EFFICIENT & SECURED FRAMEWORK FOR INTERNET OF THINGS BASED ON MOTIFS

A Thesis

by

AKASH SAHOO

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Rabi Mahapatra
Committee Members,	Jeff Huang
	Jiang Hu
Head of Department,	Dilma Da Silva

December 2016

Major Subject: Computer Engineering

ABSTRACT

Internet of Things (IoT) has allowed embedded devices to connect to the vast internet network worldwide. The amount of data produced and exchanged between them is growing exponentially and with the present hardware and software architecture it is difficult to support them. With billions of IoT devices waiting to be connected in the near future, it is necessary to build infrastructure for the upcoming change as the energy and cost associated with the continuous transmission, classification and storage will be huge. We need to build an efficient framework that can scale easily, follow consistent protocol, maintain security and save resources.

The thesis focuses in solving the major upcoming problems of the Internet of Things by proposing a lightweight framework which resides in both the server and the end device as server client model. The framework has the following benefits - it reduces network congestion, reduces data consumption and maintains security. The framework resides on the data and communication layer, classifying the data into known patterns - Motifs. We have used modified Hidden Markov Model to classify the sensor data into Motifs. The framework transfers only the motifs attributes information instead of complete sensor data. Thus the data can now be compressed by orders of magnitude into these classes of recurrent patterns. It not only saves on data storage but also on network transmission. It helps us to create a state based model and in anomaly detection and security.

We also optimize Partial Homomorphic Encryption based on El-Gamal Algorithm using OpenCL, OpenMP, SIMD, batch processing, Karatsuba algorithm and used to secure the framework while allowing simple computation to be performed on the encrypted data.

ACKNOWLEDGEMENTS

I would first like to thank my thesis advisor Dr. Rabi Mahapatra, Professor at Texas A&M University without whom the thesis would not have been possible. He has given me a lot of freedom in choosing the research topic and has been a constant source of motivation. He has also helped in procuring the hardware without which the thesis would not have been possible.

I would also like to thank the members of my committee panel, Dr Jeff Huang and Dr Jiang Hu for taking out valuable time and being part of this work. I would also like to thank Department Program Coordinator Karrie Bourquin for her help in the paper work.

I would like to thank my lab mates for constantly motivating and giving advice which was essential for the thesis. They have been patient and encouraging to listen to my research work and given valuable advice. Last but not the least, I would like to thank my friends Amit Panda, Meghna Baijal and Veenu Trehan for their help and motivation without which the thesis would not have been possible.

Finally I would like to share my achievement with my parents who provided me support, motivation and encouragement during my masters degree, research and writing this thesis.

Thank you,

Akash Sahoo

NOMENCLATURE

IoT	Internet of Things
HMM	Hidden Markov Models
DTW	Dynamic Time Warping
GPU	Graphics Processing Units
CPU	Central Processing Units
SIMD	Single instruction, multiple data
SAX	Symbolic Aggregate ApproXimation
SVM	Support Vector Machines

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

Internet of Things (IoT) term was first coined in 1999 by Britisher Kevin Ashton. He describe IoT as an ecosystem to connect physical world objects to the internet by sensors. Today internet of things (IoT) is described as the inter networking of physical devices, vehicles, building, embedded devices, wearables etc. (Fig 1.1) along with software, sensors, network connectivity that enable these devices to exchange information to perform tasks. It is known as Cyber-Physical system when it is embedded with physical process.

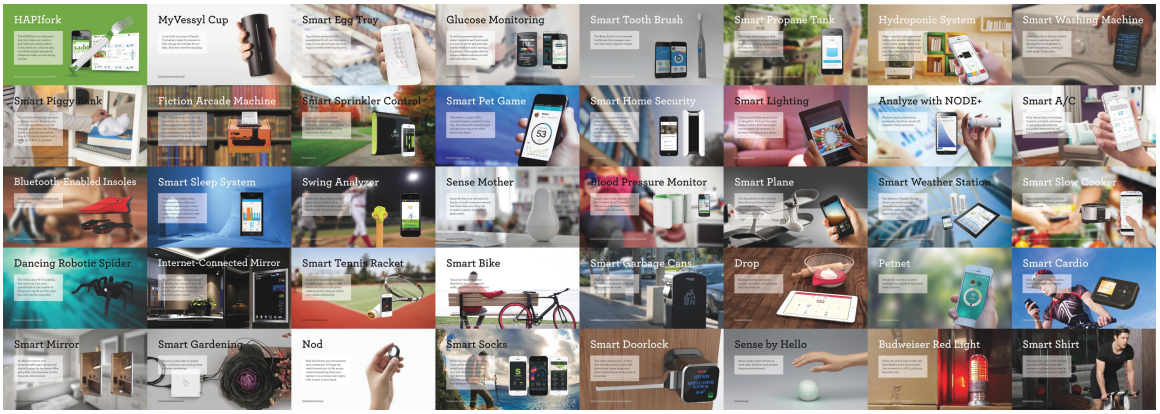

Figure 1.1: Internet of Things or Everything

Many devices such as internet connected toaster, refrigerator have been around since 2000's but until now an ecosystem is being created for such devices such that they can follow unified protocol. It will allows many small devices to be connected to internet more easily and cheaply. Certain technologies like Ubiquitous computing (low cost, high speed, pervasive connectivity connecting all devices), IPv6, low power embedded IP devices, miniaturization of devices, rise of data analytics, cloud

computing, security protocols etc. have come up to make this possible. Also many areas like smart city(energy, smart grid, water, waste management), smart homes (home controller, lights, security, cameras), wearables(giving calories, workout etc), health care (devices inside our body sending vitals), offices(smart lights, energy, security), factories(monitoring workers, time tables etc), vehicles(sensors connected) etc have created a demand for IoT to be a part of our lives. The adoption of IPv6 in 2011 which can allow 340 undecillion IP address [10$\hat{3}$6] has made it possible for IoT to grow. Lastly IPSO Alliance launched IP for Smart Object Communication in 2008, and 6LoWPAN in 2010 which had great impact on energy and communication in IoT. In spite of all the developments, vendors have followed their own protocols and have not focused on scalability.

IoT application have made to way through every part of our lives from our body to our surrounding. As the number of devices grows, the amount of traffic and data they would generate has been growing significantly as well. Cisco, a major vendor in the field has predicted data growing 40% in 2014 and to 70% in 2019 [1]. Similar results are there for M2M (machine to machine) connections. Such huge volumes of data will need better infrastructure, storage, security, protocol to work.

1.1 Problems

- Huge types of devices and applications have made the ecosystem complex. Vendors have started different protocols
- There are issues with security and privacy of data since the since edge nodes are less powerful to perform encryption and have vulnerable ports
- Such huge amount of sensors generate lot of data in order of Petabytes. The compression techniques don't work well due to randomness and non-contextual

data.

- IoTs require lot of components and software managements to work. Also most of the IoT platform have heterogeneous devices different protocols that makes it difficult for a platform to support all devices.
- Lot of upfront costs like maintaining enterprise cloud, storing data, servers, big data technologies, data analytics.
- Scalability issues with large number of devices
- Real time processing and actions
- Network bandwidth
- Power – Often devices are always on and consume lot of power
- Security and privacy is a big concern and these devices are closely connected to our world e.g. dropcam, printers to private network, etc.

1.1.1 Data Challenge

Billions of IoT devices continuously generate enormous raw data that needs to be processed. eg. 1 Hour of ECG data is 1 GB, typical web logs are 5 GBs per week, Space shuttle databases are 150 GB. So these huge amounts of data needs to be processed and analyzed and it creates a huge and computation and resource problem. We try to address IoT data produced and communicated as major problem in any IoT system. The major problem lies in the lower layers of because of naive implementation of frameworks. The sensors produce lot of redundant data which are sent over network and processed at servers. There is no real time data and context analysis and processing is done at server which creates huge computation costs for

servers with large IoT devices and data. So we try to solve this challenge by preventing the compute and resource costs of redundant data by classifying them into motifs before transmitting.

1.2 Protocols

These are the major different communication protocols followed by industry. The communication mechanism can be Wifi, Bluetooth, zigbee, radio wireless, Lifi, TCP/UDP, cellular internet etc.

- **MQTT** - MQ Telemetry Transport - It is always-on connection and thus will limit the time the devices can be put to sleep. It is bare bone in implementation and it also lacks encryption layer.
- **CoAP** - Constrained Application Protocol - It is used for resource-constrained devices. It uses UDP instead of TCP and so it is fast but lacks quality control and acknowledgment. It is one-to-one protocol, so broadcast capabilities are lacking.
- **XMPP** - Extensible Messaging and Presence Protocol - It implements TCP Protocol and is based on XML. Is similar to public subscribe system and can distribute data to IoT devices at once, but it lacks end to end encryption and QoS.
- **AMQP** - Advanced Message Queuing Protocol
- **LwM2M** - Lightweight M2M
- **DDS**- a fast bus for integrating intelligent machines (D2D)

1.3 Generic IoT Architecture

IoT generally consists of sensors, actuators as physical layer, communication layer consisting of protocols and server to do analytics and provide services.

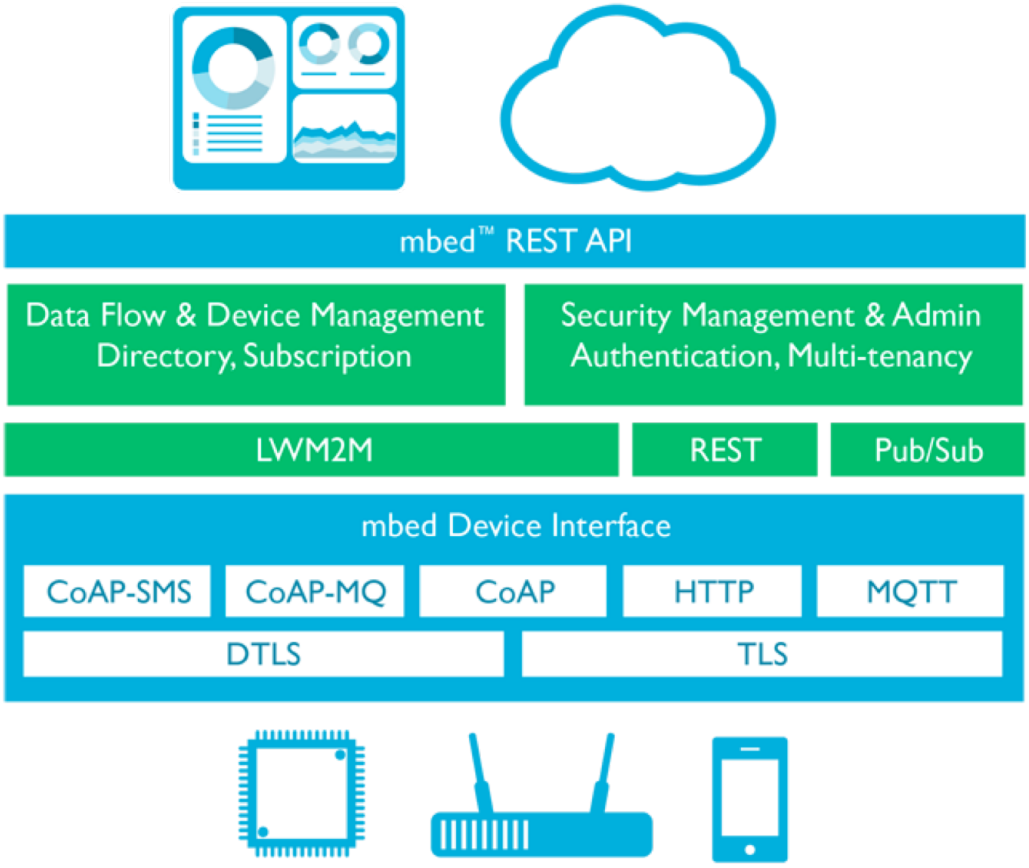

Figure 1.2: Generic IoT Architecture

The following are the major layers in any kind of IoT system.(Fig 1.2) The layers consists of majorly:

- **Sensor layer** – It is the lowest abstracted layer consisting of the actual hardware

sensors, device drivers, firmware etc. It measures the physical quantities and interconnects physical environment with digital world. It also is responsible to maintain hardware and On-Chip Security.

