Preserving Data Privacy for Smartwatch Atrial Fibrillation Detection Using Secure Multiparty Computation

A Thesis

Presented in Partial Fulfillment of the Requirements for the

Degree of Master of Science

with a

Major in Computer Science

in the

College of Graduate Studies

University of Idaho

by

Maria Swartz

Major Professors: **Dr. Jia Song**, **Dr. Constantinos Kolias**

Committee Member: **Dr. Min Xian**

Department Administrator: **Dr. Terence Soule**

May 2023

Abstract

During the past few years, the introduction of a new communication paradigm known as the Internet of Things (IoT) has revolutionized sectors such as manufacturing, transportation, and retail. Inevitably, healthcare has been deeply impacted by the proliferation of these smart devices. More specifically, IoT has been applied to tracking important health parameters, recording vital signs, and finally providing accurate diagnoses regarding some health conditions. For example, smartwatches are capable of identifying early signs of heart conditions such as certain types of arrhythmias, and provide early warnings.

Behind the scenes, the diagnosis is done by relying on Artificial Intelligence (AI) and Machine Learning (ML) algorithms that capture heart-related data through the use of a sensor. However, due to the high demands in processing, memory, and energy resources, this processing is rarely conducted locally, at the smartphone level. Typically, the data analysis is done in the cloud through an infrastructure owned by a "not-necessarily trusted" third party. However, this sharing of sensitive personal health data often raises privacy concerns for consumers. This in turn may act as a roadblock to the adoption of IoT in healthcare. More specifically, two concerns with this approach are that (a) the centralized party may misbehave or become compromised and (b) the data may be compromised by a passive eavesdropper while it is in transit.

In our work, we adopted and modified a method that allows the analysis of heart signals in a privacy-preserving manner, namely *secure multiparty computation (SMPC)*. We specifically applied the method to the problem of Atrial Fibrillation (AF) detection using a smartwatch. This method allows parties to perform joint computations on protected versions of private data, often referred to as *secret shares*, without compromising the private data itself. By using SMPC to perform the mathematical operation of addition in a privacy-preserving way, we enabled the lightweight computational nodes i.e., smartwatches to perform the most critical operations that are involved in ML processes without the need for the nodes to know each other's data. The main contributions of our work are (a) the development of a framework to apply SMPC to the problem of AF detection and (b) the evaluation of the overhead in terms of loss of accuracy and additional communication cost of the secure model compared to traditional centralized scenarios.

Acknowledgments

I would like to thank my advisors, Dr. Jia Song and Dr. Constantinos Kolias, for their guidance and mentorship throughout this project. I would also like to thank my fellow team member Koffi Anderson Koffi for providing additional guidance and support.

Moreover, I would like to express my gratitude to members of the NSF CyberCorps®: Scholarship for Service Program for their mentoring and financial support. This material is based upon work supported by the National Science Foundation under Grant No. DGE-1565572.

Dedication

I would like to thank my parents Kathy and Greg and my brother Paul for their support throughout this project.

Table of Contents

Abstract ii

Acknowledgments iii

Dedication iv

Table of Contents v

List of Figures viii

List of Tables x

List of Acronyms xi

Chapter 1: Introduction 1

1.1 Problem Description 2

1.2 Proposed Solution 3

Chapter 2: Technical Background & Definitions **5**

2.1 Morphology of ECG Signals 5

2.1.1 Structure of the Heart 5

2.1.2 Elements of ECG Signals 6

2.1.3 12-lead ECGs 7

2.1.4 Alternatives to 12-lead ECGs 11

2.2 Arrhythmia Biomarkers 12

2.2.1 Understanding Atrial Fibrillation 12

2.2.2 Detecting Atrial Fibrillation 13

2.3 Secure Multiparty Computation 15

2.3.1 Basic Background 15

2.3.2 Secret Sharing 16

2.3.3 Secure Multiparty Addition 16

2.3.4 Shamir's Secret Sharing 16

2.3.5 Shamir Secret Sharing Addition Example 18

2.3.5.1 Lagrange Interpolation 19

2.4 Application: Secure Euclidean Distance 20

2.4.1 Secure Euclidean Distance in Two-Dimensional Space 20

2.4.2 Secure Euclidean Distance in N-Dimensional Space 23

Chapter 3: Proposed Framework **26**

3.1 Data Preprocessing & Feature Extraction 26

3.2 Traditional Framework 27

3.3 Secure Framework 29

Chapter 4: Experimental Evaluation **36**

4.1 Dataset 36

4.1.1 PTB-XL Database 38

4.1.2 St. Petersburg INCART 12-lead Arrhythmia Database 38

4.1.3 Brno University of Technology ECG Quality Database (BUT QDB) . . . 39

4.1.4 PhysioNet/Computing in Cardiology Challenge 2017 Database 40

4.2 Experimental Setup . . . 42
4.2.1 Traditional Model . . . 42
4.2.2 Secure Model . . . 42
4.3 Data Preprocessing . . . 43
4.4 Feature Extraction . . . 44
4.5 Experiment 1: Comparing Distance Calculations . . . 46
4.6 Experiment 2: Comparing Maximum X-coordinates in Lagrange Interpolation . . 50
4.7 Experiment 3: Comparing Accuracy with KNN Classification . . . 52
4.8 Experiment 4: Comparing Classification Time Efficient . . . 54

Chapter 5: Discussion . . . 57

Chapter 6: Related Work . . . 58
6.1 Atrial Fibrillation Detection . . . 58
6.1.1 Traditional, 12-lead ECG detection . . . 58
6.1.2 Smartwatch Detection . . . 59
6.2 Privacy-Preserving Methodologies . . . 61
6.2.1 Homomorphic Encryption . . . 62
6.2.2 Differential Privacy . . . 66
6.2.3 Federated Learning . . . 68
6.2.4 Secret Sharing . . . 69

Chapter 7: Conclusion . . . 72
7.1 Future Work . . . 73

References . . . 74

List of Figures

1.1 Global IoT Market Growth and Forecast [1] 1

2.1 Electrical Conduction in the Heart [2] 6

2.2 Labeled ECG Signal [3] 7

2.3 ECG Electrode Placement [4] 7

2.4 Heart walls [5] 9

2.5 Sections of an ECG Reading [5] 10

2.6 Limb Leads [6] 11

2.7 Normal Heart vs AF Heart [7] 13

2.8 Normal ECG vs AF ECG [8] 14

2.9 Annotated AF Biomarkers [8] 15

3.1 Traditional Framework Steps 28

3.2 Secure Framework Steps 31

4.1 Example record from the PTB-XL dataset as obtained with the LightWAVE tool [9], [10], [11], [12] 38

4.2 Example record from the St. Petersburg dataset as obtained with the LightWAVE tool [9], [10] 39

4.3 30 seconds of an example record from the BUT QDB dataset as obtained with the LightWAVE tool [9], [10], [13] 40

4.4 Example of a normal and AF record from the PhysioNet/Computing in Cardiology Challenge 2017 dataset [14] 41

4.5 ECG record A00015 before (top) and after (bottom) applying band-pass filtering 44

4.6 Pre-processed normal ECG record A00025 (top) and AF record A00009 (bottom) with R-peaks marked 45
4.7 Histogram of RR intervals for a normal (A00025) and AF (A00009) record 46
4.8 Distances between (a) normal vs normal and (b) normal vs AF data points (signals) for the traditional and secure scheme. 48
4.9 Distances between (a) AF vs AF and (b) AF vs normal data points (signals) for the traditional and secure scheme. 48
4.10 Absolute difference of normalized Euclidean distance of an example signal in the dataset with all the rest of the examples in the same dataset. 49
4.11 Sum squared difference between traditional and secure distance calculation for a point as the maximum x-value for Lagrange interpolation increases. 51
4.12 Confusion Matrix - Traditional Model 53
4.13 Confusion Matrix - Secure Model 54

List of Tables

2.1 Secure Multiparty Addition: Calculations and Communications 19
2.2 Two-Dimensional Euclidean Differences 22
2.3 Two-Dimensional Euclidean Lagrange Interpolation Calculations 22
2.4 Two-Dimensional Euclidean Distance: Calculations and Communications 23

4.1 AF classification datasets 37
4.2 Example RR interval feature vectors for normal record (A00025) and AF record (A00009) 45
4.3 Absolute differences in Euclidean distance calculation between traditional and secure computations 47
4.4 Comparing Statistics for KNN Models in the Traditional and Secure Approaches 54
4.5 Average time (in ms) for each networking and computation operation over 100 trials 55
4.6 Average message sizes (in bytes) over 100 trials 55

List of Acronyms

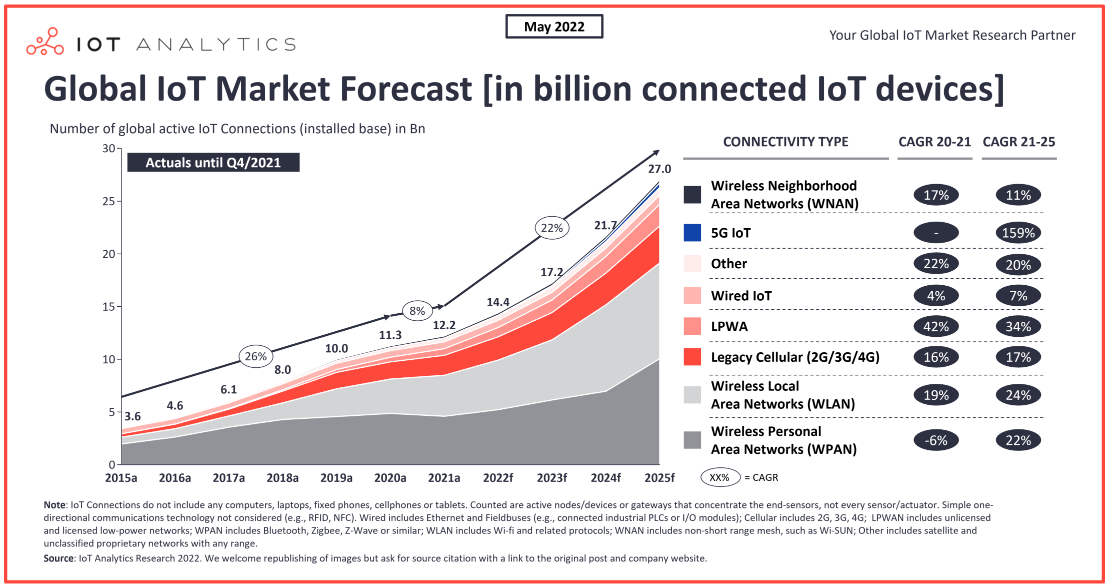
AI	Artificial Intelligence
AV Node	Atrioventricular Node
AF	Atrial Fibrillation
cPDS	cluster Primal Dual Splitting
CNN	Convolutional Neural Network
ECG	Electrocardiogram
FIR	Finite Impulse Response
HRV	Heart Rate Variability
JSON	JavaScript Object Notation
KNN	K-Nearest Neighbors
LTSM	Long Short-Term Memory
LWE	Learning With Error
LV	Left Ventricle
ML	Machine Learning
MPC	Multiparty Computation
PPG	Photoplethysmography

PPV	Positive Predictive Value
RA	Right Atria
RLWE	Ring Learning With Error
RSS	Replicated Secret Sharing
RV	Right Ventricle
SA Node	Sinoatrial Node
SMPC	Secure Multiparty Computation
SVM	Support Vector Machine
VM	Virtual Machine

Chapter 1

Introduction

In the last decade, the adoption of Internet of Things (IoT) technologies has grown exponentially. Figure 1.1, illustrates that the number of IoT devices grew to 12.2 billion in 2021, from 3.6 billion in 2015 [1]. Moreover, the global IoT market is projected to reach 27 billion devices by 2025 [1]. IoT has successfully found applications in a vast array of heterogeneous areas including manufacturing, transportation, and retail. Naturally, many of these new IoT devices fall under the category of wearable health monitoring devices. These devices can monitor a myriad of the user's physical variables including their activity levels, heart rate, respiratory rate, glucose levels, and temperature, among others.



