherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, volume 3876, pages 265–284, 2006.

[53] Cynthia Dwork and Kobbi Nissim. Privacy-preserving datamining on vertically partitioned databases. In *CRYPTO*, volume 3152, pages 528–544, 2004.

[54] Hazem El-Bakry. Fast iris detection for personal verification using modular neural nets. In *Proceedings of the International Conference, 7th Fuzzy Days on Computational Intelligence, Theory and Applications*, pages 269–283, 2001.

[55] Taher El Gamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[56] Ortega-Garcia J. et al. Mcyt baseline corpus: a bimodal biometric database, 2003.

[57] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, 1985.

[58] Faisal Farooq, Ruud M. Bolle, Tsai-Yang Jea, and Nalini Ratha. Anonymous and revocable fingerprint recognition. In *CVPR Biometrics Worshop*, pages 1–7, June 2007.

[59] Marcos Faundez-Zanuy, David A. Elizondo, Miguel Ángel Ferrer-Ballester, and Carlos M. Travieso-González. Authentication of individuals using hand geometry biometrics: A neural network approach. *Neural Process. Lett.*, 26(3):201–216, 2007.

[60] Tom Fawcett. An introduction to roc analysis. *Pattern Recogn. Lett.*, 27(8):861–874, 2006.

[61] Caroline Fontaine and Fabien Galand. A survey of homomorphic encryption for nonspecialists. *EURASIP*, 1:1–15, 2007.

[62] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 1990.

[63] Craig Gentry. Fully homomorphic encryption using ideal lattices. *STOC*, pages 169–178, 2009.

[64] O. Goldreich, D. Ron, and M. Sudan. Chinese remaindering with errors. *IEEE Transactions on Information Theory*, 46:1330–1338, 2000.

[65] Oded Goldreich. *The Foundations of Cryptography - Volume 2*. Cambridge University Press, May 2004.

[66] G. Guo, S.Z. Li, and K. Chan. Face recognition by support vector machines. *ICAFGR*, pages 196–201, march 2000.

[67] B. Heisele, P. Ho, and T. Poggio. Face recognition with support vector machines: Global versus component-based approach. 2:688–694, July 2001.

[68] Alexander Hornberg. *Handbook of Machine Vision*. Wiley-VCH, 2006.

[69] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Netw.*, 2(5):359–366, 1989.

[70] Ali Inan, Selim V. Kaya, Yucel Saygin, Erkay Savas, Ayca A. Hintoglu, and Albert Levi. Privacy preserving clustering on horizontally partitioned data. *Data Knowl. Eng.*, 63(3):646–666, 2007.

[71] Ioannis Ioannidis, , Ioannis Ioannidis, and Ananth Grama. An efficient protocol for yao's millionaires' problem. In *In Proceedings of the 36th Hawaii Internatinal Conference on System Sciences*, pages 6–9, 2003.

[72] Geetha Jagannathan and Rebecca N. Wright. Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In *KDD*, pages 593–599. ACM, 2005.

[73] Anil K. Jain, Karthik Nandakumar, and Abhishek Nagar. Biometric template security. *EURASIP*, 8(2):1–17, 2008.

[74] Anil K. Jain, Arun Ross, and Sharat Pankanti. A prototype hand geometry-based verification system. In *In 2nd Int'l Conference on Audio- and Video-based Biometric Person Authentication (AVBPA)*, pages 166–171, 1999.

[75] Anil K. Jain, Arun Ross, and Salil Prabhakar. An introduction to biometric recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(1):4–20, January 2004.

[76] Anil K. Jain and Umut Uludag. Hiding biometric data. *IEEE Transactions on Pattern Analysis & Machine Intelligence (PAMI)*, 25(11):1494–1498, November 2003.

[77] S. Jha, L. Kruger, and P. Mcdaniel. Privacy preserving clustering. In *ESORICS*, pages 397–417, 2005.

[78] Thorsten Joachims. Svm-light. http://svmlight.joachims.org/.