- **Communication Layer**- This layer consists of major communication methods like Wifi, Bluetooth, radio, cellular etc. It also consists of the major communication protocols like CoAP, MQQT, XMPP etc discussed before.

- **Gateway and network layer** – It consists of the high performance network infrastructure to support communication. It is responsible for latency, bandwidth and security. It allow multiple organization to use it and incorporates protocols.

- **Management Service Layer** – This layer extracts relevant data and information from the raw data and does streaming analytics, real time processing. It also has APIs for query processing and responsible for infrastructure and services.

- **Application Layer** – It is the high level User Interface for IoT. Analytics, Inference, statistics, status are shown.

The devices are of three classes depending on processing power and energy requirements - Medium Level(arduino), Mid Level(ARM- Arduino Yun), most Capable level (BB, Raspberry Pi). We also have various Operating systems and RTOS for IoT devices like IBM Bluemix, Contiki, node red, Angstrom etc.

1.4 Related Work

There has been lot of work in the field of time series and motifs. Most of the time series work has been used in financial markets and by statisticians. The Motif based

framework would not have been possible without the fundamental contributions by work on data mining by Keogh [2] especially the work done on Symbolic Aggregate approXimation (SAX) Algorithms which finds motifs in time series. The book on Pattern Recognition by Bishop [3] helped in finding out patterns from time series. Indyk [4] has proposed using representative trends in time series to classify sketches. This method uses random projection, FFT and PCA methods to compute sum of distances faster and efficiently. There has been work using wavelets [5] which have shown promising results. Data mining using multiple streams in real time can help IoT and similar work done by Zhu [6] has great potential.

Dynamic Time warping (DTW) has been one the most popular methods used by researchers for streaming data and finding similarity [7]. It allows dynamic scaling which has been used in speech processing, signal processing but the computation time of O(n$\hat{2}$) has been a limitation.

Finding time series motifs from accelerometer is similar to gesture recognition using accelerometer[8] [9]. Gesture recognition can be done by Microsoft Kinect or camera but has huge computation complexity on embedded devices and can not work in dark environment. Gesture recognition also performs badly in the absence of standard gesture set and therefore it needs training beforehand. Also machine techniques like support vector machines, decision trees[15] have been used for gesture recognition. Also these methods need lot of training and perform badly on random streaming data and have computation complexity. Kolmogorov complexity of an object is defined as the length of the shortest computer program that converts the object as output. It is used for compression, complexity analysis and measuring randomness in an entity. It is an clustering technique based on compression. Work by Cilibrasi [10] is based on this notion. Extensive research has already been done in the fields of sensor data compression - Time series compression [11] and model based

compression [12]. There have been models that approximate the sensor data which helps in compression [13].

There has been a lot of work in data mining that has been applied to the field of IoT. Techniques such as SAX (Symbolic Aggregate approXimation) [14] which is a symbolic representation of time series and allows dimensionality reduction performs badly on random data and can work for limited number of symbols.

There have been number of techniques [15] to find patterns among two time series such as DTW (Dynamic Time Warping), HMMs (Hidden Markov Model) etc., but they perform badly if the data is noisy or if the data has lot a of temporal and spatial variation. There are many problems associated with classification into motifs and it does not have a straightforward solution. The exact start and end position of the motif is difficult to detect as there could be a gap between the start of actual signal. Another big problem is the frequency, the data could come faster or slower; so the motifs could be time dilated or constricted. Also the morphology of the pattern creates problems for classification. This thesis does addresses the above problems and gives plausible solutions which are discussed in the next section.

Now coming to security, Homomorphic encryption is the new encryption scheme that has huge potential of providing end to end security while allowing computation on the encrypted data. Much work has been done [16] but the major limitation is the computation time which is three order greater than the counterpart AES.

1.5 Summary of the Thesis Work

The thesis work has been a step towards the developing the framework for the booming field of Internet of Things (IoT). Current IoT systems require huge storage space and network bandwidth because of large number of devices communicating continuously.

The development is carried out on android devices as end IoT devices due to the fact that android mobile devices have a lot of sensors and libraries to emulate edge IoT devices. We take accelerometer sensors which are the most complex sensors because of high frequency of data to test them in our framework. We also provide place for slower data rate sensors like temperature sensors and pressure sensors in our framework.

The thesis consists of the following parts -

- Initially we explain the problems and limitations with the current state of art IoT system. We describe the motivation behind the research.

- We propose a Motif based framework for IoT that will help reduce the communication, storage costs, and describe its architecture.

- We describe the motifs recognition algorithms that form the essential core of the proposed framework. We also discuss in detail the other gesture recognition algorithms and their limitations.

- We test the framework against accelerometer data for various patterns and record their accuracy. We extend it for various applications like hand gestures, human activities and exercises. We also test it against slow data rate sensors like temperature sensors.

- We propose an upcoming encryption technology Homomorphic encryption using ElGamal Algorithm that will provide end to end security and enable computation on encrypted data.

- We accelerate the Homomorphic Encryption algorithm using software optimization, hardware accelerators and GPUs.

The final discussion summarizes the main results, performance reports, advantages and future work to be done on analytics.

2. IOT MOTIF FRAMEWORK ARCHITECTURE

The previous chapter threw light on the major limitations and challenges faced with IoT. We discuss the architecture and working of the motif based framework for IoT in this chapter. The motivation to build the framework was due the fact that it is challenge for platform to scale up large number of IoT devices. The challenge has motivated to work on the framework layer of IoT to : Find ways to reduce data transmission, optimized storage and built end to end encryption to protect user's sensitive information.

We had previously discussed that one of the biggest problems in the field of IoT is the vast amount of data generated. So the main idea behind the framework is to convert the data into motifs or recurrent patterns and transmit only the meta information of the motifs. Time series motifs are recurrent patterns of interest in a long time series data, carrying precise information of the events with time-stamps. Current IoT framework are naive and follow client server model to transmit data. People are generally concerned with the events rather than raw data e.g. people care about the steps in fitbit rather than sensor data. So instead of communicating sensor data, we propose transmission of events (steps in case of fitbit, heart beat classes in case on ECG sensors) (Fig 2.1).

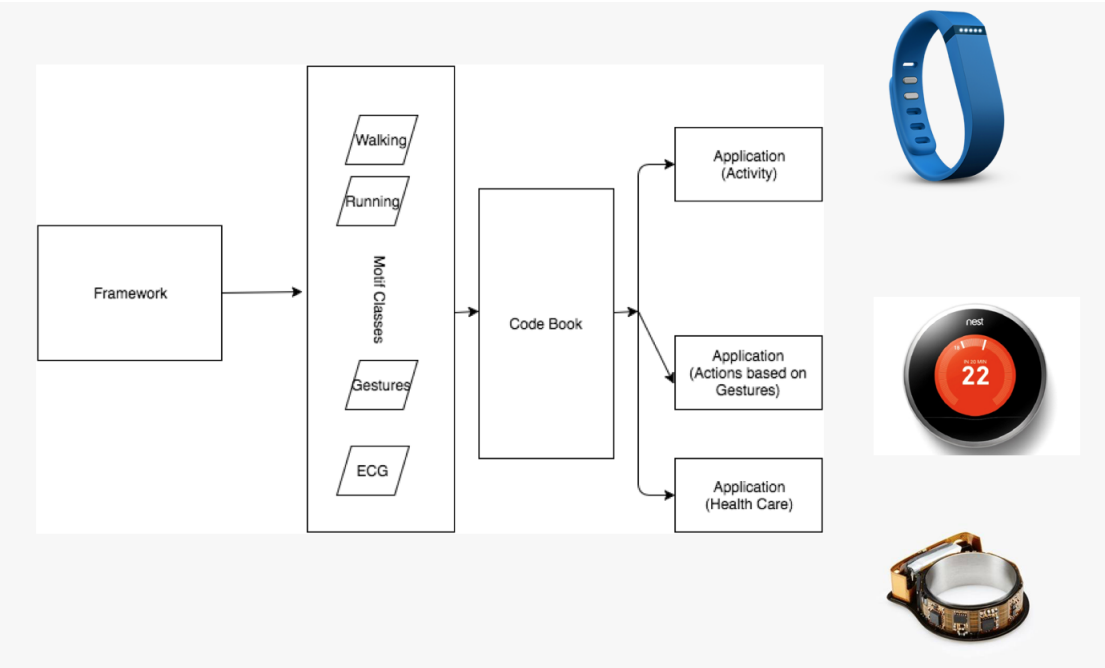

Figure 2.1: Motifs Classes Change with Application

2.1 Framework Layers

The Motif based framework for IoT's architecture consists of different layers:

- **Sensor layer and Listener** - responsible for data acquisition. It consists of firmware, device driver, interrupts, I/O, buses etc low level details to acquire data.

- **Filtering, processing layer and feature extraction** - Filtering the noisy data, extract features

- **Motifs and gesture recognition layer** - It consists of the previously trained motifs and algorithms to detect, recognize motifs in the time series data.

- **Communication layer with encryption** - wireless communication layer with encryption block
- **Server, Cloud and visualization** - Store the time series motifs events, anomalies in structured form and show visualization when needed.

The motifs and gesture recognition block forms the most important layer in our framework (Fig 2.2). It is the most computation intensive layer. We will now discuss all the above layers, their working and importance in the forthcoming chapters. The figure below shows the basic blocks of the proposed motif framework for IoT.