Note: IoT Connections do not include any computers, laptops, fixed phones, cellphones or tablets. Counted are active nodes/devices or gateways that concentrate the end-sensors, not every sensor/actuator. Simple one-directional communications technology not considered (e.g., RFID, NFC). Wired includes Ethernet and Fieldbuses (e.g., connected industrial PLCs or I/O modules); Cellular includes 2G, 3G, 4G; LPWAN includes unlicensed and licensed low-power networks; WPAN includes Bluetooth, Zigbee, Z-Wave or similar; WLAN includes Wi-fi and related protocols; WNAN includes non-short range mesh, such as Wi-SUN; Other includes satellite and unclassified proprietary networks with any range.

Source: IoT Analytics Research 2022. We welcome republishing of images but ask for source citation with a link to the original post and company website.

Figure 1.1: Global IoT Market Growth and Forecast [1]

In particular, new-generation smartwatches are equipped with a lead sensor capable of monitoring detailed characteristics of heartbeats that go beyond the traditional heart rate. In fact, such devices are capable of capturing electrocardiograms (ECG) and have been used to detect a wide range of heart conditions, mainly certain types of arrhythmias. Atrial fibrillation (AF) is defined as *"an irregular and often very rapid heart rhythm (arrhythmia) that can lead to blood clots in the heart"* [15]. Having AF increases the risk of stroke 5-fold and is responsible for at least 15 to 25% strokes in the United States [16, 17]. Because AF occurs irregularly, it is often undiagnosed and untreated. 18% of stroke patients suffer from AF which was only diagnosed at the time of the stroke [16]. It is estimated that there are about 700,000 people in the United States with undiagnosed AF [16, 17]. This is a problem because oral anticoagulation can greatly reduce the risk of strokes by 49 to 74% for AF patients [16]. Therefore, AF detection is an area where an easy-to-access, easy-to-use, wearable device such as the smartwatch can be extremely useful. Smartwatches are attached to the user's wrist and they have the advantage over traditional short-term hospital checks because they can collect data about the user longitudinally.

1.1 Problem Description

Although wearables, due to their ease of use, have boosted the early diagnosis of AF, they are associated with privacy concerns regarding the user's data. Typically, the diagnosis is done via analysis of the user's data through Machine Learning (ML) methods. However, since smartwatches tend to have limited resources, most vendors *offload* critical parameters of the ML computations to cloud services i.e., a third party. Thus, in typical workflows, as a first step, the user's smartwatch device collects raw ECG signals and, as a second step, it sends the data (or certain features of the data) to a third party for diagnosis. As a third step, the third-party views and analyzes the data by performing a certain number of computations to determine if the user has AF. As a last step, they send the diagnosis result (normal or anomalous) back to the user. Notice, that the third party does not only have detailed access to the user's sensitive vital signs but also knows the results of the diagnosis.

Having an untrusted third party with access to and the capacity to store the user's health-related data or diagnosis results raises serious privacy concerns. More specifically, the third party may choose to share/sell such data with other vendors for advertising purposes or with insurance companies. Even if the third party operates in a benign way, there is always the fear of becoming the target of attackers and unwillingly leaking the stored sensitive data. Finally, the act of transmitting such sensitive information has its own dangers. Even if the data is encrypted during transmission, an eavesdropper may exploit vulnerabilities in the transmission workflows and gain access to the data while *in transit*. Finally notice, that traditional ML methods cannot perform computations directly upon encrypted data. The data may be encrypted on the client-side and be transmitted to the cloud securely, but at that point, the data must be decrypted before being stored and analyzed.

1.2 Proposed Solution

This work introduces a framework for the detection of heart conditions using smartwatch devices capable of capturing ECGs in a privacy-preserving manner. While the proposed framework is generic, our methods focus on the detection of a specific type of arrhythmia, namely AF.

The privacy mechanisms adopted by our framework rely upon the use of Secure Multiparty Computation (SMPC). SMPC allows nodes to communicate in a secure fashion to perform basic mathematical operations like addition, subtraction, and multiplication without the need of exposing raw data. Notice that these operations are the foundation of virtually all types of analysis including those used as part of ML say, for comparing normal to normal or anomalous examples and then proceeding to diagnosis. Thus, the proposed framework achieves privacy-preserving diagnosis without the need of sharing raw private data but rather a set of *shares*. All calculations are performed directly upon the shares, and the data never get *decrypted* [1]. Only a portion of the shares are transmitted publicly but having access to only a fraction of the shares does not allow an attacker to infer the original data. In this way, a third party never gains access to the raw data or the results of the diagnosis. At the same time, a passive eavesdropper that monitors the connection cannot make assumptions about the raw data.

[1]Here the term decrypted is used loosely as the raw values never get encrypted but are transformed to secret shares that do not convey any information.

Through simulations, we show that the proposed framework achieves the same degree of predictive accuracy while maintaining the latency of receiving the results at realistic levels.

Chapter 2

Technical Background & Definitions

2.1 Morphology of ECG Signals

An ECG signal shows details about the electrical activity in the heart and can help determine various health conditions of a patient, including the detection of AF. There are numerous different configurations and lead setups that can be used to get an ECG reading for a patient. To understand the morphology of ECG signals, we will begin by examining the structure of the heart, the different elements of ECG signals, the standard 12-lead ECG configuration, and different alternatives to 12-lead ECGs.

2.1.1 Structure of the Heart

To understand ECG signals, first, we must examine the general structure of the heart and how electric signals are sent within it. The heart is composed of four chambers. The *right atria (RA)* and *left atria (LA)* make up the top half of the heart, and the *right ventricle (RV)* and *left ventricle (LV)* are on the bottom half of the heart. In the RA, there is a bundle of cells called the *sinoatrial (SA) node*. This node is the heart's natural pacemaker. The SA node generates electrical activity that spreads throughout both atria chambers and causes atrial depolarization. During depolarization, both atria contract and push blood into the ventricles.

When the electrical signal reaches the *atrioventricular (AV) node* (which is located in the bottom right of the RA), the AV node makes sure that both atria are empty before passing along the signal to the ventricles. During this phase, the atria will repolarize and relax. Then, the signal travels from the AV into the atrioventricular bundle (bundle of His) which is responsible for conducting impulses from the atria into the ventricles. The right and left branches of the bundle of His will send the signal to the RV and LV, respectively. This will cause ventricular depolarization, so the ventricles will contract and push out blood into the arteries. The cycle completes when the ventricles repolarize and relax. This process is illustrated in Figure 2.1 [2].

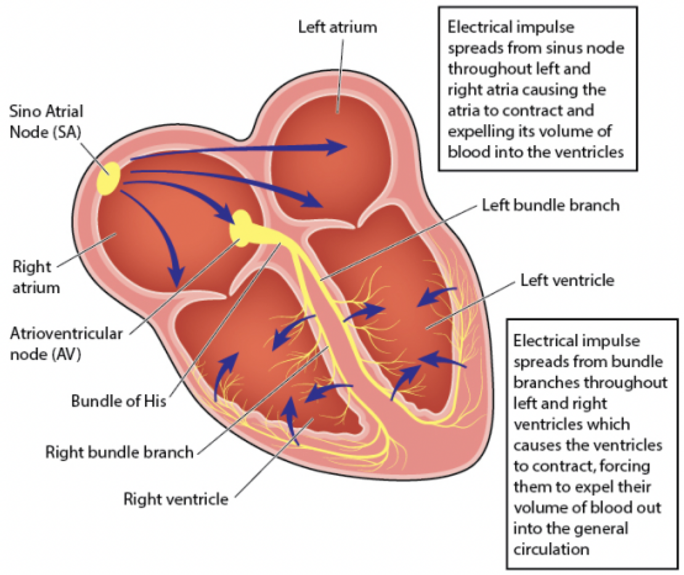

Left atrium
Electrical impulse spreads from sinus node throughout left and right atria causing the atria to contract and expelling its volume of blood into the ventricles
Sino Atrial Node (SA)
Left bundle branch
Right atrium
Left ventricle
Atrioventricular node (AV)
Electrical impulse spreads from bundle branches throughout left and right ventricles which causes the ventricles to contract, forcing them to expel their volume of blood out into the general circulation
Bundle of His
Right bundle branch
Right ventricle

Figure 2.1: Electrical Conduction in the Heart [2]

2.1.2 Elements of ECG Signals

An ECG signal can be examined by looking at the different waves that make up one cycle. The ECG reading should start off with a *P-wave* which shows atrial depolarization. Next, the signal will show the *QRS complex* which is composed of a *Q-wave*, *R-wave*, and *S-wave* that occur very close to each other. Two things happen in the QRS complex, atrial repolarization and ventricular depolarization. Since the ventricles are stronger than the atria, the atria repolarization is masked, but it happens slightly before ventricular depolarization. Depolarization is always followed by repolarization, so we should always expect to see a repolarization wave after depolarization. The final part of the signal is the *T-wave* which indicates ventricle repolarization.

To measure the heart's rhythm, we measure the distance between R waves. The number of

R-waves measured in a minute is the person's heart rate. We can also measure other intervals and segments of the signal including the PR interval, PR segment, QT segment, and the QT interval. A normal PR interval should be between .12 and .2 seconds, and the QRS complex should normally be between .08 and .12 seconds [18]. A standard ECG signal along with these intervals and segments is shown in Figure 2.2 [3].