[79] Ari Juels and Madhu Sudan. A fuzzy vault scheme. *Designs, Codes and Cryptography*, 38(2):237–257, 2006.

[80] Liu Jun, Zou LingLing, Xie Changsheng, and Huang Hao. A two-way selective encryption algorithm for mpeg video. In *IWNAS '06: Proceedings of the 2006 International Workshop on Networking, Architecture, and Storages*, pages 183–187, 2006.

[81] H. Kargupta, S. Datta, Q. Wang, and Krishnamoorthy Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *ICDM*, pages 99–106, 2003.

[82] Joe Kilian. Founding crytpography on oblivious transfer. In *STOC*, pages 20–31, 1988.

[83] Andrej KiselL, Alexej Kocochetkov, and Justas Kranauskas. Fingerprint minutiae matching without global alignment using local structures. *INFORMATICA*, 19(1):31–44, 2008.

[84] A. Kong, K.H. Cheung, D. Zhang, M. Kamel, and J. You. An analysis of biohashing and its variants. *Pattern Recognition*, 39(7):1359–1368, July 2006.

[85] Zsolt Miklós Kovács-Vajna. A fingerprint verification system based on triangular matching and dynamic time warping. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 22(11):1266–1276, 2000.

[86] D. Kumar, A.; Zhang. Hand-geometry recognition using entropy-based discretization. *IEEE Transactions on Information Forensics and Security*, 2(2):181–187, 2007.

[87] Yuan Li, Liwei Liang, Zhaopin Su, and Jianguo Jiang. A new video encryption algorithm for h.264. pages 1121–1124, 2005.

[88] Yehuda Lindell and Benny Pinkas. Privacy preserving data mining. In *CRYPTO*, pages 36–54, 2000.

[89] Yehuda Lindell and Benny Pinkas. Secure multiparty computation for privacy-preserving data mining. Cryptology ePrint Archive, Report 2008/197, 2008.

[90] Helger Lipmaa. An oblivious transfer protocol with log-squared communication. *Information Security*, 3650/2005:314–328, 2005.

[91] Kun Liu, Chris Giannella, and Hillol Kargupta. A survey of attack techniques on privacy-preserving data perturbation methods. *Privacy-Preserving Data Mining*, 34:15:359–381, 2008.

[92] T. B. Maples and G. A. Spanos. Performance study of a selective encryption scheme for the security of networked, real-time video. In *Proceedings of Fourth International Workshop on Multimedia Software Development)*, 1995.

[93] Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, October 1996.

[94] Maurice Mignotte. How to share a secret. *CRYPTO*, 1983.

[95] Tom Mitchell. *Machine Learning*. The McGraw-Hill, 1997.

[96] Donald M. Monro, Soumyadip Rakshit, and Dexin Zhang. Dct-based iris recognition. *IEEE Transactions on Pattern Analysis & Machine Intelligence (PAMI)*, 29(4):586–595, 2007.

[97] Kei Nagai, Hiroaki Kikuchi, Wakaha Ogata, and Masakatsu Nishigaki. ZeroBio: Evaluation and development of asymmetric fingerprint authentication system using oblivious neural network evaluation protocol. In *The Second International Conference on Availability, Reliability and Security (ARES)*, pages 1155–1159, April 2007.

[98] Victor-Emil Neagoe. New self-organizing maps with non-conventional metrics and their applications for iris recognition and automatic translation. In *ICCOMP*, pages 145–151. WSEAS, 2007.

[99] E. Newton, L. sweeney, and B. Malin. Preserving privacy by de-identifying facial images. *IEEE Transactions on Knowledge and Data Engineering*, 17(2):232–243, February 2005.

[100] Stanley R. M. Oliveira. Privacy preserving clustering by data transformation. In *18th Brazilian Symposium on Databases*, pages 304–318, 2003.

[101] C. Orlandi, A. Piva, and M. Barni. Oblivious neural network computing via homomorphic encryption. *EURASIP*, 2007-1:1–10.