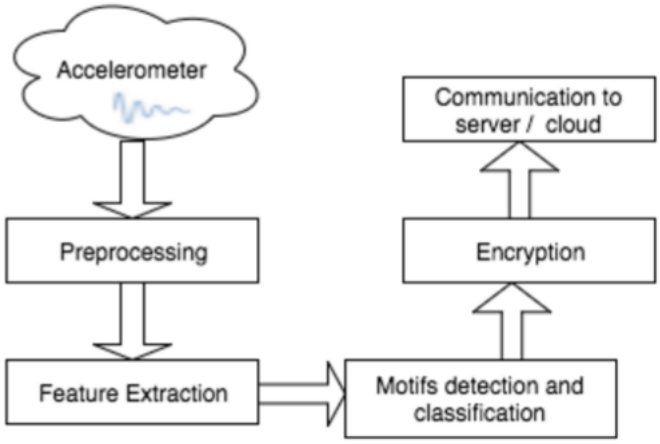

Figure 2.2: Architecture of the Framework

2.2 Sensor Layer

We have taken android devices as our IoT edge nodes due to code portability, available sensors APIs, android libraries etc and the work can be extended to any embedded device(s) with sensors and RTOS.

Current generation of android phones have sensor stack with sensor device drivers, firmware, framework, APIs so that user can interact with the sensor data from the phone (Fig 2.3). The Hardware Abstraction Layer(HAL) represents the interface between the Android framework and the hardware specific software and can help in setting the rate to receive data from the sensors, set delays, sampling, activate or deactivate sensors, flush, save power etc (Fig 2.4).

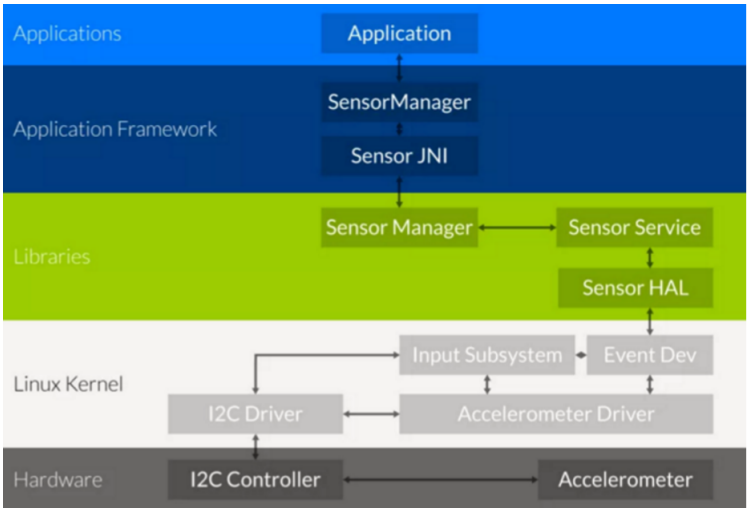

Figure 2.3: Sensor Architecture

The Sensor API requests for the sensor values and orientation of the mobile with respect to screen and gives back the raw data for further processing.

2.2.1 Accelerometer Sensors

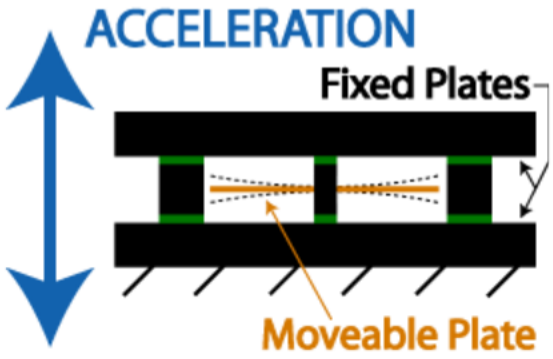

Figure 2.4: Accelerometer Sensor

An accelerometer sensor detects the acceleration of the device along the 3 sensor axes (Fig 2.5). The accelerometer has three capacitors in each direction which change values with mobile acceleration. Whenever the sensor is accelerated an opposite force acts due to acceleration and the distance between the dielectric plates of the capacitor changes. Thus the capacitance value too changes and the value is processed to give acceleration value. The values are then processed and normalized by the sensor framework and android gives values in (m/s$\hat{2}$)

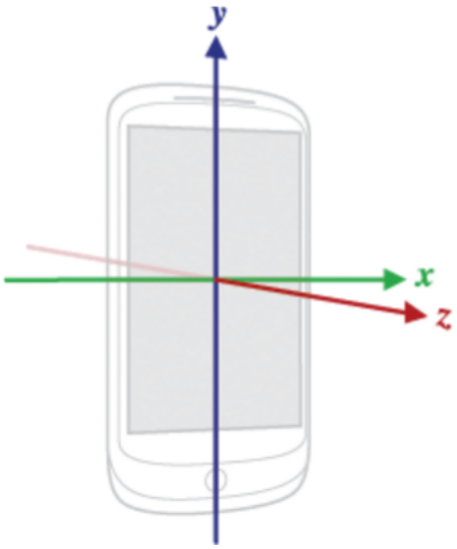

Figure 2.5: Orientation of the Mobile

The accelerometer shall follow the physics laws of giving norm of (x,y,z) zero in free fall and 9.8 m/s$\hat{2}$ when the devices lies flat on the table. Practically the signal received is quite noisy and if we integrate its values to velocity or displacement, they are way off from the real values. So the accelerometer values are directly processed.

2.2.2 Listener

The listener as the name suggests listens to accelerometer or sensor data and sends it to the preprocessing and motifs recognition module. It can help to control frequency, prevent transmission of redundant data, aligning time series data with time stamp etc. In case of temperature sensor, listener becomes an important part of framework. It stores the state information and only transmits information only when the state or temperature changes.

2.3 Preprocessing

We saw that accelerometer signals are complex, have noise distortion (Fig 2.6) and vary with same gestures. In our case of accelerometer gestures, it not ideally not possible that the user have same pattern for same gestures. Also there might be a random tilt while doing the experiment. So we have to correct the noise and tilt before motifs recognition.

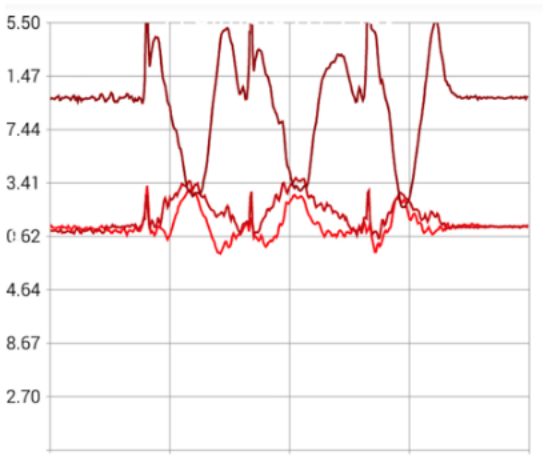

Figure 2.6: Raw Sensor Data from the Accelerometer

2.3.1 Tilt Compensation

To compensate the initial tilt of the phone we need to first find the orientation of the phone and actual gravitational pull. We therefore first estimate the axis that is more affected by gravity and transform the coordinate system to it, making it the Y-axis. We then correct the raw accelerometer values. This is important as different orientation of phone can make pattern look different.

2.3.2 Noise Removal and Signal Smoothing

Accelerometer signals have a lot of white noise and random spikes which will make it difficult to get accurate results. So we need to remove these components. Also linear trending and detrending need to be done to make the signal close to the baseline axes. To remove the accelerometer noise we have a Butter-worth low pass filter. The low pass filter eliminates noise to the data which are not significant to the gesture recognition process. It removes all the taps, high sudden peaks, jitter, spikes etc. Also to remove the white noise we have a high pass filter too. We have a sliding moving average window filter to smoothen out the signals if necessary and also we have envelope detection filter to detect the boundary shape and morphology of the signal. After the preprocessing, the data is quantized and discretized into levels for the training the models and motifs detection.

2.4 Motifs

Time series are collection of observations of physical quantities or events made with respect to time. They are generally measured at regular intervals and successive points. These are made for some phenomenon or application (Fig 2.7). In case of IoT the time series consists of the raw sensor data and events of the environment for some application or physical quantity. These are generally best represented in line charts and scatter plots with time on x-axes and values on y-axes. Some examples are ECG Data, Stock data, signals processing, weather etc.

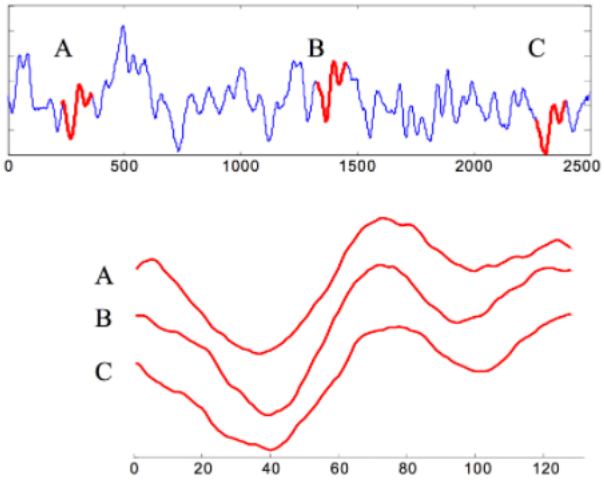

Figure 2.7: Time Series Motifs[7]

We have to do clustering, classification, motifs detection, labeling etc to make sense of such large data. Typical time series dataset sizes are huge e.g. 1 Hour of ECG data is 1 GB, typical weblogs are 5 GBs per week, Space shuttle databases are 158 GBs and growing. So one needs to find out the important necessary information

required from these datasets to make sense.

Detection of previously unknown, frequently occurring patterns are called Motifs. Existing methods to detect motifs include wavelets, Markov model templates, k-means clustering, similarity search, dynamic time warping, HMMs, decision trees, Support vector machines etc. Finding motifs would be an useful tool for summarizing and visualizing massive time-series database.

2.5 Existing Methods for Motifs Classification

There have been popular existing methods described below for motifs detection and classification.

2.5.1 Dynamic Time Warping

One of the oldest, simple and most common techniques of matching time series is Dynamic Time warping (Fig 2.8). It has been used to match handwriting, gesture, sign language recognition, time series matching, signal similarity etc. It has order of complexity as $O(n\hat{2})$. In DTW we store templates of stream of time series data and label them as different classes. We then do brute force matching of time series data to these classes and find the matches. It is based on calculation of minimum distance between the input stream you want to match with all the classes in the codebook. The elegance of the technique lies in the euclidean distance calculation which does not necessarily do linear mapping and it does dynamic neighboring distance mapping.