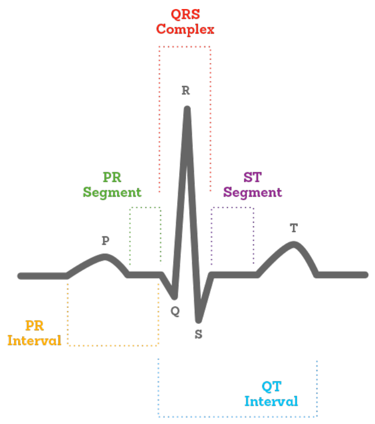


Figure 2.2: Labeled ECG Signal [3]

2.1.3 12-lead ECGs

ECG signals can be measured with different types of devices that may have different numbers of leads. ECG measurements are commonly taken with a 12-lead configuration. This configuration allows us to capture numerous different data points and make more deductions about the patient's heart.

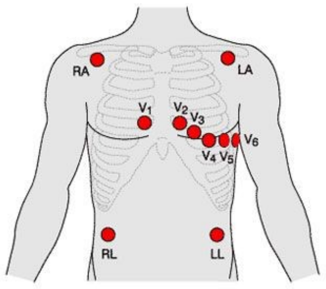


Figure 2.3: ECG Electrode Placement [4]

In Figure 2.3, we can see the typical placements for the electrodes in a 12-lead ECG [4].

Note that there are 10 electrode leads placed on the body. This highlights the difference between electrode leads and tracing leads. The electrode leads are the physical pads that are placed on the patient, and the tracing leads are ECG signals that are derived from the data collected from the electrodes. The electrodes can be organized into two groups, the chest leads and the limb leads. There are six chest leads and four limb leads. Each of these sets of leads will give us six readings which combined provide a 12-lead ECG reading. The chest lead readings are collected with the electrodes numbered V1 - V6, and the limb leads used the electrodes RA, LA, RL, and LL. Note, RA and LA are often placed on the wrists instead of below the collarbone, and RL and LL are often placed near the knees or ankles instead of on the waist. Only 3 of the 4 limb electrodes are use to provide readings. This is because the RL electrode serves as the ground and isn't directly used for the leads. The 6 leads that we obtain from the limb leads are referred to with code names as aVF, aVL, aVR, I, II, and III leads.

The 12 leads allow us to look at the electrical activity in the heart from different angles at the same time. Combining the different leads can be used to give us the following four different views of the heart (as shown in Figure 2.4 [19]):

1. **Interior Surface** - Leads: II, III, aVF

2. **Lateral Surface** - Leads: I, aVL, aVR, V5, V6

3. **Anterior Surface** - Leads: V3, V4

4. **Septal Surface** - Leads: V1, V2

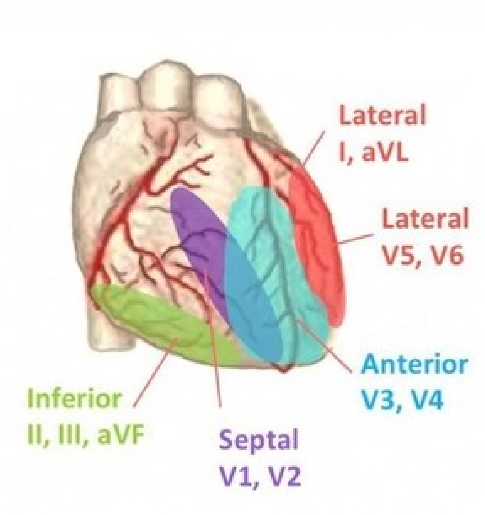


Figure 2.4: Heart walls [5]

We can see each of the 12 different lead readings on a standard, 12-lead ECG reading in Figure 2.5 [5]. The strip along the bottom of the figure is the rhythm strip and is usually the reading from Lead II. This strip is used to allow medical professionals to determine the patient's heart rate and can be helpful in diagnosing conditions like AF.

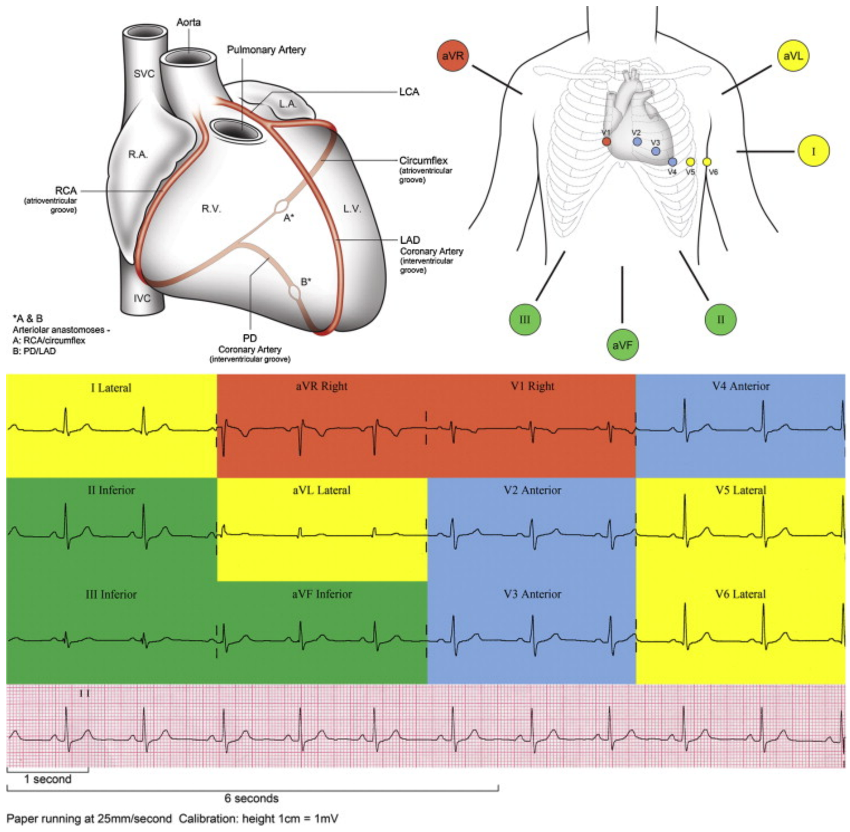

Aorta
Pulmonary Artery
SVC
L.A.
LCA
R.A.
Circumflex (atrioventricular groove)
RCA (atrioventricular groove)
R.V.
A*
L.V.
LAD Coronary Artery (interventricular groove)
B*
IVC
*A & B Arteriolar anastomoses - A: RCA/circumflex B: PD/LAD
PD Coronary Artery (interventricular groove)
aVR
aVL
V1
V2
V3
V4
V5
V6
I
III
aVF
II
I Lateral
aVR Right
V1 Right
V4 Anterior
II Inferior
aVL Lateral
V2 Anterior
V5 Lateral
III Inferior
aVF Inferior
V3 Anterior
V6 Lateral
1 second
6 seconds
Paper running at 25mm/second Calibration: height 1cm = 1mV

Figure 2.5: Sections of an ECG Reading [5]

The readings from each of the leads look different since each lead is placed in a different position in relation to the heart and the electrical signal. The readings show us the relationship between the general direction of depolarization and where each lead views this depolarization from. As a depolarization wave heads towards a lead, it causes an upward deflection from the baseline, and as the wave heads away from a lead, it causes a downward deflection from the baseline. We can use these rules when looking at the QRS complex. When depolarization heads towards a lead, the complex will be positive with a tall R-wave and a short S-wave. If the depolarization is heading away from the lead, the complex will be negative with a short R-wave and a deeper S-wave [6].

This is most easily seen on the 6 chest leads (V1 to V6) because they are organized on a horizontal plane [6]. V1 has the depolarization going away from it the most, so its QRS complex is very negative. For the chest leads, the QRS complex progresses from being very negative (V1) to being very positive (V5 and V6). The limb leads are a bit different because they aren't organized on a horizontal plane. Figure 2.6 shows the orientation of the 6 limb leads in relation to the heart and what a typical QRS complex would look like from each lead's point of view [6].

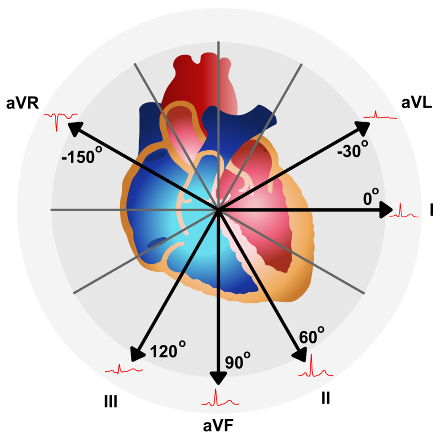


Figure 2.6: Limb Leads [6]

Since depolarization moves away from the right atria, the aVR lead will have the most negative QRS complex. The QRS complexes for the aVL Lead and Lead III both will have an R-wave that is about as tall as the S-wave is deep because the depolarization signal passes them at a right angle. Leads I, II, and aVF will have positive QRS complexes because the depolarization signal is heading toward them.

12-lead ECGs can be used to diagnose a variety of different heart arrhythmias including atrial fibrillation, atrial flutter, bundle branch blocks, AV branch block, and ventricular fibrillation.

2.1.4 Alternatives to 12-lead ECGs

Most wearable health monitoring devices that measure heart rate only have one lead which gives us less data to analyze than a 12-lead configuration. This means that while these devices can still be used to diagnose some medical conditions, they cannot diagnose as many conditions as a 12-lead configuration. An additional challenge with wearable health monitoring devices is that some noise may be introduced in the data. Most 12-lead readings are taken in a hospital under controlled scenarios. Wearable health monitoring devices normally take readings while the user is doing daily activities in a less controlled environment.

Wearable health monitoring devices typically give us the same readings as Lead I in the 12-lead configuration. Lead I can be used to detect both atrial fibrillation and atrial flutter. If we had three electrodes, we could set up bipolar lead monitoring. This setup typically uses Leads I, II, and III, but it may also use a modified chest lead (MCL) [20]. Its goal is to track heart rate, determine whether the R waves are synchronized, and detect ventricular fibrillation.

In general, Lead V1 is considered the best lead for diagnosing arrhythmias. However, some conditions like right and left bundle-branch block need multiple leads to be detected. This is commonly done with five-electrode limb leads and one additional precordial lead combination [20].