[102] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. *Eurocrypt*, pages 223–238, 1999.

[103] Benny Pinkas. Cryptographic techniques for privacy-preserving data mining. *SIGKDD Explorations*, 4(2):12–19, 2002.

[104] L. Qiao and Klara Nahrstedt. A new algorithm for MPEG video encryption. In *Proceedings of First International Conference on Imaging Science System and Technology*, pages 21–29, 1997.

[105] Michal Quisquater, Bart Preneel, and Joos Vandewalle. On the security of the threshold scheme based on the chinese remainder theorem. *Public Key Cryptography*, pages 199–210, 2002.

[106] D. Rappe. Homomorphic cryptosystems and their applications. In *Ph.D. dissertation, University of Dortmund*, 2004.

[107] Vit Niennattrakul;Dachawut Wanichsan;Chotirat Ann Ratanamahatana. Hand geometry verification using time series representation. Sep 2007.

[108] Nalini K. Ratha, Jonathan H. Connell, and Ruud M. Bolle. Enhancing security and privacy in biometrics-based authentication systems. *IBM Systems Journal*, 40(3):614–634, March 2001.

[109] N.K. Ratha, S. Chikkerur, J.H. Connell, and R.M. Bolle. Generating cancelable fingerprint templates. *IEEE Transactions on Pattern Analysis & Machine Intelligence (PAMI)*, 29(4):561–572, April 2007.

[110] Rijndael. The advanced encryption standard in perspective NIST, FIPS 197. 2001.

[111] Ronald Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[112] Ronald L. Rivest. The RC5 encryption algorithm. In *Proceedings of the second international workshop on fast software encryption (FSE)*, pages 86–96, 1994.

[113] Kaushik Roy and Prabir Bhattacharya. Iris recognition with support vector machines. In *International Conference on Biometrics (ICB)*, pages 486–492, 2006.

[114] M. Savvides and B.V.K. Vijaya Kumar. Cancellable biometric filters for face recognition. *International Conference on Pattern Recognition (ICPR)*, 3:922–925, 2004.

[115] Walter J. Scheirer and Terrance E. Boult. Bipartite biotokens: Definition, implementation, and analysis. In *International Conference on Biometrics (ICB)*, pages 775–785, 2009.

[116] A. Senior, S. Pankanti, A. Hampapur, L. Brown, Ying-Li Tian, A. Ekin, J. Connell, Chiao Fe Shu, and M. Lu. Enabling video privacy through computer vision. *Security and Privacy, IEEE*, 3(3):50–57, May-June 2005.

[117] Lifeng Sha, Feng Zhao, and Xiaoou Tang. Minutiae-based fingerprint matching using subset combination. *ICPR*, pages 566–569, 2006.

[118] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[119] J. Shashank, P. Kowshik, Kannan Srinathan, and C.V. Jawahar. Private content based image retrieval. In *CVPR*, 2008.

[120] C. Shi and Bharat Bhargava. A fast MPEG video encryption algorithm. In *Proceedings of ACM Multimedia*, 1998.

[121] Gustavus J. Simmons. How to (really) share a secret. In *CRYPTO*, pages 390–448, 1990.

[122] T. Spindler, C. Wartmann, L. Hovestadt, D. Roth, Luc VanGool, and A. Steffen. Privacy in video surveilled spaces. *Journal of Computer Security*, 16(2):199–222, January 2008.

[123] William Stallings. *Cryptography and Network Security*. Pearson Prentice Hall, 2006.

[124] N.S. Szabo and R. I. Tanaka. *Residue Arithmetic and its Applications to Computer Technology*. McGraw-Hill, 1967.

[125] L. Tang. Methods for encrypting and decrypting MPEG video data efficiently. In *Proceedings of ACM Multimedia*, pages 219–229, 1996.

[126] A.B.J. Teoh, D.C.L. Ngo, and A. Goh. Biohashing: Two factor authentication featuring fingerprint data and tokenised random number. *Pattern Recognition*, 37(11):2245–2255, November 2004.