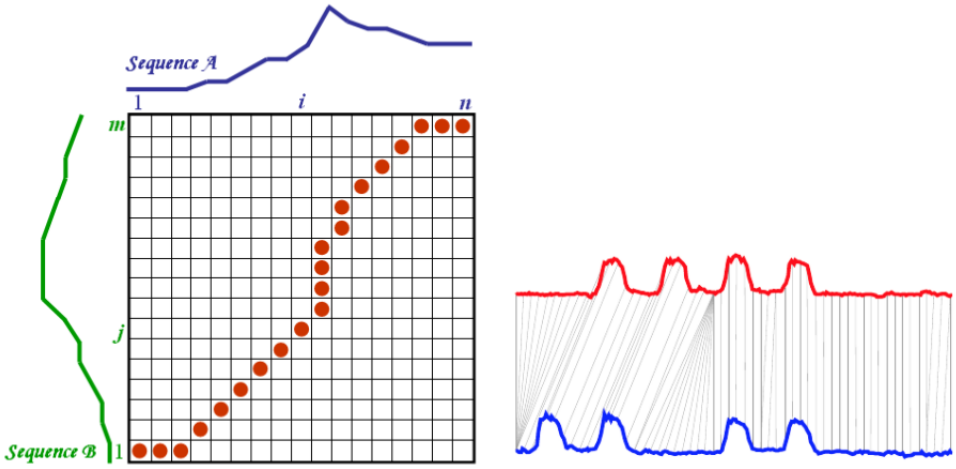

Figure 2.8: DTW Matching of Signal with Template

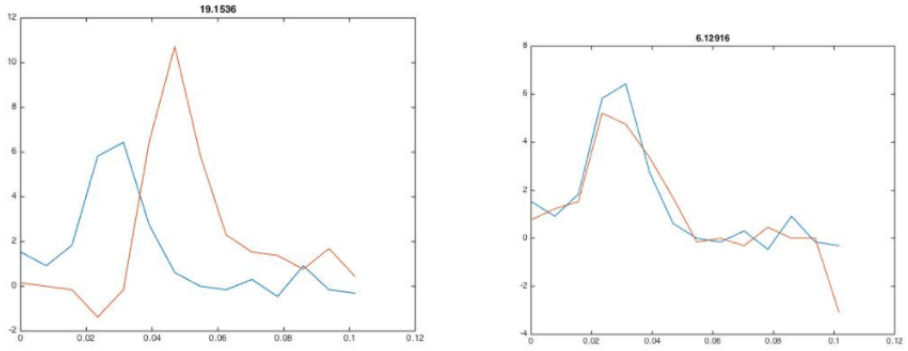

Figure 2.9: Similarity between the Time Series Data with Template by Euclidean distance

Although the technique is simple, it has disadvantages and weaknesses. The complexity of the algorithm is O(n$\hat{2}$* no of classes) and therefore is slow (Fig 2.10). It is naive in just matching and does not do any inferences or description of the data

(Fig 2.9). Its accuracy vastly decreases if there is noise in the signals. Due to lack of inferential parameters it is difficult to scale up, transfer learning or implement in distributed systems. It is also difficult to set the end boundaries, monotonicity and step size conditions.

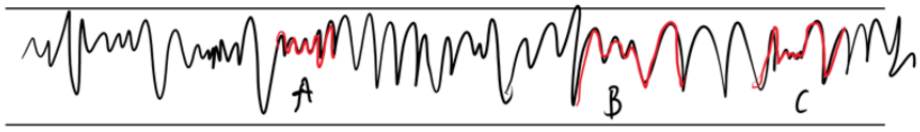

Figure 2.10: DTW making 6Million request to Euclidean Function to find 3 Motifs

2.5.2 Symbolic Aggregate ApproXimation (SAX)

E.Keogh group at Riverside [2] has used an algorithm called the Enumeration of Motifs through Matrix Approximation (EMMA) which has detected motifs faster (Fig 2.11). It has shows faster results. SAX Algorithm encodes a time-series X of length n into the string of arbitrary length. The algorithm consist of two steps: (i) it transforms the original time-series into the PAA (Piece wise Aggregate Approximation) representation and (ii) it converts the PAA data into a string.

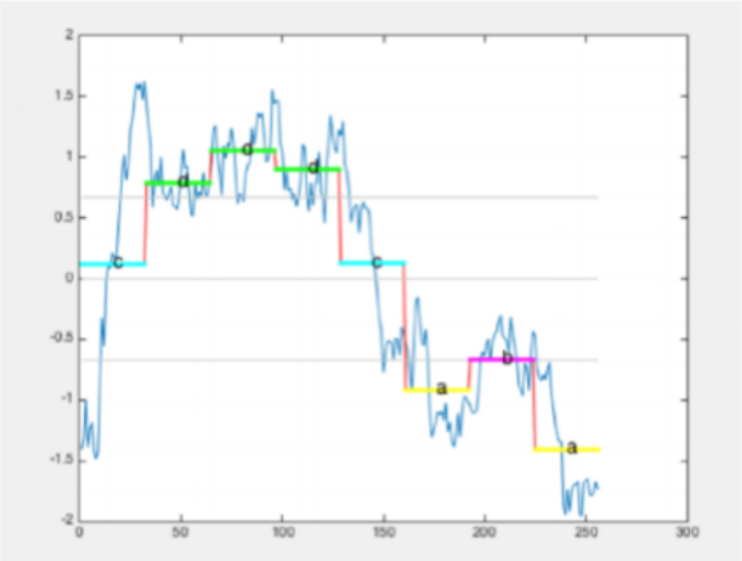

Figure 2.11: Signal Encoded into "cdddcaba" by SAX Algorithm

The signal is divided into multiple windows of fixed length and the windows are matched with each other. The similar windows are named same characters. The signal is represented by the aggregation of the represented characters instead of the raw signal. If another signal with different morphology has the same encoding then it is mis-classified. Also this method gives poor accuracy in real life scenarios.

2.5.3 Hidden Markov Models

Hidden Markov Models (HMMs) (Fig 2.12) is one of the most popular techniques for gesture recognition and found applications in speech, handwriting, sketch etc recognition. It is a variant of state machine and Markov chains. These are however not deterministic. Markov chains are probabilistic based state pattern based machines in which the next state depends on the previous state and are deterministic whereas in HMMs the observation is a probabilistic function of the state (Fig 2.13). For eg if a person is asked to tell whether it is raining outside or not on the notion

that people inside the room have wet umbrella or not is solving a HMM problem.

It is widely used as due to the fact that it is simple. HMMs allows you to estimate the parameter and do inference while abstracting the hidden layers. Also it is powerful enough to handle real world applications and scaling well. There are some hidden random variables Z1 ... Zn E {1, ... m} and there are some other random variable X1 .... Xn E to some set cap * (real/ discrete/ finite)

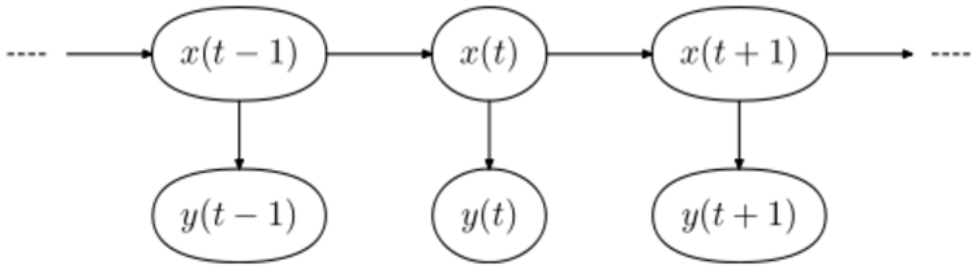

Figure 2.12: Graphical Model of HMMs

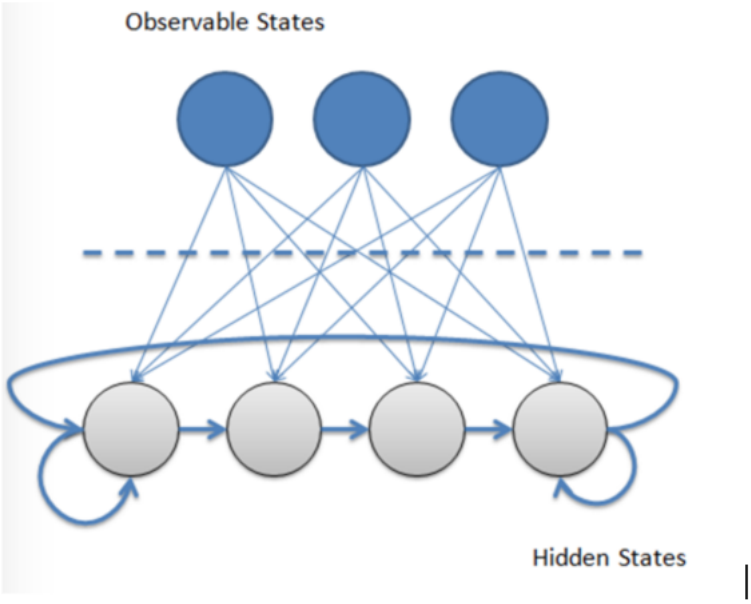

Figure 2.13: Hidden States

The major advantages of HMMs are that they can be extended to any temporal classification application like speech, gestures etc. They handle scalability well and produce good accuracy and they can learn incrementally. The disadvantage of HMMs are that they make large Markovian assumptions about the data and that the transition probabilities depend on the current state. It means that we approximate the order of hidden states possibilities for simplification. Also the distribution assumption is Gaussian one. Also it requires lot of parameters to be set, so we need large data to train the model.

People have also used clustering and decision tree based approaches to classify time series based data but as they are random, clustering techniques don't work well. Also, providing parameters for decision tree is difficult as these change a lot.

2.5.4 k-Means Clustering

Clustering are methods to group a collection of objects into subsets or clusters such that the elements within a cluster a closely related to each other than member from other clusters (Fig 2.14). K-Means clustering is simple clustering algorithm for unsupervised classification by creating k clusters. It can cluster around any point but it may lead to different clusters if variance in data set is not large. If variables are huge, then K-Means most of the times computationally faster than hierarchical clustering, if we keep k smalls. It is also difficult to predict k value. It also does not work with global clusters. Moreover, different initial partitions can result different clusters. It uses euclidean distance between the objects to minimize the sum of intra-cluster variances.

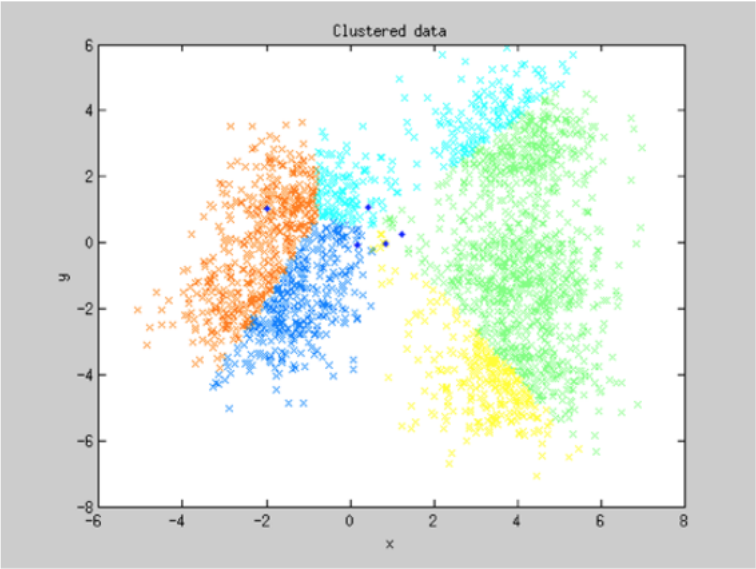

Figure 2.14: k-Means Clusters

2.6 Motif Addition and Classification Algorithm in Framework

We see that the recognition of the gestures deteriorates as the morphology, amplitude, duration changes. So we take the best features of each algorithm as the base algorithm of the framework. The SAX algorithm divides the time series data into windows. We then pass the signal through pyramids which changes the amplitude and duration of the signal so that we can do accurate recognition (Fig 2.15). We do an envelope detection and find out the morphology of the signal. We then use HMMs to train the morphology and classify the signals. This approach not only reduces errors due to amplitude and duration but by averaging the envelope, we also get better matches with the classes.