2.2 Arrhythmia Biomarkers

2.2.1 Understanding Atrial Fibrillation

In this section, we shall examine biomarkers that can be found in ECG signals that have a correlation and can be used for the detection of arrhythmias. This discussion will focus primarily on AF. AF occurs when electrical impulses are sent from places other than the SA node, called ectopic sites. These other electrical signals cause the atria to quiver or fibrillate instead of contract. Since these signals don't come from the SA node, they may arrive at the AV node at random angles, and many of them do not pass through the AV node. Since many of the signals don't go through the AV node, signals get to the ventricles at irregular intervals which makes the individual's heart rhythm irregular. The ectopic sites may not send these irregular signals all the time, so an *irregularly* irregular heartbeat is one of the main indicators of AF. The process of AF signals being generated and the resulting ECG signal is shown in Figure 2.7 [7].

Some of the other characteristics of AF include the absence of P-waves and irregular or narrow QRS complexes. Often, the narrow QRS complexes will be less than 120 ms. Additionally, the ECG's baseline may appear to undulate, which makes to baseline look like it is composed of many very small, irregular waves, or it may appear to be totally flat. In general, a higher number of ectopic sites results in a flatter baseline.

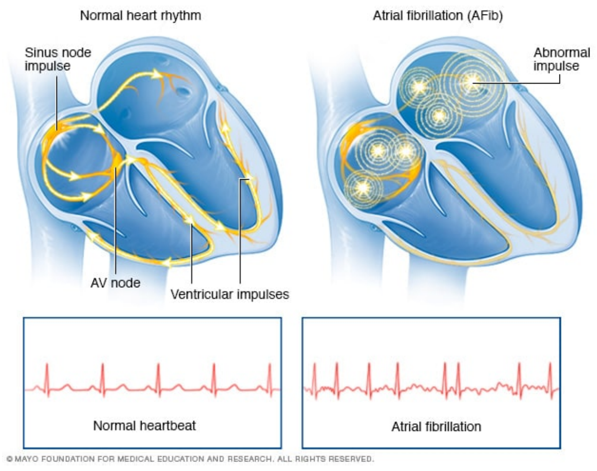


Figure 2.7: Normal Heart vs AF Heart [7]

There are 3 types of AF. **Paroxysmal AF** has transient AF episodes that last seconds to up to a week. These episodes stop on their own. **Persistent AF** occurs when the patient has episodes that last more than a week or episodes that last less than a week that can only be stopped through pharmacological or electrical cardioversion. **Long-standing AF** (also known as chronic or permanent AF) lasts longer than a year.

When someone has AF, the blood in the atria doesn't completely empty into the ventricles which can lead to blood clots in the atria. Many problems can occur if these clots pass into the bloodstream. For example, if a blood clot blocks an artery in the brain, the patient may have a stroke.

2.2.2 Detecting Atrial Fibrillation

When determining if an ECG signal has characteristics of AF, there are a few metrics that we should check for.

- Irregular heart rhythm
- Absence of P-waves
- Irregular or narrow QRS complexes (often less than 120 ms)
- Undulating or completely flat baseline

We can see these metrics in practice by looking at an ECG of a normal heart and one for a patient with AF as shown in Figure 2.8 [8].

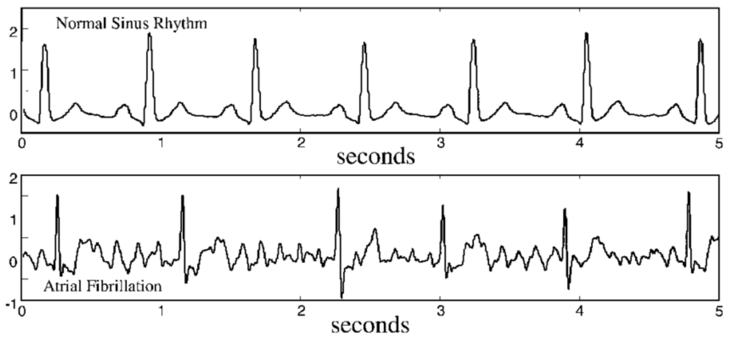


Figure 2.8: Normal ECG vs AF ECG [8]

In Figure 2.9, we can see the irregular heart rhythm that is often associated with AF [8]. The normal heart rhythm shows R-peaks that are evenly spaced, while the irregular heart rhythm shows variation in the length of time between the R-peaks. In the AF rhythm, one heartbeat sometimes takes over a second, and other times it is about 3/4 of a second. The AF signal also does not have any P-waves, and the QRS complexes are narrower than those in the normal signal. The baseline is extremely wavy and erratic in the AF signal while it is relatively flat in the normal signal. All 4 of these biomarkers can be an indication that this patient has AF.

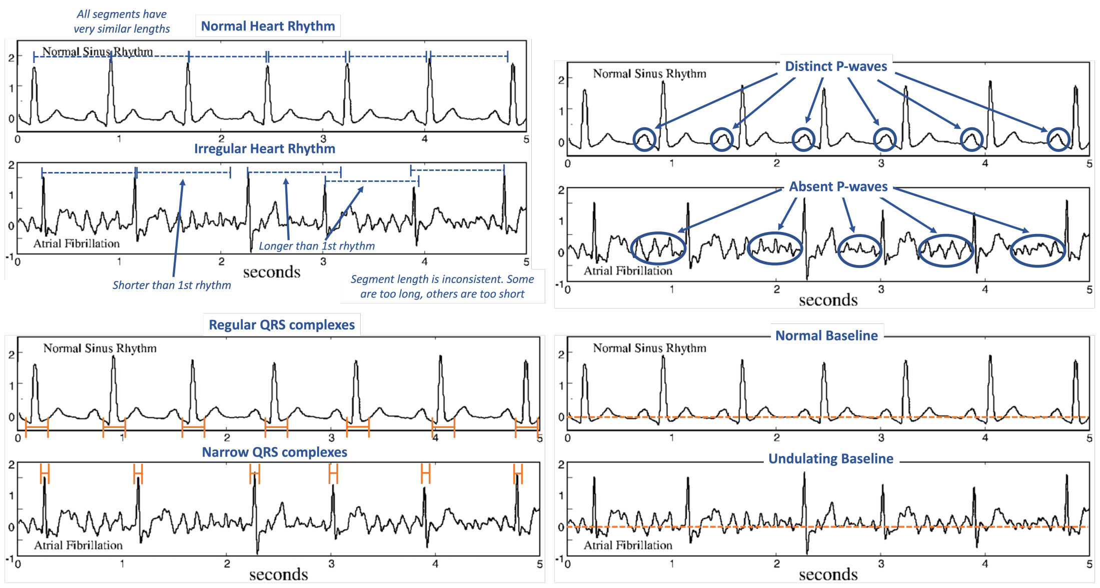


Figure 2.9: Annotated AF Biomarkers [8]

2.3 Secure Multiparty Computation

2.3.1 Basic Background

Most current Artificial Intelligence (AI) and machine learning (ML) applications require data to reside at a centralized location. Storing data at a central node can be problematic. One traditional solution to this problem has been to encrypt the data. This solves the passive eavesdropper problem; however, the third party still can decrypt the data and possibly sell it. Most ML and AI applications also don't work with encrypted data. Since the data needs to be decrypted before it can be used in the algorithms, an active adversary could attack the services running the AI and ML algorithms to obtain the raw user data.

One solution to this problem is to use secure multiparty computation (SMPC). This approach allows a set of parties to jointly perform computations on protected versions of their private data, called shares, without actually revealing the value of their private data. With SMPC, the parties can perform basic mathematical operations like addition, multiplication, and comparison collaboratively. The calculations performed on the data shares carry over to the original data that remain private. This allows the parties to perform collaborative computations without revealing their private data.

SMPC takes a secret value y and generates a number of shares $s_1, s_2, \ldots s_n$ that are distributed among multiple parties $p_1, p_2, \ldots p_n$ such that none of the parties can determine any information about the original secret value y. Only once a certain number of parties contribute their secret shares, can the secret value y be inferred. Note that any computations (e.g. addition of a value) applied to the shares are reflected to y.

2.3.2 Secret Sharing

Given a secret value s, let $[[s]]$ represent the secret shared value of s. There are two main algorithms that are used to share the secret s with n parties:

1. **Secret Share Algorithm:** $ShareSecret(s, n) \rightarrow (s_1, s_2, \ldots, s_n)$. This algorithm takes n parties and a secret input s and then generates a set of n shares $s_1, s_2, \ldots, s_n$.

2. **Secret Reconstruction Algorithm:** $SecretReconstruct(s_1, s_2, \ldots, s_n) \rightarrow \tilde{s}$. This algorithm takes the set of n shares $\{s_1, s_2, \ldots, s_n\}$ and calculates an approximation of the secret input s. The probability that the secret $\tilde{s}$ should be close to the original secret s is close to 1.

2.3.3 Secure Multiparty Addition

If $[[z]]$ and $[[w]]$ are shared, then the parties can obtain the value of $z + w$ by adding/subtracting their corresponding shares: $[[z \pm w]] = [[z]] \pm [[w]]$. The parties do not need to communicate to compute the addition/subtraction of shares. Note that to perform secure subtraction, we used signed addition.

2.3.4 Shamir's Secret Sharing

Shamir's Secret Sharing [21] is a well-known scheme that is often adopted by analogous applications. It uses a **threshold scheme** which is different from simply using additive secret sharing because the secret can be reconstructed by using a subset of the parties instead of requiring all parties for the computation.

A (t, n)-threshold scheme allows you to share a secret among n parties such that any subsets of t parties can reconstruct the secret, but no subset of size smaller than t can reconstruct the secret. This method still uses the secret share and secret reconstruction algorithms mentioned in Section 2.3.2, but it makes some modifications to them.

1. **Secret Share Algorithm:** $ShareSecret(s, n) \rightarrow (s_1, s_2, \ldots, s_n)$

   Each point (x_i, y_i) represents a share for party i. The index x_i lies on the x-axis, and the share y_i is a point on polynomial at the location x_i. Shamir's Secret Sharing Algorithm requires t points to define a polynomial of degree $t-1$. For example, it would require two points to define a line and three points to define a parabola. The share generation is completed as follows:

   - Generate a random polynomial $f(x)$ such that $f(0) = s$ which has the equation: $f(x) = s + r_1 x + r_2 x^2 + \cdots + r_{t-1} x^{t-1} \mod p$, where $r_i (1 \leq i \leq t-1)$ is randomly chosen from an uniform distribution and p is a integer (often prime) larger than (s, n).
   - Generate the secret shares $s_1, s_2, \ldots, s_n$ by evaluating the polynomial $f(x)$ at $x = 1,\ \ x = 2,\ \ \ldots,\ \ x = n$ for each party. Each party $P_i (1 \leq i \leq n)$ receives secret share $s_i = f(x_i)$.