[127] Andrew Teoh, Beng Jin, Tee Connie, David Ngo, and Check Ling. Remarks on BioHash and its mathematical foundation. *Information Processing Letters*, 100(4):145–150, November 2006.

[128] Joseph Turow. Americans and online privacy: The system is broken. *Technical Report*, June 2003.

[129] Z.D. Ulman. Sign detection and implicit-explicit conversion of numbers in residue arithmetic. *Computers, IEEE Transactions on*, C-32(6):590–594, June 1983.

[130] Umut Uludag, Sharat Pankanti, Salil Prabhakar, and Anil K. Jain. Biometric cryptosystems: Issues and challenges. *Proceedings of the IEEE*, 92(6):948–960, June 2004.

[131] J. Vaidya and C. Clifton. Privacy-preserving data mining: why, how & when. *Security & Privacy*, pages 19–27, 2004.

[132] Jaideep Vaidya and Chris Clifton. Privacy-preserving k-means clustering over vertically partitioned data. In *KDD*, pages 206–215. ACM, 2003.

[133] Vassilios S. Verykios, Elisa Bertino, Igor Nai Fovino, Loredana Parasiliti Provenza, Yucel Saygin, and Yannis Theodoridis. State-of-the-art in privacy preserving data mining. *SIGMOD Rec.*, 33(1):50–57, 2004.

[134] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *CVPR*, 1:511–518, 2001.

[135] Terrance E. Boult Walter J. Scheirer. Cracking fuzzy vaults and biometric encryption. In *Biometrics Symposium*, 2007.

[136] Li Wan, Wee Keong Ng, Shuguo Han, and Vincent Lee. Privacy-preservation for gradient descent methods. In *KDD*, pages 775–783, San Jose, CA, August 2007.

[137] Yong Wang and Jiuqiang Han. Iris recognition using support vector machines. In *ISNN (1)*, pages 622–628, 2004.

[138] Hoeteck Wee. On obfuscating point functions. In *STOC*, pages 523–532. ACM, 2005.

[139] Peter Williams and Radu Sion. Usable pir. In *NDSS*, 2008.

[140] L. Wiskott, J.-M. Fellous, N. Krueuger, and C. von der Malsburg. *Face Recognition by Elastic Bunch Graph Matching*. Intelligent Biometric Techniques in Fingerprint and Face Recognition. CRC Press, 1999.

[141] Haiyun Xu, R.N.J. Veldhuis, T.A.M. Kevenaar, A.H.M Akkermans, and A.M. Bazen. Spectral minutiae: A fixed-length representation of a minutiae set. *CVPR Workshop*, pages 1–6, jun 2008.

[142] JuCheng Yang, jinWook Shin, BungJun Min, JongBin Park, and DongSun Park. Fingerprint matching using invariant moment fingercode and learning vector quantization neural network. *ICCIS*, 1:735–738, Nov 2006.

[143] Andrew Chi-Chih Yao. How to generate and exchange secrets. *Foundations of Computer Science*, pages 162–167, 1986.

[144] Xiaoyi Yu and Noboru Babaguchi. Privacy preserving: Hiding a face in a face. In *Proceedings of the 8th Asian Conference on Computer Vision*, pages 651–661, Tokyo, Japan, November 2007.

[145] Wei Zhang, S.S. Cheung, and Minghua Chen. Hiding privacy information in video surveillance system. volume 3, pages II–868–71, Sept. 2005.

Appendex

6.1 Negative number representation and homomorphic property

A two's-complement system is a system in which negative numbers are represented by the two's complement of the absolute value. An N-bit two's-complement numeral system can represent every integer in the range -2^{N-1} to $+2^{N-1}-1$.