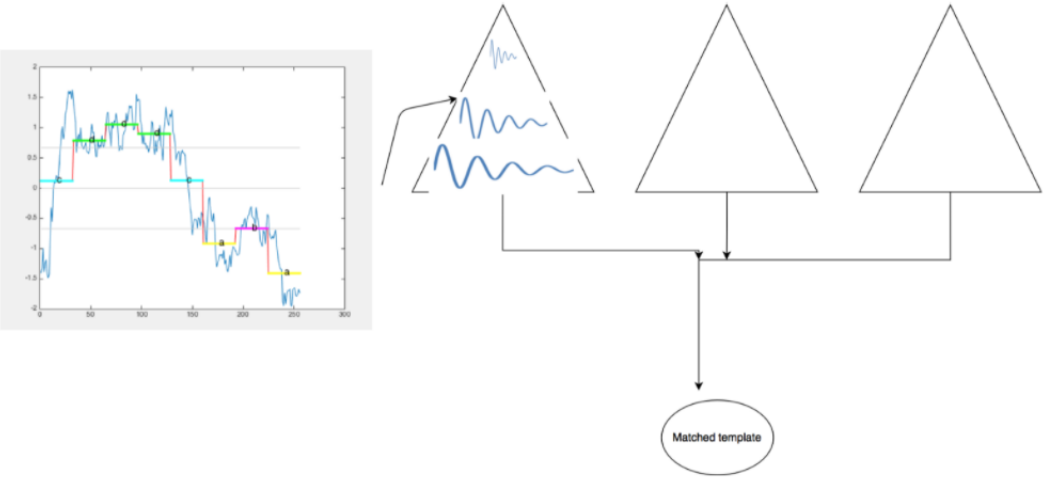

Figure 2.15: Dynamic Scaling of Motifs for better Training

The first step after preprocessing the raw signals is training of Motif classes. Motif classes vary from application to application. For eg in case of fitbit, walking

steps is the motif class and in case of myo gesture band, hand gestures are the motif class. The proposed IoT framework allows new motifs to be added or modified easily.

2.6.1 Hidden Markov Models Training

Hidden Markov Models or HMMs are useful in finding and modeling motifs or patterns over time and have a lot of applications. Markov chains are probabilistic based state pattern based machines in which the next state depends on the previous state and are deterministic. In some cases Markov modeling cannot describe the system sufficiently and we need to describe some variable without even seeing it from other parameters; we use HMMs in such cases. In other words the underlying system is hidden and changes with time and there are observable states which are related to hidden states. Markov chain's each state maps to a physical event that is observable. It differs with HMMs with the fact that observation is a probabilistic function of the state and we are never directly measuring the observable states. HMMs are expressed as phi = (T,E,pi) where

T: transitional probability matrix

E: observation symbol probability matrix

pi: initial state probability vector.

There are three steps to model HMMs :

1. Evaluation - Probability of an observed state given a HMM (state estimation) - p(O / phi)

2. Decoding - Finding the sequence of hidden states that generated the observed state (inference) - Q* = argmax p(Q/O)

3. Learning - Generate HMM given sequence of observation. Estimate HMM parameter phi = (T,E,pi) We make the following assumption in HMMs (Independence

assumption)

Where pi(s) = initial probabilities

t(s) = transitional probabilities

e(s) = emission probabilities

The complexity is in the order of O(TN$\hat{2}$) where T, N are number of time steps and states respectively. State is estimated by Forward-Backward algorithm, Inference by Viterbi decoding and Learning by Baum-Welch algorithm. We use the Baum-Welch for the learning phase of the gestures and getting the parameters (Fig 2.16). We get HMMs1-n parameters after training n classes of motifs and these now shall help in recognizing gestures. We return the gesture with the highest probability state and return it. We use Bayes classification for it.

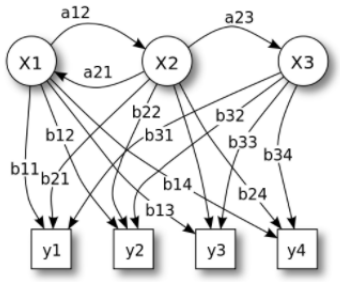

Figure 2.16: Probabilistic Parameters of a Hidden Markov Model

2.7 Results

We describe the experiment process and compare the results of the motif classification of the framework vs other algorithms.

2.7.1 Experiment on Fast Data Sensors

We take examples of three applications of high baud rate sensors on accelerometer - hand gestures, human activities in offices and exercise classification. We also test against slow baud rate sensor like temperature sensor. We use android mobile as the IoT device as it makes the setup in one package and provide libraries for simplification. We use the accelerometer sensor in the mobile to capture gestures, do preprocessing, create motif classes and train them and finally test against gestures.

2.7.1.1 Application 1: Hand Gesture Recognition using Accelerometer

For hand gestures, we have 20 different patterns which we train our framework with 50% data and test with remaining data (Fig 2.18). Data is preprocessed as discussed earlier before training. The dataset has 20 different gestures or motif class each having 20 repetitions of each gestures from 8 users. So there are 20*20*8 = 3200 gestures accelerometer data. The dataset has accelerometer data (x,y,z axes) with time stamps. The data was taken from Sony smart watch. The rest 50% of the data is tested with the proposed algorithm vs various motifs recognition algorithms discussed previously. The gestures data set is given below with the black arrows showing gestures with gesture number.

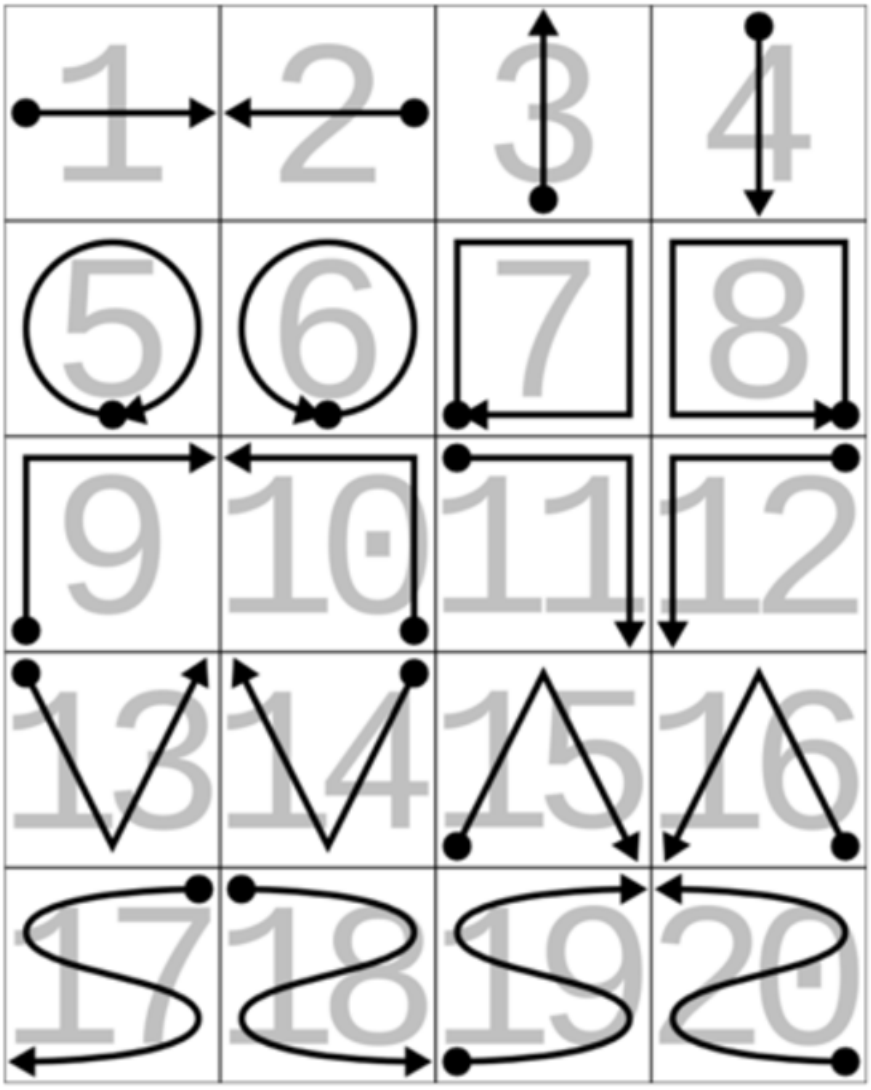

Figure 2.17: Hand Gestures Motif Classes

We train the gestures with the HMM method after the gestures are passed through pyramid model for dynamic scaling. Dynamic scaling shall not only make training set more but also shall cover different morphology of signals hence giving better accuracy in real world (Fig 2.19).

We note down the mis-classification of motifs for each algorithm and calculate the efficiency.

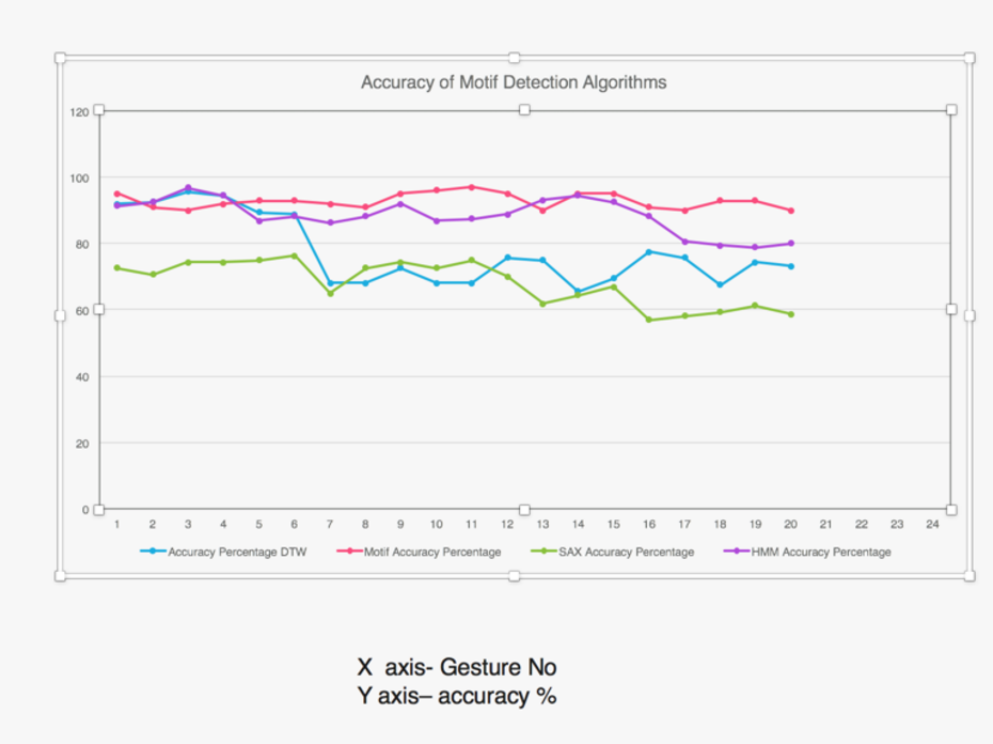

Figure 2.18: Accuracy of Motif Framework vs Other Algorithms on Hand Gesture Dataset

2.7.1.2 Analysis

We see that SAX algorithm although is fast performs bad for complex or large window gestures due to the fact that most gestures are similar or laterally inverted and SAX is unable to distinguish between them. The DTW algorithm performs well on simple gestures and dynamically adjusts for longer duration similar motifs giving better accuracy for simple gestures but performs poorly for complex gestures. Markov models perform well on patterns and detect have less accuracy with two similar

patterns as the hidden Markov parameters are almost same. Also for HMMs, same gestures having different timings are not classified that well. The proposed motif classification algorithm on the other hand having more training due to synthetic warped data performs better around 10-15% in real world scenario. We use this algorithm to build our framework. Thus the proposed motif framework which has dynamic time and amplitude scaling performs better than the other algorithms.