2. **Secret Reconstruction Algorithm:** $SecretReconstruct(s_1, s_2, \ldots, s_n) \rightarrow \tilde{s}$

   We use Lagrange interpolation to retrieve the secret. Lagrange interpolation requires that t values of the polynomial be known to derive the polynomial. We can retrieve the polynomial $f(x)$ as follows:

   $$f(x) = \sum_{i=1}^{t} s_i \cdot \delta_i(x) \mod p \text{ where } \delta_i(x) = \prod_{j=1, j \neq i}^{t} \frac{x-j}{i-j} \tag{2.1}$$

   The formula can be further simplified since we want to evaluate the polynomial at $x = 0$ since $s = f(0)$:

   $$s = f(0) = \sum_{i=1}^{t} s_i \prod_{j=1, j \neq i}^{t} \frac{-j}{i-j} \mod p \tag{2.2}$$

2.3.5 Shamir Secret Sharing Addition Example

Assume that there are three parties Alice, Bob, and Charlie that want to sum their secret values without any of the other participants knowing their secret value. We will assume a *curious-but-honest* adversarial model for this scenario. Each party will follow the steps of the protocols, but given the chance, they will try to infer the secret values of the other parties.

Let's look at a simple scenario where Alice's secret value is $y_a = 1$, Bob's secret value is $y_b = 2$, and Charlie's secret value is $y_c = 3$. Each party needs to generate a private polynomial such that $f(0) = y_a = 1, g(0) = y_b = 2, h(0) = y_c = 3$. Let's take an example where Alice chooses $f(x) = 1 + 2x + 3x^2$, Bob chooses $g(x) = 2 + 4x + 5x^2$, and Charlie chooses $h(x) = 3 + 6x + 7x^2$.

Next, the shares need to be generated and sent between the parties. Each party will receive the shares corresponding to the same x-value in each of the generated polynomials. For example, Alice will keep $f(1)$ and share $f(2)$ with Bob and $f(3)$ with Charlie. Similarly, Bob will keep $g(2)$ and share $g(1)$ with Alice and $g(3)$ with Charlie. Charlie will keep $h(3)$ and share $h(1)$ with Alice and $h(2)$ with Bob.

After the sharing, Alice will have the shares $f(1), g(1), h(1)$, Bob will have $f(2), g(2)$ and $h(2)$, and Charlie will have $f(3), g(3), h(3)$. Note that no two parties can collaborate to derive the generated polynomials f, g, or h. Because the polynomials are second-degree, at least three points on the polynomial must be provided to achieve a successful interpolation. This means that Alice and Bob can not collaborate to determine Charlie's polynomial (and therefore his secret value) or vice versa.

One of the most useful features of secure computation lies in the addition of polynomials. When we add polynomials of the same degree, the resulting polynomial is also of the same degree. In our case, let's assume that $S = f + g + h$. Additionally, we know that for each x the following is valid: $S(x) = f(x) + g(x) + h(x)$. In our scenario, this means that each party by having access to a single point in f, g, h can automatically obtain a single point on the aggregation polynomial S. More specifically, Alice can generate $S(1)$, Bob can generate $S(2)$, and Charlie can generate $S(3)$. Then, all three can meet and exchange their shares. By doing so, they will all have three points in the polynomial S, and using Lagrange interpolation will allow them to infer the polynomial. After the parties have computed S, they can evaluate $S(0)$. That value will allow them to determine the sum of their secrets. Table 2.1 shows the values that Alice, Bob, and Charlie each compute.

Table 2.1: Secure Multiparty Addition: Calculations and Communications

Party	Generated Polynomials	Maintained Share	Received Shares	Sum Point
Alice	$f(x) = 1 + 2x + 3x^2$	$f(1) = 6$	$g(1) = 11, h(1) = 16$	$s(1) = 33$
Bob	$g(x) = 2 + 4x + 5x^2$	$g(2) = 30$	$f(2) = 17, h(2) = 43$	$s(2) = 90$
Charlie	$h(x) = 3 + 6x + 7x^2$	$h(3) = 84$	$f(3) = 34, g(3) = 59$	$s(3) = 177$

2.3.5.1 Lagrange Interpolation

We can use Lagrange Interpolation to determine the value of our $S(x)$ function at $S(0)$. In our example, Alice computed $S(1) = 33$, Bob computed $S(2) = 90$, and Charlie computed $S(3) = 177$. Then, we can define a set of three smaller functions as:

$$\delta_1(x) = \begin{cases} 33, \text{if } x = 1 \\ 0, \text{if } x \neq 1 \end{cases} \tag{2.3}$$

$$\delta_2(x) = \begin{cases} 90, \text{if } x = 2 \\ 0, \text{if } x \neq 2 \end{cases} \tag{2.4}$$

$$\delta_3(x) = \begin{cases} 177, \text{if } x = 3 \\ 0, \text{if } x \neq 3 \end{cases} \tag{2.5}$$

Our summation function S can be defined as $S(x) = \delta_1(x) + \delta_2(x) + \delta_3(x)$. The δ functions we defined can be rewritten as one function in a more compact form as:

$$\delta_i(x) = \begin{cases} 1, \text{if } x = i \\ 0, \text{if } x \neq i \end{cases} \tag{2.6}$$

In that case, the function S can be rewritten as $S(x) = 33\delta_1(x) + 90\delta_2(x) + 177\delta_3(x)$.

Using the $S(x)$ values obtained from Alice, Bob, and Charlie, we can evaluate Equation 2.2

and obtain the sum of their private values as:

$$\begin{aligned} S(0) &= 33 * \frac{(-2)*(-3)}{(1-2)(1-3)} \\ &+ 90 * \frac{(-1)*(-3)}{(2-1)(2-3)} \\ &+ 177 * \frac{(-1)*(-2)}{(3-1)(3-2)} \\ &= 33*3 + 90*(-3) + 177 \\ &= 99 + (-270) + 177 \\ &= 6 \end{aligned}$$

By using secure multiparty addition, Alice, Bob, and Charlie were able to derive the sum of their secret values, 6, without revealing the secret values at $f(0)$, $g(0)$, or $h(0)$. These calculations and communications are based on the BGW protocol[22].

2.4 Application: Secure Euclidean Distance

ML algorithms internally perform recursive rounds of comparisons upon the data. In this context, comparison usually refers to a type of distance calculation typically Euclidean distance. The Euclidean distance measures the *similarity* between two data points in a multidimensional space. Computing the Euclidean distance between two data points allows us to see how close (or similar) the two data points are. Let's examine how we can apply SMPC to compute the Euclidean Distance between two parties in a secure fashion.

2.4.1 Secure Euclidean Distance in Two-Dimensional Space

Given two pairs of data points (A_x, A_y) and (B_x, B_y) in a two-dimensional space, the Euclidean distance between them can be computed with the formula:

$$d_{AB} = \sqrt{(A_x - B_x)^2 + (A_y - B_y)^2}$$

Let's assume that Alice and Bob want to compute the distance between their data points in a two-dimensional Euclidean space. Alice's coordinates are $(1, 2)$ and Bob's are $(4, 6)$. Using the traditional, centralized Euclidean distance calculation, we can see that the distance between Alice and Bob's data points is 5.

$$\begin{aligned} d_{AB} &= \sqrt{(1-4)^2 + (2-6)^2} \\ &= \sqrt{9+16} \\ &= 5 \end{aligned}$$

To compute the distance between Alice and Bob in a secure manner, we are going to use secure signed addition. We will also assume that Alice and Bob are both honest-but-curious participants. They will follow the rules for computing shares in an honest fashion, but they will also try to learn each other's private coordinate values if they have the information necessary to compute them. We will also assume that each party only wants to share one point with the other, so they will need to generate one-degree polynomials.

Shamir Secret Sharing. First, Alice and Bob each need to generate a random polynomial of degree one for each of their secret coordinates. Alice's polynomials will be $f_1(x)$ and $f_2(x)$ where $f_1(0) = 1$ and $f_2(0) = 2$, and Bob's polynomials will be $g_1(x)$ and $g_2(x)$ where $g_1(0) = 4$ and $g_2(0) = 6$.

$$\begin{aligned} f_1(x) &= A_x + r_{1f}x \\ &= 1 + 2x \\ f_2(x) &= A_y + r_{2f}x \\ &= 2 + 3x \\ g_1(x) &= B_x + r_{1g}x \\ &= 4 + 4x \\ g_2(x) &= B_y + r_{2g}x \\ &= 6 + 5x \end{aligned}$$

Alice will keep the shares $f_1(1)$ and $f_2(1)$ and will send the shares $f_1(2)$ and $f_2(2)$ to Bob. Similarly, Bob will keep the shares $g_1(2)$ and $g_2(2)$ and will send the shares $g_1(1)$ and $g_2(1)$ to Alice.

Compute distance between points in each dimension. Next, Alice and Bob want to compute the distance between the shares for their x points and those for their y points. For example, Alice will compute $S_1(1) = f_1(1) + (-g_1(1))$ and $S_2(1) = f_2(1) + (-g_2(1))$. Alice and Bob's computations are shown in Table 2.2.

Table 2.2: Two-Dimensional Euclidean Differences

Party	Input Points	$S_i(n)$ Points
Alice	$f_1(1) = 3, g_1(1) = 8$	$S_1(1) = -5$
	$f_2(1) = 5, g_2(1) = 11$	$S_2(1) = -6$
Bob	$f_1(2) = 5, g_1(2) = 12$	$S_1(2) = -7$
	$f_2(2) = 8, g_2(2) = 16$	$S_2(2) = -8$

Lagrange interpolation. Then, Alice and Bob will share the $S(x)$ points that they calculated with each other, and they will each use Lagrange interpolation to determine $S_1(0)$ and $S_2(0)$ values. These calculations are shown in Table 2.3

Table 2.3: Two-Dimensional Euclidean Lagrange Interpolation Calculations

S_i	Input Points	$S_i(0)$ Points
$S_1(x)$	$S_1(1) = -5$, $S_1(2) = -7$	$S_1(0) = -3$
$S_2(x)$	$S_2(1) = -6$, $S_2(2) = -8$	$S_2(0) = -4$

Then, Alice and Bob will take the sum of the squares of the $S_i(0)$ results from each dimension as follows:

$$\begin{aligned}(d_{AB}(x))^2 &= S_1(0)^2 + S_2(0)^2 \\ &= (-3)^2 + (-4)^2\end{aligned}$$

Note that $(d_{AB}(0))^2$ is the distance squared between the data points A and B. To determine the Euclidean distance between themselves, Alice and Bob just need to take $\sqrt{S_1(0)^2 + S_2(0)^2}$. We can see that the result of computing the Euclidean distance securely using SMPC is the same as the centralized result. A summary of the computations and shares sent and received by each party can be seen in Table 2.4.