Basically, we use the implicit sign representation of the numbers. If the range of numbers used is (0, M), then we use the numbers in the range (0, M/2) to represent positive numbers, and the remaining to represent negative numbers. The representation is chosen to ensure a single representation of zero, obviating the subtleties associated with negative zero. In our system, a negative number: -x is represented as x' = M-x. Efficiently handling the negative numbers is an implementation issue. We note that once the numbers are encoded using the implicit sign representation, we can carry out the regular arithmetic operations on it to get the correct result.

Fundamentally, the two's complement system represents negative integers by counting backward and wrapping around. For example: -95 modulo 256 is equivalent to 161 since: $-95+256=-95+255+1=160+1=161$

To understand the homomorphic property of the representations, consider the following examples. For the purpose of explanation, let us consider M to be 101, a simple encryption function to be exponentiation and the corresponding decryption function would be logarithmic. The example encryption function we have used is additive homomorphic.

In short, given a number x, we normalize it and then scale to maintain an acceptable decimal precision. We then compute the corresponding representation x' in our system. It is then encrypted to $w=e^{x'}$. This when decrypted gives us $z=(ln(w)+M)\%M$. x is recovered from z as follows, if $z>M/2$, then $z=z-M$, else z.

Let us now consider a few numerical examples to compute $y=x_1+x_2$ in encrypted domain. The examples are considered to cover all possible sign combinations of x_1 and x_2.

1) Let x_1 = 20 and $x_2=22$, we compute $x_1'=20$, $x_2'=22$. These are then encrypted using the encryption function to get: $y_1=e^{x_1'}=e^{20}$, and $y_2=e^{x_2'}=e^{22}$. The sum is computed in the encrypted

domain as: $y' = y_1 \cdot y_2 = e^{20} \cdot e^{22} = e^{42}$. The final sum is recovered post decryption to $y = ln(y') = 42$.

2) Let x_1 = 20, $x_2 = -12$, we compute $x'_1 = 20$, $x'_2 = 101 - 12 = 89$. These are then encrypted using the encryption function to get: $y_1 = e^{x'_1} = e^{20}$, and $y_2 = e^{x'_2} = e^{89}$. The sum is computed in the encrypted domain as: $y' = y_1 \cdot y_2 = e^{20} \cdot e^{89} = e^{109}$. The final sum is recovered post decryption to $y = ln(y') = 109 > 50$, therefore $y = 109 - 101 = 8$.

3) Let x_1 = 12, $x_2 = -25$, we compute $x'_1 = 12$, $x'_2 = 101 - 25 = 76$. These are then encrypted using the encryption function to get: $y_1 = e^{x'_1} = e^{12}$, and $y_2 = e^{x'_2} = e^{76}$. The sum is computed in the encrypted domain as: $y' = y_1 \cdot y_2 = e^{12} \cdot e^{76} = e^{88}$. The final sum is recovered post decryption to $y = ln(y') = 88 > 50$, therefore $y = 88 - 101 = -13$.

4) Let x_1 = -13, $x_2 = -23$, we compute $x'_1 = 101 - 13 = 88$, $x'_2 = 101 - 23 = 78$. These are then encrypted using the encryption function to get: $y_1 = e^{x'_1} = e^{88}$, and $y_2 = e^{x'_2} = e^{78}$. The sum is computed in the encrypted domain as: $y' = y_1 \cdot y_2 = e^{88} \cdot e^{76} = e^{166}$. The final sum is recovered post decryption to $y = ln(y') = 166\%101 = 65 > 50$, therefore $y = 65 - 101 = -36$.

5) Let x_1 = -28, $x_2 = 28$, we compute $x'_1 = 101 - 28 = 73$, $x'_2 = 28$. These are then encrypted using the encryption function to get $y_1 = e^{x'_1} = e^{73}$, and $y_2 = e^{x'_2} = e^{28}$. The sum is computed in the encrypted domain as: $y' = y_1 \cdot y_2 = e^{73} \cdot e^{28} = e^{101}$. The final sum is recovered post decryption to $y = ln(y') = 101 > 50$, therefore $y = 101 - 101 = 0$.