2.7.1.3 Application 2 : Exercise and Office activities

Different kinds of exercises are one of the most common activities we do in our lives. We extend our framework to this application. We choose exercise as application and common exercises as push up, twist, sit up and jump as motif classes. We trained and tested the framework for various applications and motifs for 4 users and 80 iterations on each motif.

We choose another simple application as office activities as it has potential for context awareness like controlling lights, logging off computers etc. So we choose simple desk actions we perform in our office like sit to stand, stand to sit, rotate chair etc and classified them. We found our algorithm classifying 80% of the actions correctly. We test it using android phone and motif based framework. (Table 2.1 & Table 2.2)

We can extend the framework for any kind of application such as daily human activity recognition (walking, sit to stand, stand to sit, rotate etc), exercise, gait of walking etc.

Table 2.1: Results for different exercise classification

Motif Class	No of Mis-classified Motifs	Accuracy
Push ups	7	91.25
Clockwise twist	12	85
Anti clockwise twist	14	82.5
Sit ups	10	87.5
Jumps	6	92.5
Average		87.75

Table 2.2: Results for different activities performed in offices

Motif Class	No of Mis-Classified Motifs	Accuracy
Sit to stand from chair	14	82.5
Stand to sit	16	80
Rotate clockwise on chair	7	91.25
Rotate anti-clockwise on chair	6	92.5
Average		86.56

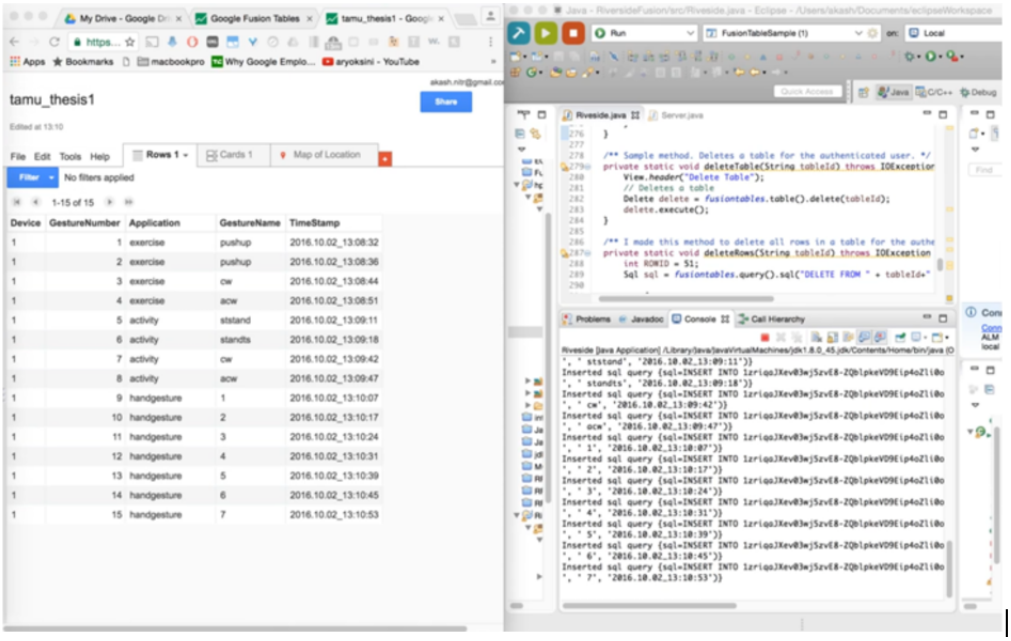

Figure 2.19: Pushing Motifs to the Google Fusion Tables over Internet using OAuth

2.7.1.4 Storage and Network traffic:

As discussed of the advantage of the motif framework we now store only the class information and the events of happening instead of storing the whole raw data. So for the 3200 gestures the dataset with time stamp was around 40MB. With our motif framework only the motif class and time stamps were stored for the 3200 gesture events and it was around 7MB including the raw data for the unrecognized gestures. So we consumed around 82.5% less storage and network traffic. So the framework saves around 80% of the expensive resources of the vendor by doing computation on the edge nodes for gesture recognition application. Similar results can be achieved for other physical processes.

Concluding: Generally in IoT, server is used for all the real time computation of the sensor data. With large number of IoT devices, the server is not capable of

processing such huge amount of IoT data in real time. The Motif framework has huge potential for scalability as it reduced the amount of raw data and only send the analyzed data. So the data is stored in structured form and user can query the events fast. Instead of transmitting the complete raw sensor data, we classify them as motif or anomaly and transmit the motif class attributes. The server having the motif information can populate it any time. Also we send the dynamic scaling information for accurate reproduction.

2.7.2 Experiment on Slow Data Sensors

Slow data rate changing such as temperature sensor transmit lot of redundant data. These kind of sensors generally measure physical quantities such as temperature, pressure which change slowly. The whole system being naive transmits redundant data every few seconds and it does waste lot of resources in storage, transmission, analytics and visualization. Our proposed framework transmits data when the physical quantity measure changes by a tolerance factor defined. So instead of sending the redundant data, it transmits data whenever it is changed (Fig 2.21). The server interpolates the missing redundant data. We deploy observer pattern that polls the physical quantity for changes. We performed the experiment with different tolerance levels for data similarity (Fig 2.22). Data is transmitted once the data changes above or below tolerance level. We also see the time series trends like ticks, monotonic increasing or decreasing etc. We take three months of temperature data of the city of Austin to test against our framework (Fig 2.20). We also find the number redundant package prevented increasing exponentially when we increase the tolerance level. (Table 2.3)

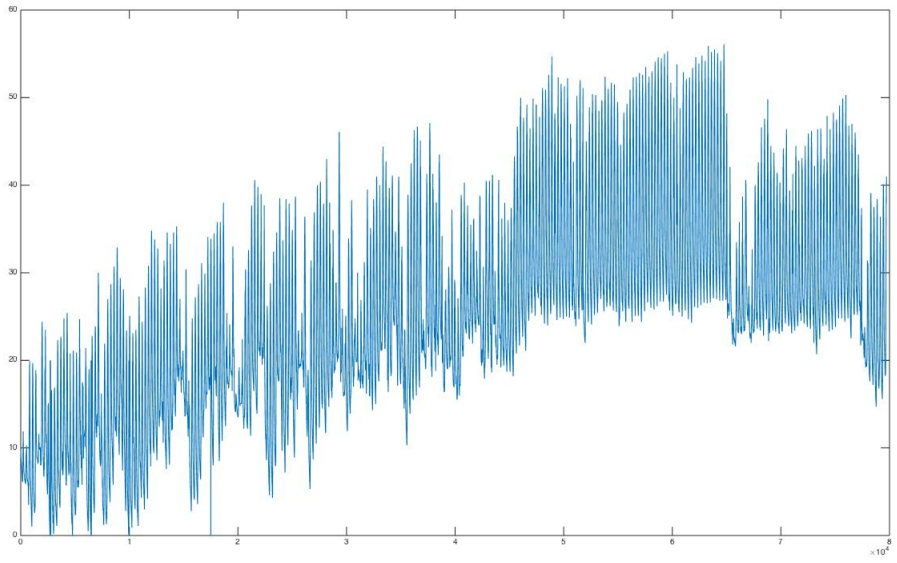

Figure 2.20: Austin City 3 Months Temperature Data

Table 2.3: Percentage of redundant data prevented from transmission with tolerances level

Sl No.	Tolerance(Celsius)	packets of data transmitted	percentage of redundant data prevented
1	transmit all	79728	0
2	0	59588	25.2
3	0.2	49662	37.7
4	0.4	25987	67.4
5	0.6	29998	62.3
6	0.8	23524	70.4
7	1	19023	76.1
8	1.2	18003	77.4
9	1.4	14446	81.8
10	1.6	14075	82.3
11	1.8	12406	84.4

2.7.2.1 Results

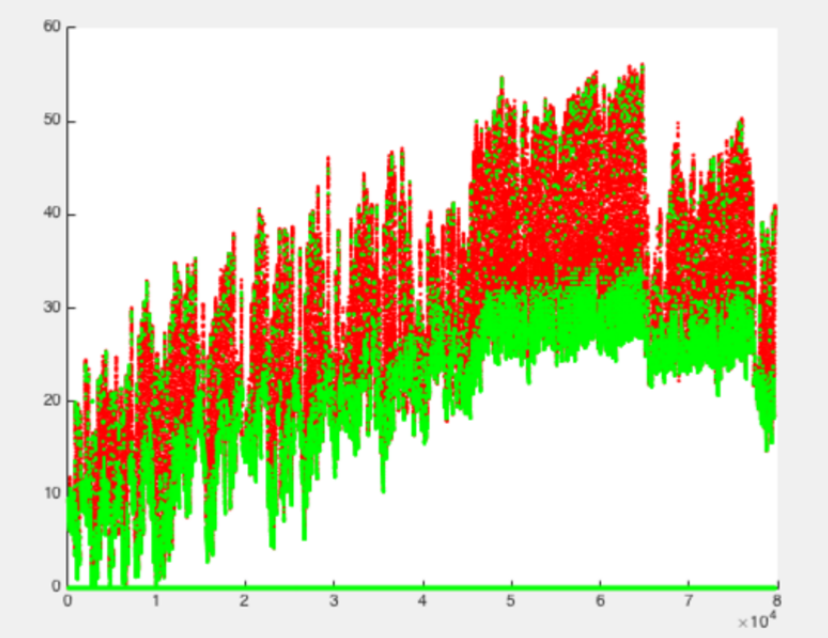

Figure 2.21: Redundant Packets of Data prevented from Transmission with Zero Tolerance