Table 2.4: Two-Dimensional Euclidean Distance: Calculations and Communications

Party	Generated Polynomials	Maintained Shares	Received Shares	S(x) Points
Alice	$f_1(x) = 1 + 2x$ $f_2(x) = 2 + 3x$	$f_1(1), f_2(1)$	$g_1(1), g_2(1) \leftarrow Bob$	$S_1(1), S_2(1)$
Bob	$g_1(x) = 4 + 4x$ $g_2(x) = 6 + 5x$	$g_1(2), g_2(2)$	$f_1(2), f_2(2) \leftarrow Alice$	$S_1(2), S_2(2)$

2.4.2 Secure Euclidean Distance in N-Dimensional Space

Given two pairs of data points A and B in an n-dimensional space, the Euclidean distance between them can be computed with the formula:

$$d_{AB} = \sqrt{(A_1 - B_1)^2 + (A_2 - B_2)^2 + \cdots + (A_n - B_n)^2}$$

Let's assume that Alice and Bob want to use 1-degree polynomials, that Alice holds the data point A with coordinates $(A_1, A_2, \ldots, A_n)$, and that Bob holds the data point B with coordinates $(B_1, B_2, \ldots, B_n)$ in an n-dimensional euclidean space.

Alice and Bob can apply a similar procedure as in the two-dimensional example to compute the distance between their data points in n-dimensional space. They will each share a vector of secret coordinates with each other instead of a scalar of secret coordinates.

Shamir Secret Sharing. Alice and Bob will each generate a polynomial function for each of their coordinates such that $f(0)$ for each of the functions is equal to that private coordinate's value.

Alice's vector of $f(x)$ polynomials:

$$(\vec{f}(x))^T = \begin{bmatrix} f_1(x) & f_2(x) & \cdots & f_n(x) \end{bmatrix}$$

$$\vec{f}(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_n(x) \end{bmatrix} = \begin{bmatrix} A_1 + r_{1f_1}x \\ A_2 + r_{2f_2}x \\ \vdots \\ A_n + r_{nf_n}x \end{bmatrix}$$

Bob's vector of $g(x)$ polynomials:

$$(\vec{g}(x))^T = \begin{bmatrix} g_1(x) & g_2(x) & \cdots & g_n(x) \end{bmatrix}$$

$$\vec{g}(x) = \begin{bmatrix} g_1(x) \\ g_2(x) \\ \vdots \\ g_n(x) \end{bmatrix} = \begin{bmatrix} B_1 + r_{1g_1}x \\ B_2 + r_{2g_2}x \\ \vdots \\ B_n + r_{ng_n}x \end{bmatrix}$$

Alice will keep the shares $\vec{f}(1) = \begin{bmatrix} f_1(1) & \cdots & f_n(1) \end{bmatrix}$ and will send the shares $\vec{f}(2) = \begin{bmatrix} f_1(2) & \cdots & f_n(2) \end{bmatrix}$ to Bob. Similarly, Bob will keep the shares $\vec{g}(2) = \begin{bmatrix} g_1(2) & \cdots & g_n(2) \end{bmatrix}$ and will send the shares $\vec{g}(1) = \begin{bmatrix} g_1(1) & \cdots & g_n(1) \end{bmatrix}$ to Alice.

Compute distance between points in each dimension. Next Alice will perform secure signed addition on the set of their shares:

Alice's vector of $S(1)$ polynomials:

$$(\vec{S}(1))^T = \begin{bmatrix} S_1(1) & S_2(1) & \cdots & S_n(1) \end{bmatrix}$$

$$\vec{f}(x) = \begin{bmatrix} S_1(1) \\ S_2(1) \\ \vdots \\ S_n(1) \end{bmatrix} = \begin{bmatrix} f_1(1) + (-g_1(1)) \\ f_2(1) + (-g_2(1)) \\ \vdots \\ f_n(1) + (-g_n(1)) \end{bmatrix}$$

Bob's vector of $S(2)$ polynomials:

$$(\vec{S}(2))^T = \begin{bmatrix} S_1(2) & S_2(2) & \cdots & S_n(2) \end{bmatrix}$$

$$\vec{g}(x) = \begin{bmatrix} S_1(x) \\ S_2(x) \\ \vdots \\ S_n(x) \end{bmatrix} = \begin{bmatrix} f_1(2) + (-g_1(2)) \\ f_2(2) + (-g_2(2)) \\ \vdots \\ f_n(2) + (-g_n(2)) \end{bmatrix}$$

Lagrange interpolations. Next, Alice and Bob will share the $S(x)$ points that they calculated with each other, and they will each use Lagrange interpolation to determine $S_1(0)$ and $S_2(0)$ values. Alice will send $\vec{S}(1) = \begin{bmatrix} S_1(1) & \cdots & S_n(1) \end{bmatrix}$ to Bob, and Bob will send $\vec{S}(2) = \begin{bmatrix} S_1(2) & \cdots & S_n(2) \end{bmatrix}$ to Alice. Then, Alice and Bob will use $\vec{S}(1)$ and $\vec{S}(2)$ to compute $\vec{S}(0)$ with Lagrange interpolation.

Distance Calculation. Finally, they will compute $\sqrt{S_1(0)^2 + S_2(0)^2 \ldots S_n(0)^2}$ to determine their distance in an n-dimensional Euclidean space.

Chapter 3

Proposed Framework

Our study aims to demonstrate the feasibility of our proposed privacy-preserving framework for the detection of AF using smartwatches i.e., a cloud/edge distributed environment. In this chapter, we will discuss critical procedures including the data preprocessing and feature extraction steps that must be taken on the client side. Moreover, we will outline and compare the steps that need to be taken in a traditional, centralized workflow and in our proposed, privacy-preserving version.

3.1 Data Preprocessing & Feature Extraction

Feature extraction is the first process that takes place. The purpose of feature extraction is multifold: (a) in the smartphone realm, one of the most important requirements is to minimize the set of data to be transmitted. Typically the raw ECG signal is composed of hundreds of samples (floating points numerical values); (b) a well-engineered feature extraction step might be beneficial for the predictive accuracy. For example, it may provide robustness against noise and remove biomarkers that are irrelevant to the detection of AF; (c) comparing raw signals is challenging because typically the waves of the ECG signals are not expected to be synchronized. Moreover, users may have different heart rates. Regardless, since the pre-processing is fully done on the client side, it is important to keep the steps as simple and computationally light as possible.

Removing the noise for smartwatch data is especially important because various factors like the wristband's tightness, vertical placement on the arm, and the amount of sweat on the wrist can introduce more noise than in a traditional, 12-lead ECG reading. Some basic noise reduction techniques include removing the baseline wander of the signal, filtering out certain frequency bands, etc. For our frameworks, we elected to adopt the *Finite Impulse Response (FIR)* filter. This is a beneficial option because it allows the client to reduce the noise to an acceptable level without requiring too many computational resources. Based on empirical observations, after this step, the processed records may still contain noise but at a level that may not significantly impact the prediction accuracy.

Typical biomarkers for AF include having a variable RR interval (the distance between R-peaks in a record), an undulating baseline, narrower QRS complexes, and the absence of a P-wave. Since the main indicator of AF is the irregular RR intervals, we focused on that for our feature extraction. First, we identified the locations of all the R-peaks in the record. Then, we calculated all of the RR intervals for the record by taking the time of the first R-peak and subtracting it from the next R-peak's time for all the pairs of R-peaks in the record. After we got the length of all of the RR intervals, we performed some basic statistics on them. The seven features that we choose to perform on the RR intervals were: minimum, maximum, mean, median, standard deviation, skew, and kurtosis.

3.2 Traditional Framework

Before we explain the details of our proposed framework let us start by reviewing some of the important steps in the traditional framework. These are illustrated in Figure 3.1, and the functions used are outlined in Algorithm 1. We will examine these functions and steps in detail in this section.

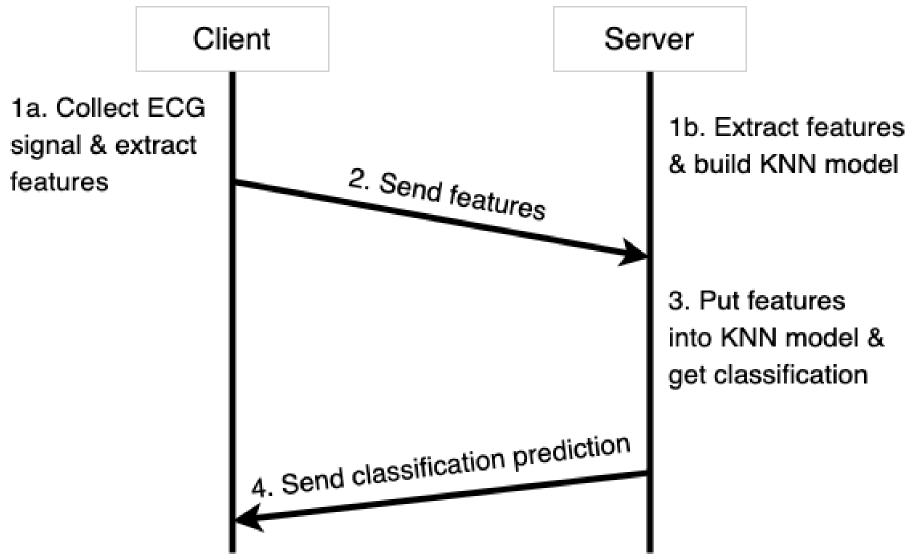

Client
Server
1a. Collect ECG signal & extract features
1b. Extract features & build KNN model
2. Send features
3. Put features into KNN model & get classification
4. Send classification prediction

Figure 3.1: Traditional Framework Steps

Algorithm 1 Traditional Framework

```
1: **function** Client Setup(client ECG signal $ecg_{client}$)
2:     $ecg_p \leftarrow preprocess(ecg_{client})$
3:     $features_{client} \leftarrow extractFeatures(ecg_p)$
4:     **return** $features_{client}$
5: **end function**
6:
7: **function** Server Setup(ECG signals dataset $X$, number of KNN neighbors $k$)
8:     **for** $\forall i \in X$ **do**
9:         $ecg_p \leftarrow preprocess(i)$
10:        $features_i \leftarrow extractfeatures(ecg_p)$
11:    **end for**
12:    $knnModel \leftarrow trainKnn(features, k)$
13: **end function**
14:
15: **function** Server Computation(client features $features_{client}$)
16:     $prediction \leftarrow knnModel(features_{client})$
17:     **return** $prediction$
18: **end function**
```

Step 1: The client collects the ECG record, and then, they perform preprocessing and feature extraction on their ECG signal (1a) as discussed in Section 3.1. At the same time, the server also repeats the same preprocessing steps with its records and then sets up its K-Nearest Neighbors (KNN) model (1b).