The number representation, is basically an implementation issue. The data can be imagined as points in an n-dimensional space. The classifier is a hyperplane in this space, while the confidence score is the distance of the data point from this plane. Now, one can employ any efficient translation, rotation and scaling as long as it is made sure that the distance comparision in the original space is equivalent to the corresponding distance comparision in the transformed space.

6.2 Residue Number System (RNS)

A *residue number system (RNS)* [124] represents a large integer using a set of smaller integers, so that computation may be performed more efficiently. It relies on the *chinese remainder theorem (CRT)* [124] of modular arithmetic for its operation.

A residue number system is defined by a set of k integer constants,

$$\{m_1, m_2, m_3, ..., m_k\}, \tag{6.1}$$

referred to as the moduli. Let M be the least common multiple of all the m_i.

Any arbitrary integer X smaller than M can be represented in the defined residue number system as a set of k smaller integers

$$\{x_1, x_2, x_3, ..., x_k\}, \tag{6.2}$$

with $x_i = X$ modulo m_i representing the residue class of X to that modulus.

6.2.1 Chinese Remainder Theorem (CRT)

The CRT is the method to reconstruct integers in a certain range from their residues modulo a set of pairwise relatively prime moduli. We denote

$$\mathbb{Z}_n \;=\; \{1,\; ...,\; n\} \tag{6.3}$$

Let M = $m_1 m_2$. Suppose $x \in \mathbb{Z}_M$. An equation of the form

$$ax \;\equiv\; b \, mod \, n \tag{6.4}$$

is called a *linear congruence*. Consider the numbers

$$a_1 \equiv x \; mod \; m_1 \tag{6.5}$$
$$a_2 \equiv x \; mod \; m_2$$

The CRT considers the question of recombining $a_1,\; a_2$ back to get x. CRT tells us, when the system will have a solution, and if does have a solution, it provides an algorithm for finding one. We will want to solve this equation for x.

6.2.1.0.1 Theorem 1 The linear congruence $ax \;\equiv\; b \, mod \, n$ has a solution if and only if $d \mid b$, where $d \;= gcd(a, n)$. If d does not divide b, then there are d mutually incongruent solutions modulo n.

If $gcd(a, n) = 1$, then the linear congruence $ax \;\equiv\; b \, mod \, n$ has a unique solution modulo n.

6.2.1.0.2 Theorem 2 Let $m_1, m_2, ..., m_k$ be pairwise relatively prime integers. That is, $gcd(m_i, m_j) = 1$ for $1 \leq i < j \leq k$. Let $a_i \in \mathbb{Z}_{m_i}$ for $1 \leq i \leq k$ and set $M = m_1 m_2 m_3 ... m_k$. Then there exists a unique $y \in \mathbb{Z}_M$ such that $y \equiv a_i \ mod \ m_i$, for $i = 1, ..., k$. Furthermore there is an $O(k^2)$ time algorithm to compute y given a_1, a_2, m_1, m_2, where $k = max(|m_1|, |m_2|)$.

6.2.1.0.3 Proof For each i, let

$$n_i = (M/m_i) \in \mathbb{Z} \tag{6.6}$$

By hypothesis, $gcd(m_i, n_i) = 1$ and hence $\exists \ b_i \ in \ \mathbb{Z}_{m_i}$ such that

$$n_i b_i \equiv 1 \, mod \, m_i \tag{6.7}$$

Let $c_i = b_i n_i$. Then

$$c_i \equiv 1 \, mod \, m_i \equiv 0 \, mod \, m_j; \ \ for: \ j \neq i \tag{6.8}$$

Set

$$y \equiv \sum_i c_i a_i \ mod \ M \tag{6.9}$$

Then for each i

$$y \equiv a_i \, mod \, m_i \tag{6.10}$$

Further, if $y' \equiv a_i \ mod \ m_i$ for each i then $y' \equiv y \ mod \ m_i$ for each i and since m_is are pairwise relatively prime, it follows that $y \equiv y' \ mod \ M$, proving uniqueness.