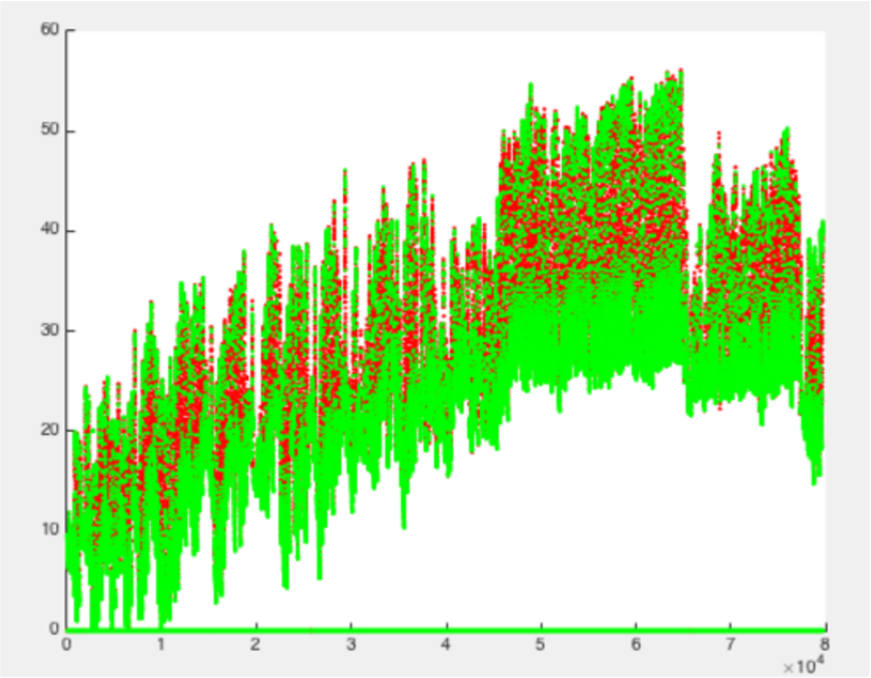

Figure 2.22: Redundant Packets of Data prevented from Transmission with 0.1 degree Tolerance

Thus we saved lot of redundant sensor data packets transmission. The green packets are the packets the server interpolates and are not transmitted. We save 80% of the redundant data transmission and storage from the sensors to servers.

2.8 Communication, Storage, Server

Communication of sensor data, meta information is one the most important portion of IoT framework as it requires lot of energy resources. The architecture of the server system can be majorly divided into components as wifi server and listener, data integrity check, storage, database, visualization, encryption and event controller. The packet data consists of the size of the packets raised to the power 2, the client ID of the IoT edge nodes, the motif classified or anomaly class and the

time stamp at which it receives packet. (Table 2.4) It also receives the length of the motif to adjust the frequency and scaling factor of motif to adjust the class data.

Table 2.4: Data packets anatomy

Packet size(3)	Client ID(8)	Motif Class (8)	Time(5)	Scaling Factor(8)

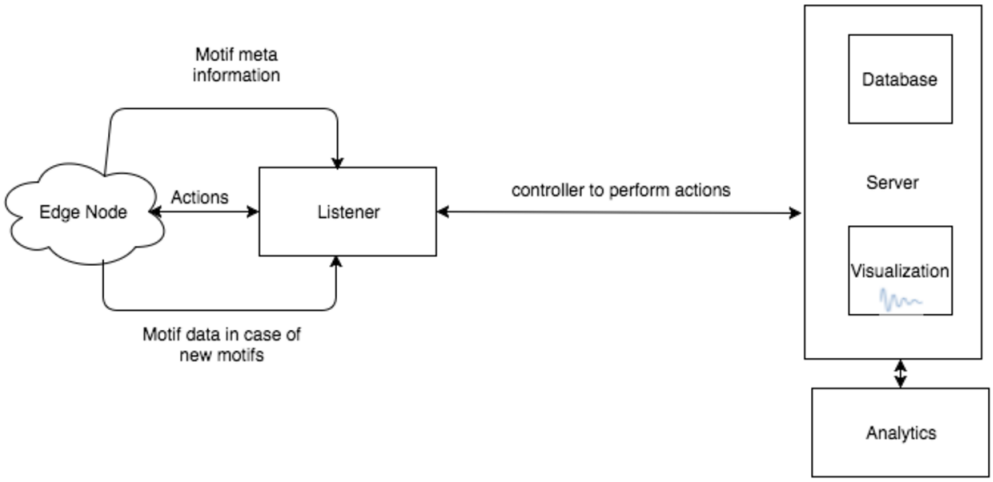

Figure 2.23: Server Components

2.8.1 Server Listener and Controller

The server listens to motifs classes sent by the IoT clients (edge nodes) and carries on the handshaking, checking of packets for sanity. If it detects unknown class, new class or anomaly, it ask client to send the whole motif data and it labels it appropriately. In case of anomaly it stores data in a sandbox for further analysis. In case of slower baud rate sensors like temperature sensors, it receives only changes

in data like an observer pattern, therefore it is responsible to interpolate data and send server to energy saving mode etc. The controller can block the edge node if it has been infected or there is any kind of attacks.

2.8.2 Storage and Database

The framework does an excellent job and saving storage resources. Traditional IoT frameworks on server store the raw data from the edge nodes and therefore they consume lot of storage. In case of temperature sensor it shall store the event of temperature change. The proposed framework shall store only the motifs class information in case of accelerometer sensors. On an average for a 100 Hz accelerometer sensor, with a 2sec motif, it shall save 200 data values storage. Also when we get previously occurred motif we update the motif count and time stamp. We get more than 200x storage space optimization. Also now looking up data shall be faster, articulated. The motifs are stored in structure SQL database with meta information like time stamp, scaling, motifs class, anomaly, number of occurrences, duration etc. We store the motifs actual raw data in a secured abstract sandbox so that normal user cannot modify the sensitive information. Encryption and security has been discussed in detail in the next chapter.

2.8.3 Visualization

Visualization of data, database can show inferential statistics and can give decision to applications. Building a real time visualization for IoT shall enable systems and user to take immediate actions in real time. With enormous amounts of sensor and their data, it becomes overwhelming to make sense. Also to label such huge amounts of data is difficult. The proposed framework does this task before hand, so there is no need to label the data. Even the anomalies are labeled before. It uses look-up table and hashing to generate data from the database using the time

stamps, duration, scaling properties. Therefore it is much faster than the traditional IoT frameworks which read huge amount of data, analyze them, label them and display them. In case of temperature sensor the visualization module interpolates data continuously until the new changed temperature data arrives.

3. ENCRYPTION & SECURITY

Cloud computing has been one the most emerging areas of development recently and has the entire world depending on it. It has carried all our banking records, social media, work, medical records and other sensitive data. We have been trusting our important information to be secured on the cloud, and it has churned out billions of dollars out the pockets of large corporations to maintain security. With the advent of big data, it has become a major headache for the cloud vendors.

Currently we have been using encryption solution such as AES which have limitation on computation over the encrypted data. The data has to be decrypted first on the cloud to be able to do computation on it. Decrypting on the cloud opens possibilities of data theft and interception to hackers who can easily access data from taint analysis, buffer overflow attacks etc. Leakage of critical data can be costly to corporations and people. Thus we need to perform computation on confidential data without decrypting them in untrusted environments.

The proposal here is to encrypt data before sending to the cloud vendors and allow them to perform computation on the encrypted data without decrypting them. To achieve this feat there are special kinds of encryption schemes such as Fully Homomorphic encryption or Partial Homomorphic encryption that have special property of performing computation on encrypted data. It shall enable queries on encrypted data to protect sensitive information. It should also provide fine-grained access control to support secure sharing between individual/organizations.

For over 30 years cryptographers have tried to build an encryption scheme that shall encrypt data right from the user side and shall be stored in encrypted form and be do arbitrary computation on encrypted data. Though it sounds simple but it has

been notoriously difficult to achieve this feat. The major limitation of implementing such schemes is that they are computationally very expensive and run too slow for practical purposes. We have chosen partial Homomorphic encryption, ElGamal algorithm based on elliptic curves for encryption on edge line embedded systems (Fig 3.1). These are one of secure encryption schemes. The major limitation is computation overhead but as we transmit just the class information, the data encrypted is small, the encryption cost is not that high.

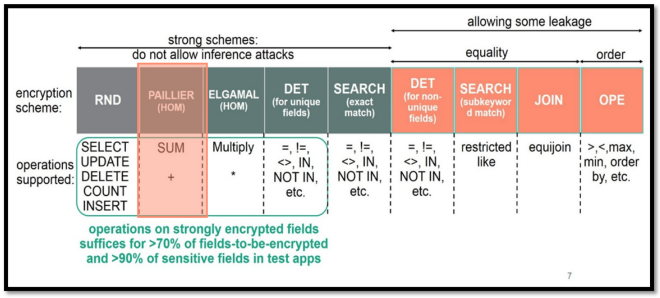

Figure 3.1: Operation on Encrypted data in SQL Database

The number of occurrences of the motifs are stored in the SQL database. Any computation to it can be done by Homomorphic addition which is the multiplication of the two big encrypted numbers. We do not need to decrypt the data to do computation. Also as we transmit just the motif class instead of raw data, the intruder cannot know what the raw data is.

3.1 ElGamal Encryption

ElGamal Encryption scheme was written by Tahel Elgamal in 1985. It is partial Homomorphic encryption that allows simple computation such as addition and multiplication on encrypted data. Homomorphic is an object that preserve maps

between two algebraic operations and if the encryption is homomorphic we call it as Homomorphic Encryption (Fig 3.2). The equation below describes homomorphic addition

$$E(a) \oplus E(b) = E(a \otimes b) \tag{3.1}$$

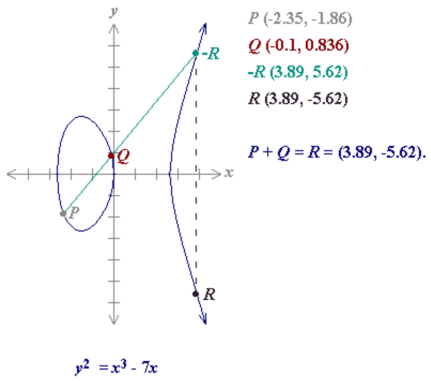

Figure 3.2: ElGamal Addition

3.2 Optimization

We compare run times of AES encryption vs ElGamal and we see that the AES encryption is faster than that of ElGamal encryption and therefore it has not yet been commercialized but when we compare the results of computation on encrypted data we find that the ElGamal algorithm performs better than AES. The reason is that AES has to first decrypt data, do operations on it and again encrypt back the sum and numbers whereas in case of ElGamal it just has to perform addition or multiplication without the need to perform encryption or decryption. This motivated us to improve speed of encryption of ElGamal algorithm.

AES:

We read N random numbers from file

Then we encrypt them, decrypt them, perform

addition, encrypt numbers, sum and then return the decrypted sum.