Step 2: Then, the client sends their data to the server (2). Since the features can describe a newly obtained ECG signal, they can be used to determine sensitive health data. Thus, it is important that the communication between the client and server is encrypted. This is typically done with TLS and prevents a malicious actor from passively sniffing the network traffic and learning the client's private data.

Step 3: Once the server receives the client's features, it performs an analysis that is based on comparing the record iteratively against a set of records that is already possessed by the server. The aim of this process is to generate a classification prediction $\textcircled{3}$. The prediction distinguishes the signal as normal or AF. There are many options regarding the classification of the records. They can use centralized machine learning algorithms like K-Nearest Neighbors (KNN), decision trees, or unsupervised methods like local outlier factor (LOF) or K-Means clustering. They can also use more sophisticated approaches such as deep learning models like convolutional neural networks (CNNs). For our traditional framework, we decided to adopt the KNN classifier because it is a simple algorithm that gives us an accuracy score sufficient to our needs. Moreover, the particular algorithm belongs to the family of lazy classifiers. Unlike other approaches, lazy classifiers do not construct a model a priori (i.e., during training) time but rather go through this process during inference time. As we will see in the process, this has clear benefits for privacy reasons. *The reader should notice that the choice of the algorithm was not done for optimizations but rather to provide a baseline for evaluation between traditional approaches and our privacy-preserving framework.*

Step 4: After the server determines the record's classification, it sends this prediction back to the client. Since this is also sensitive health data, the communication is again encrypted with TLS. Once the client gets the prediction, it does not need to take any other steps to determine if its record shows signs of AF or not. The reader may notice that, through this scheme, the server becomes fully aware of the ID of the client and the result of the diagnosis.

3.3 Secure Framework

In the secure framework, the client and server use SMPC to keep the client and server's data private. The framework uses secure signed addition to allow the client to perform KNN in a secure manner. For each feature of its ECG signal, the client will have a private function f

such that $f(0)$ is the value of that feature. The server will also have two private functions g and h for each of the features for each of its data points. Since we have only two parties in the proposed scheme, it would be sufficient for the client/server to rely only on two functions. However, for additional complexity and possibly enhanced privacy, we also chose to rely on an additional function h. Notice that in this case, $g(0)$ and $h(0)$ must be chosen carefully so that their sum gives the value of a feature multiplied by -1. For example, if the value of the feature is l and $g(0) = -l$ then $h(0)$ must be chosen to be 0. For reasons of simplicity but without loss of generality in our experiments $h(0) = 0$ always. Recall that SMPC allows us to create a secure function $S(n) = f(n) + g(n) + h(n)$. In our framework, we elect to use signed addition so that $S(n) = f(n) + (-(g(n) + h(n)))$. Moreover, the Euclidean distance between a pair of points A and B in an n-dimensional space is defined as:

$$d_{AB} = \sqrt{(A_1 - B_1)^2 + (A_2 - B_2)^2 + \cdots + (A_n - B_n)^2}$$

In our framework, $(A_1 - B_1)^2$ is equal to $S_1(0)^2$ for the distance of one feature of two records. Also, notice that the exponent operator (multiplication) of d_{AB} raises the degree of the corresponding polynomial. To avoid this additional computation, we chose to compute the result of $(d_{AB})^2$. *Notice that while the difference is computed collaboratively, the exponent operation is done locally in the smartwatch. To avoid additional overhead one option would be to calculate the Manhattan distance instead.*

The steps for our privacy-preserving scheme are shown in Figure 3.2, and the functions used are outlined in Algorithms 2, 3, and 4. We will examine these functions and steps in detail in this section.

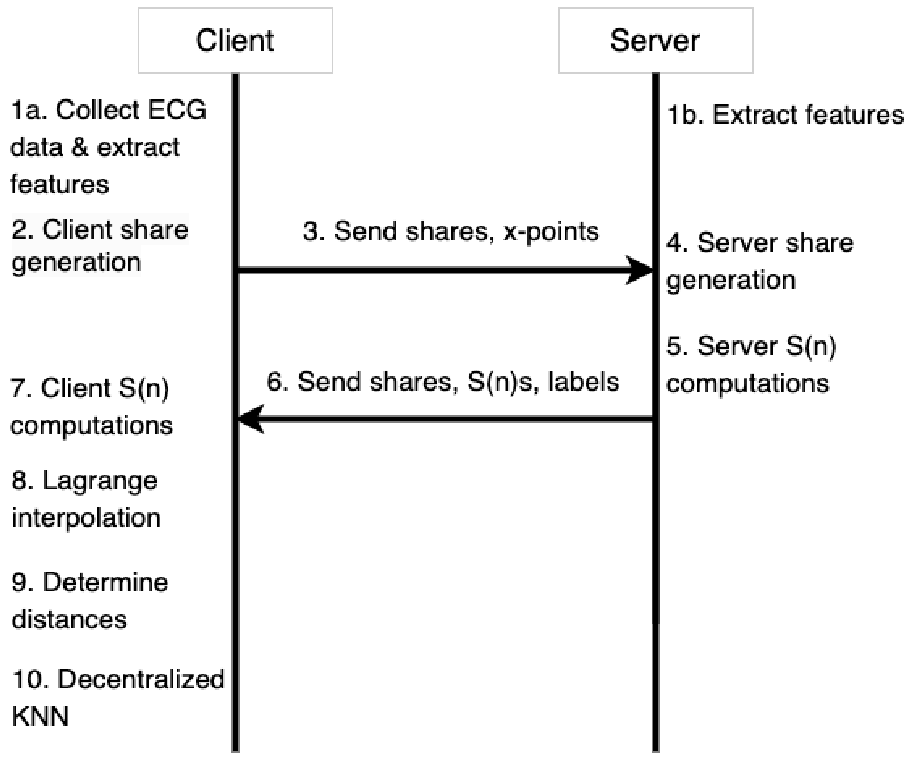

Client
Server
1a. Collect ECG data & extract features
1b. Extract features
2. Client share generation
3. Send shares, x-points
4. Server share generation
5. Server S(n) computations
6. Send shares, S(n)s, labels
7. Client S(n) computations
8. Lagrange interpolation
9. Determine distances
10. Decentralized KNN

Figure 3.2: Secure Framework Steps

Step 1a - Client Preprocessing: The client collects the ECG record, and then, they perform preprocessing and feature extraction on their ECG signal (1a) as discussed in Section 3.1.

Step 1b - Server Preprocessing: The server has its own set of data points that are classified as AF or normal records. It will also preprocess its records and perform feature extraction on them (1b) as discussed in Section 3.1. After extracting the features for an ECG signal, the server will multiply all of the features by -1. This is an important step because we want to securely subtract each server feature from the corresponding client feature. We can achieve this subtraction by using signed addition. The server only needs to perform preprocessing and feature extraction once, but the client will need to do this for every record they want to classify.

Step 2 - Client Share Generation: For each of the 7 features it has, the client generates a random polynomial function $f(n)$ such that $f(0)$ is the value of that feature (2). The polynomial should be second-degree with no 0-value coefficients. Having a second-degree polynomial means that we will need at least three points to determine the polynomial. This number could be further increased to enhance the privacy of the framework, but it would also greatly increase the computation time and complexity at later steps for both the client and the server. We believe that a second-degree polynomial is sufficient to achieve privacy and security. The client chooses three random x-values to use to generate the shares for all of the features by calculating the $f(n)$ values at these points. The x-values need to be between 0 and a supplied maximum x-value.

Step 3 - Client to Server Communication: Next, the client connects to the server to transmit the secret shares to the server. This message will contain a vector of secret shares of features. More specifically, for each feature the second and third $f(n)$, shares will be included. Since we have seven features, this message will contain a total of 14 $f(n)$ shares. It is important to note that the client will *never send the first $f(n)$ share for any feature.* If it were to send all three $f(n)$ shares for a feature, the server could compute the $f(0)$ which is the private feature of the client, and privacy would be compromised. Steps 1a, 2, and 3 are shown in Algorithm 2.

Algorithm 2 Secure Framework - Client Setup & Communication

```
1: function CLIENT PREPROCESSING(ECG Signal $ecg$)
2:     $ecg_p \leftarrow preprocess(ecg)$
3:     $features \leftarrow extractFeatures(ecg_p)$
4: end function
5:
6: function CLIENT SHARE GENERATION(max x-coordinate for shares $xMax$, client features $features$)
7:     $xPoints \leftarrow pickRandom3XPoints(xMax)$
8:     for $\forall i \in features$ do
9:         $f_i \leftarrow generatePolynomial(i)$
10:    end for
11:    $fSharesClient \leftarrow getShares(f, xPoints_1)$
12:    $fSharesServer \leftarrow getShares(f, [xPoints_2, xPoints_3])$
13: end function
14:
15: function SEND DATA TO SERVER(client's public shares $fSharesServer$, client's chosen x-values $xPoints$)
16:     $sendToServer(xPoints, fSharesServer)$
17: end function
```

Step 4 - Server Share Generations: Once the server receives the client's message, the server generates a polynomial function $g(n)$ and $h(n)$ for each feature of each signal in its dataset such that $g(0)$ is the value of -1 multiplied by that feature and $h(0) = 0$ $\textcircled{4}$. Similar to the client, these functions will be random, second-degree polynomials that are free of coefficients of 0. It is important that the server will generate a different set of polynomials for each client request because it prevents clients from collaborating and determining the server's raw features. Then, for each feature in each signal, the server will use the list of x-values received from the client to calculate the $g(n)$ and $h(n)$ shares at these points. For example, if the client used x-values 4, 7, and 21, the server will compute $g(4), g(7), g(21), h(4), h(7)$, and $h(21)$ for each of its features.

Step 5 - Server $S(n)$ Computations Next, the server will calculate the $S(n)$ values for the second and third shares for each feature for each record compared to the client's record feature shares $\textcircled{5}$. For example, if the second x-value is 7, the server will compute $S(7) = f(7) + g(7) + h(7)$ for each feature in each record.

Step 6 - Server to Client Communication: Then, the server will send a message to the client $\textcircled{6}$. This message will contain a vector that contains, for each record, a set of the $S(n)$ values computed in Step 5, along with the first $g(n)$ and $h(n)$ shares for each feature, and the class of that record. Steps 1b, 4, 5, and 6 are shown in Algorithm 3.