6.2.1.0.4 Algorithm Let,

$$M = \prod_{i=1}^{k} m_i \tag{6.11}$$

where m_i are pairwise relatively prime. We can represent any integer in $\mathbb{Z}_M$ by a $k - tuple$ whose elements are in $\mathbb{Z}_{m_i}$ using the following correspondence:

$$A \leftrightarrow (a_1, \ a_2, \ ..., \ a_k) \tag{6.12}$$

where $A \in \mathbb{Z}_M$, $a_i \in \mathbb{Z}_{m_i}$, and $a_i = A \, mod \, m_i$, for $1 \leq i \leq k$.

For every integer A such that $0 \leq A < M$ there is a unique k-tuple $(a_1, a_2, ..., a_k)$ with $0 \leq a_i < m_i$ that represents it, and for every such k-tuple $(a_1, a_2, ..., a_k)$ there is a unique A in $\mathbb{Z}_M$. Computing A from $(a_1, a_2, ..., a_k)$ can be done as follows:

Let $M_i = M/m_i$ for $1 \leq i \leq k$. Note that $M_i = m_1 \times m_2 \times ... \times m_{i-1} \times m_{i+1} \times ... \times m_k$ so that $M_i \equiv 0 \, mod \, m_j \ \forall \, j \neq i$.

Then $for:\ 1\ \leq\ i\ \leq\ k$ let

$$c_i\ =\ M_i \times (M_i^{-1}\ mod\ m_i) \tag{6.13}$$

By the definition of M_i, it is relatively prime to m_i and therefore has a unique multiplicative inverse mod m_i. Thus the above equation is well defined and produces a unique value c_i. We can now compute

$$A\ \equiv\ (\sum_{i=1}^{k} a_i c_i)\ mod\ M \tag{6.14}$$

6.2.1.0.5 Example [123] To represent 973 mod 1813 as a pair of numbers mod 37 and 49, define $m_1\ =\ 37, m_2\ =\ 49, M\ =\ 1813$, and $A\ =\ 973$. We also have $M_1\ =\ 49$ and $M_2\ =\ 37$. Using the extended Euclid's algorithm, we compute $M_1^{-1}\ =\ 34\ mod\ m_1$ and $M_2^{-1}\ =\ 4\ mod\ m_2$. (Note that we only need to compute each M_i and each M_i^{-1} once for all.) Taking residues modulo 37 and 49, our representation of 973 is (11,42), because 973 mod 37 = 11 and 973 mod 49 = 42.

Now suppose we want to add 678 to 973. What do we do to (11,42)? First we compute (678) $\leftrightarrow$ (678 mod 37, 678 mod 49) = (12, 41). Then we add the tuples element-wise and reduce (11+12 mod 37, 42+41 mod 49) = (23, 34). To verify that this has the correct effect, we compute:

$$\begin{aligned}(23, 34) \leftrightarrow a_1 M_1 M_1^{-1}\ +\ a_2 M_2 M_2^{-1}\ mod\ M \\ = [(23)(49)(34)\ +\ (34)(37)(4)]\ mod\ 1813 \\ = 43350\ mod\ 1813 \\ = 1651\end{aligned} \tag{6.15}$$

and check that it is equal to (973+678) mod 1813 = 1651.

Suppose we want to multiply 1651 (mod 1813) by 73. we multiply (23, 24) by 73 and reduce to get (23x73 mod 37, 34x73 mod 49) = (14, 32). It is easily verified that

$$\begin{aligned}(14, 32) \leftrightarrow a_1 M_1 M_1^{-1}\ +\ a_2 M_2 M_2^{-1}\ mod\ M \\ = [(14)(49)(34)\ +\ (32)(37)(4)]\ mod\ 1813 \\ = 28060\ mod\ 1813 \\ = 865 \\ = 1651\ \times\ 73\ mod\ 1813\end{aligned} \tag{6.16}$$