ElGamal:

We generate keys for two large random prime numbers

Then read N random numbers from file

Finally we encrypt the numbers, perform Homomorphic addition

return the decrypted sum

The general optimization of a real time application or library is followed as follows.

1) Software Parallelism - Batch Processing [16]

2) Hardware Parallelism

a. OpenMP

b. Batch + OpenMP

c. SIMD

d. OpenCL

3.2.1 Software Parallelism

The optimization is based on the notion that instead of doing computation on each element (32bit), multiple elements could be merged together in a single large element (128bit) and computation can be done on multiple large elements at once (Fig 3.3). This however has limitation of preprocessing elements into batches and also to access the element the whole batch needs to be decrypted. Also space needs to be left for carry overs.

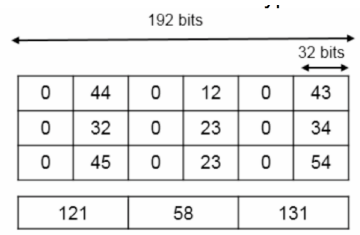

Figure 3.3: Software Batch Processing

We added two 32bit numbers into 128 bit vector padding them with 32 bits of zeros for carry overs during computation. All the encryption and Homomorphic addition were done on the 128bit batched numbers. We observe speedup of upto 2x with the optimization. If 16 bit number are taken four at once we achieve 4x speedup with it.

3.2.2 Parallelism using Hardware

We use dedicated hardware which were idle to improve speeds of computation. Traditionally we can do task or data parallelism. Task parallelism is done using threads or multiple process and require additional creation of resources for each task. Data parallelism can be done using special compute units like SIMD etc which computes data in batches at once. Same data operations are performed on multiple data streams.

3.2.2.1 OpenMP

A thread is the smallest unit of processing usually part of process doing specialized that can be scheduled by an operating system. Today's generation processor are multicore and have two threads per core generally. We can use the capabilities of hardware to run our dataset in parallel (Fig 3.4).

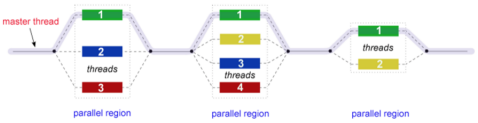

Fig. OpenMP fork - join

Figure 3.4: Fork-Join Model of OpenMP

OpenMP is a wrapper for creating, managing and running the threads. It deploys a fork join process for individual threads with a master thread managing all. Dependency, data race conditions, critical sections needs to be taken care while using OpenMP else it might give wrong results.

We tested the algorithm with 1,2,4 and 8 threads on the server. Data was divided accordingly for threads and the whole process was done on them in parallel and we took timings of the encryption and addition module. We achieved a 8x speedup using optimization.

We performed the same experiment but in this case the data was in batches of two and we got gain of around 16x. We used the batch processing optimization made in the described section Software Parallelism.

3.2.2.2 SIMD

SIMD or Single instruction, multiple data is multi processing units that perform same operation on data in batches. The SIMD have large registers that are able to fit multiple lanes of smaller data and do computation at once. They have different assembly language to code which makes it difficult for programmer to code on them. Traditionally in encryption the encrypted numbers are big. So to do multiplication we need to do it recursively by breaking into smaller numbers. We follow Karatsuba multiplication algorithm to do so (Fig 3.5).

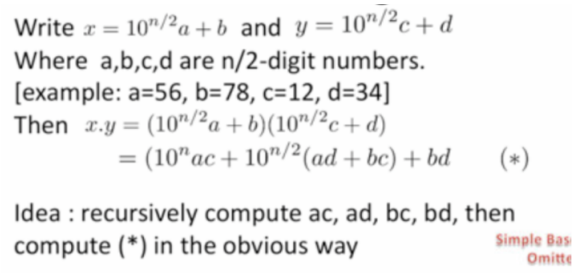

Figure 3.5: Karatsuba Recursive Multiplication

We write functions of 128 bit and 256 multiplication for large numbers using SIMD registers (Fig 3.6).

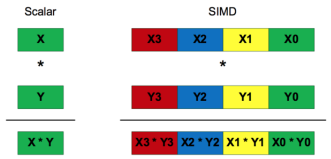

Figure 3.6: SIMD Addition in Batches on SIMD Registers

We find gain of around 1.6x using the SIMD registers. The gain is not much as the data is already BigNum.

3.2.2.3 OpenCL

OpenCL or Open Computing Language is a specification and not technology and maintained by Khronos Group. It helps heterogeneous computing environment to work together to do computation (Fig 3.7). The smallest bit of computation is called Kernel which run on all compute unit multiple times. We define the read, write

buffer which is responsible to transfer data batch and forth the compute devices. The compute devices can be GPUs, CPUs, FPGAs etc. We used on board GPU of Intel to gain speedups.

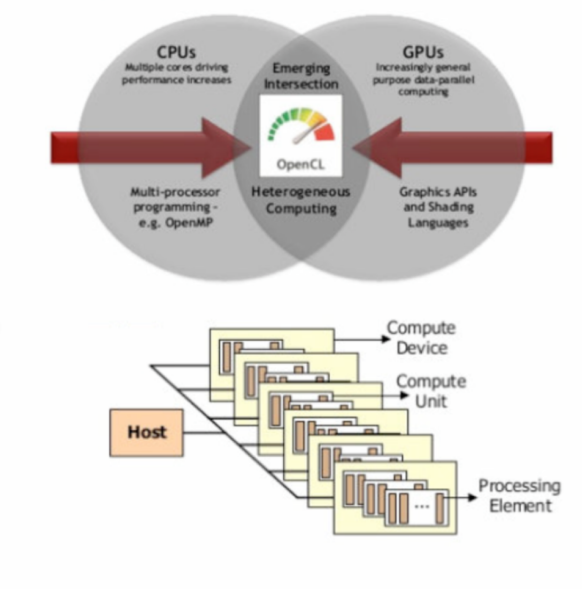

Figure 3.7: OpenCL Compute Model

Traditionally GPU have higher bandwith of 3-4GB/s but the communication bandwidth can be limiting. So we have to take that into account to submit higher computation intensive tasks to GPU. Also GPUs do not handle error well and are not that smart. This makes it difficult to debug on them. Also data needs to be arranged in a specific way to take advantage of GPU. OpenCL is not suitable for sequential problems or calculation that require a lot of pointer changing.

OpenCL consists of Compute objects and Memory Objects (Arrays, Images) and Executable Objects (Compute program, Compute Kernel).Compute devices may be grouped together to form device group A compute program is a group of compute

kernels and functions that is responsible for the whole application .A unit of work is called a work-item and Work items are grouped into a work- group. These form the core of OpenCL.

We implement Karatsuba Multiplication on GPU using OpenCL. We implement 128bit and 256 bit multiplication and achieve gain of 6-7x.

3.2.3 Results

We implemented several software optimizations for Partial Homomorphic encryption

- Batch based optimizations resulted in a gain of 2x
- Thread level parallelism using OpenMP resulted in a gain of 8x
- Combining Batch optimization with OpenMP resulted in a gain of 16x
- Implementation of Karatsuba algorithm on SIMD 256 bit registers gave speedup of 1.6x as compared to BigNum multiplication
- Implementation of Karatsuba algorithm on OpenCL gave speedup of 6.25x as compared to BigNum multiplication
- So we combined everything to get overall gain of 100x which enabled us to use Homomorphic encryption on HPE IoT edge devices

Now we shall be able to use ElGamal encryption in our framework as it shall have comparable run times with respect to AES.

4. CONCLUSION

The thesis has left a lot of spaces for future work. There is a need to make the framework as a custom package for different architecture and platforms. Analytics and Statistical Modeling on the data libraries needs to be added to the framework. Use of custom hardware accelerators supporting BigNum computation for encryption and machine learning algorithms can be done which shall give a huge speedup. Extend the code book algorithms for different application to improve better accuracy can be beneficial and flexible to the framework. More Visualization tools can be developed to give better inference to the user. Other complex computation function such as modulus, exponent, Division of BigNum in SIMD, OpenCL needs to be implemented

Internet of things is the upcoming field of this era. It is the next big thing in computing after desktop, laptop, tablet, mobile. It shall be a part of our lives. We discussed a lot of factors factors and problems that limit the scalability of these devices to be part of our lives. This research is a step towards it to solving problems of IoT. It shall save huge platform costs. We offered a framework that has huge potential to solve the major problems in the IoT field while proving the security as well. It has potential to save 90% of the vendor network and storage requirements. We developed the time series motif detection algorithm and made framework out of it for mobile devices.

BIBLIOGRAPHY

[1] Cisco. Cisco visual networking index: Forecast and methodology. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html.

[2] A. Mueen and E. Keogh. Online discovery and maintenance of time series motifs. *SIGKDD*, 2010.

[3] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[4] N. Koudas P. Indyk and S. Muthukrishnan. Identifying representative trends in massive time series data sets using sketches. *In Proceedings of VLDB, Egypt*, page 363–372, 2000.

[5] S. Muthukrishnan A. C. Gilbert, Y. Kotidis and M. Strauss. Surfing wavelets on streams: One-pass summaries for approximate aggregate queries. *In Proceedings of VLDB, Italy*, pages 79–88, 2001.

[6] J. Gehrke V. Ganti and R. Ramakrishnan. Mining data streams under block evolution. sigkdd explorations. *SIGKDD Explorations*, pages 1–10, 2002.

[7] C. Faloutsos Y. Sakurai and M. Yamamuro. Stream monitoring under the time warping distance. *ICDE*, pages 1046–1055.

[8] M. Joselli and E. Clua. gr mobile: Framework for touch and accelerometer gesture recognition for mobile games, in: Games and digital entertainment. *VIII Brazilian Symposium on, Rio de Janeiro, Brazil*, 2009.

[9] S. Pirttikangas M. Kauppilla, T. Inkeroinen and J. Riekki. Mobile phone controller based on accelerative gesturing. *Sixth International Conference on Pervasive Computing, Sydney*, 2008.

[10] R. Cilibrasi and P. Vitanyi. Clustering by compression. *IEEE Trans. Inf. Theory*, pages 1523–1545.

[11] S. Hawkins III and E. Darlington. Algorithm for compressing time-series data. *NASA Tech Briefs*, pages 7–8, 2012.

[12] H. Jeung N. Hung and K. Aberer. An evaluation of model-based approaches to sensor data compression. *IEEE Trans. on Knowl. and Data Eng*, pages 2434 – 2447, 2013.

[13] S. Bandyopadhyay A. Ukil and A. Pal. Iot data compression: Sensor-agnostic approach. *Data Compression Conference*, 2015.

[14] J. Shieh A. Camerra, T. Palpanas and E. Keogh. isax 2.0: Indexing and mining one billion time series. *ICDM*, pages 58 – 67, 2010.

[15] T. Acharya S. Mitra. Gesture recognition: A survey. *IEEE Transactions on Systems, Man and Cybernetics*, Volume 37 Issue 3:311–324.

[16] A. Hithnawi H. Shafagh and A. Dröscher. Talos: Encrypted query processing for the internet of things. *ACM Sensys*, pages 197–210.