Algorithm 3 Server Setup & Computations

```
1: function SERVER PREPROCESSING(set of ECG Signals $ecgs$)
2:     for $\forall i \in ecgs$ do
3:         $ecg_p \leftarrow preprocess(i)$
4:         $features_i \leftarrow extractFeatures(ecg_p)$
5:         $records_i \leftarrow makeNegative(features_i)$
6:     end for
7: end function
8:
9: function SERVER SHARE GENERATION(ECG signals features dataset $records$, set of x-values from client $xPoints$)
10:     for $\forall record \in records$ do
11:         $g \leftarrow generatePolynomials(record)$
12:         $h \leftarrow generatePolynomialsWith0Features()$
13:         $gSharesAllRecords_{record} \leftarrow getShares(g, xPoints)$
14:         $hSharesAllRecords_{record} \leftarrow getShares(h, xPoints)$
15:     end for
16: end function
17:
18: function SERVER COMPUTE POINTS ON S(N)(received client shares $fServer$, set of g(n) shares for all records $gSharesAllRecords$, set of h(n) shares for all records $hSharesAllRecords$)
19:     for $\forall record \in gSharesAllRecords$ do
20:         $gServer \leftarrow [x[:1] \forall x \in gSharesAllRecords_i]$
21:         $hServer \leftarrow [x[:1] \forall x \in hSharesAllRecords_i]$
22:         $sSharesServerAllRecords_record \leftarrow sumSharesOneRecord(fServer, gServer, hServer)$
23:     end for
24: end function
25:
26: function SEND DATA TO CLIENT(Client $client$, set of g(n) shares for all records $gSharesAllRecords$, set of h(n) shares for all records $hSharesAllRecords$, set of s(n) shares for all records $sSharesServerAllRecords$, classifications for all records $labels$)
27:     $gSharesClient \leftarrow [x[0] \forall x \in gSharesAllRecords_i]$
28:     $hSharesClient \leftarrow [x[0] \forall x \in hSharesAllRecords_i]$
29:     $client \leftarrow (gSharesClient, hSharesClient, sSharesServerAllRecords, labels)$
30: end function
```

Step 7 - Client $S_i(n)$ Computation: When the client receives the message from the server, it use the first share from its $f(n)$ functions along with the first shares from each $g(n)$ and $h(n)$ function to compute the final point on the $S_i(n)$ functions $\textcircled{7}$.

Step 8 - Client Lagrange Interpolation: Now that the client is in possession of three shares (one calculated, two received) on each $S_i(n)$ (where i is the number of features), it can use these points to derive each of the $S_i(n)$. Then, the client will calculate $S_i(0)$ for each $S_i(n)$. The result gives the client the distance between each of its features and the features of each of the server's records. A small $S_i(0)$ value implies that that feature is very similar to the client's record and server's record.

Step 9 - Client Determine Distances: To determine the distance between the client point A and a server point B, we use the following equation:

$$d_{AB} = \sqrt{S_1(0)^2 + S_2(0)^2 + \cdots + S_n(0)^2}$$

where n is the number of features. By taking the square root of the sum of the $S_i(0)$ values for each record obtained in Step 8, we get the Euclidean distance between the client and each of the server's records.

Step 10 - Client Decentralized KNN: Now that the client has the Euclidean distances, the client orders the distances from smallest to largest. Since we want to use KNN for our classification model, we will take the k shortest distances from the list where k is the number of neighbors. Then, the client determines what classification is the most common for the top k points and uses this for its prediction of the client's ECG signal classification. Steps 7, 8, 9, and 10 are shown in Algorithm 4

Algorithm 4 Client Computations

```
1: function CLIENT COMPUTE POINT ON S(N)(g(n) shares from server $gSharesClient$, h(n) shares from server $hSharesClient$, client's f(n) shares $fClient$)
2:     for $\forall record \in gSharesAllRecords$ do
3:         $gClient \leftarrow [x[:1] \forall x \in gSharesAllRecords_i]$
4:         $hClient \leftarrow [x[:1] \forall x \in hSharesAllRecords_i]$
5:         $sSharesClientAllRecords_record \leftarrow sumSharesOneRecord(fClient, gClient, hClient)$
6:     end for
7: end function
8:
9: function CLIENT LAGRANGE INTERPOLATION(s(n) shares from server $sSharesServerAllRecords$, s(n) shares from client $sSharesClientAllRecords$)
10:    for $\forall record \in sSharesClientAllRecords$ do
11:        $sServer \leftarrow sSharesServerAllRecords_i$
12:        $sClient \leftarrow sSharesClientAllRecords_i$
13:        for $\forall i \in sSharesServerAllRecords_i$ do
14:            $s0_i \leftarrow lagrangeInterpolation(sServer_i, sClient_i)$
15:            $s0_iSquares \leftarrow s0_i^2$
16:        end for
17:        $distances_i \leftarrow \sqrt{sum(s0_iSquares)}$
18:    end for
19: end function
20:
21: function CLIENT DECENTRALIZED KNN(number of KNN neighbors $k$, client's calculated distances $distances$)
22:    $sortedDistances \leftarrow sortSmallestToLargest(distances)$
23:    $shortestKDistances \leftarrow topDistances(sortedDistances, k)$
24:    $prediction \leftarrow findMostCommonLabel(sortedDistances, labels)$
25: end function
```

Chapter 4

Experimental Evaluation

We conducted a series of experiments (four in total) to evaluate the efficiency of our framework. The main goal of these experiments was to compare the efficiency of the secure method to the traditional, centralized method. More specifically, we wanted to quantify the penalties in terms of the accuracy and time needed to complete the computations.

Hereunder, we shall explain the structure of the data used for our evaluations, and the specific parameters of important procedures such as preprocessing and feature extraction we performed on the dataset. This chapter concludes with the results of each of the experiments we conducted.

4.1 Dataset

Today, several publicly available datasets exist that aim to facilitate research regarding the automated detection of heart conditions. Table 4.1 outlines the structure of some of the most popular datasets that have been used for the detection of various heart conditions.

Table 4.1: AF classification datasets

Dataset	Leads	Records	Normal	AF	Sample Length	Classes
PhysioNet/ Computing in Cardiology Challenge 2017 [9, 14]	1	12,186	5224	805	9-61 sec	4
PTB-XL [9, 11, 12]	12	18,869	N/A	N/A	10 sec	71
St. Petersburg INCART 12-lead Arrhythmia Database [9]	12	75	0	3	30 mins	10
Brno University of Technology ECG Quality Database (BUT QDB) [9, 13]	1	18	N/A	N/A	24+ hours	3

The reader should recall that our experiments are geared towards the smartwatch realm. Smartwatches possess only one lead, and the corresponding readings are expected to be highly noisy due to possible misplacement of the smartwatch sensor on the user's wrist or due to spontaneous movement. Several of the existing datasets are deemed inappropriate for our purposes because they were obtained using the standard 12-lead configuration instead of with a single lead. Moreover, the corresponding ECG data records were often collected in hospitals by trained professionals in near-ideal conditions. Hence, the data is not expected to be corrupted by significant levels of noise. Additionally, many datasets contain a variety of different arrhythmia classifications but no normal records to compare the abnormal ones to or only have samples for a small number of patients with very long sample times. Below we shall discuss the most common datasets that have been considered in the bibliography for research in the field of automated heart disease detection.

4.1.1 PTB-XL Database

In the PTB-XL dataset [9, 11, 12], there are no normal or AF classes. The dataset is divided into 5 superclasses which combined contain 71 subclasses. Although there are no classes for normal records or AF records, there are diagnosis notes in a separate file that may be used to indicate patients with normal heart rates and those with AF. The number of patients used for this dataset is large, and each record is 10 seconds long. The readings were taken with a clinical 12-lead ECG, thus the data is relatively clean. An example record from the dataset as obtained with the LightWAVE tool [9], [10] is shown in Figure 4.1.

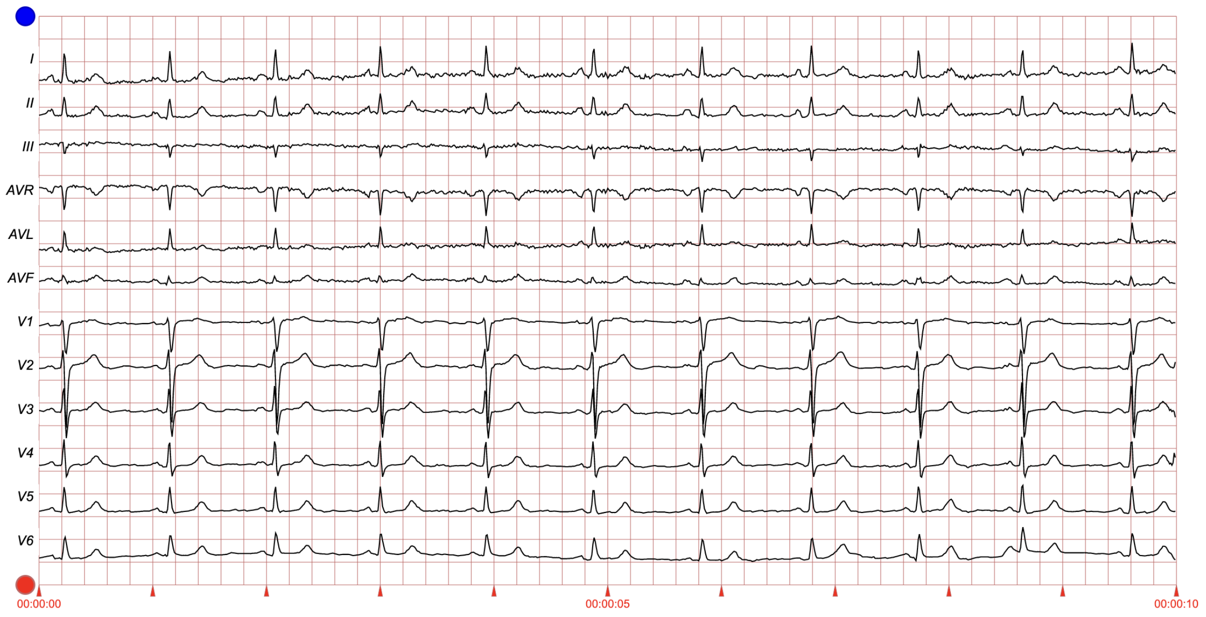


Figure 4.1: Example record from the PTB-XL dataset as obtained with the LightWAVE tool [9], [10], [11], [12]

4.1.2 St. Petersburg INCART 12-lead Arrhythmia Database

In the St. Petersburg dataset [9], there are no patients with a normal diagnosis and only 3 with AF. There are only 75 records, and the records are 30 minutes long. An example record for this dataset is shown in Figure 4.2

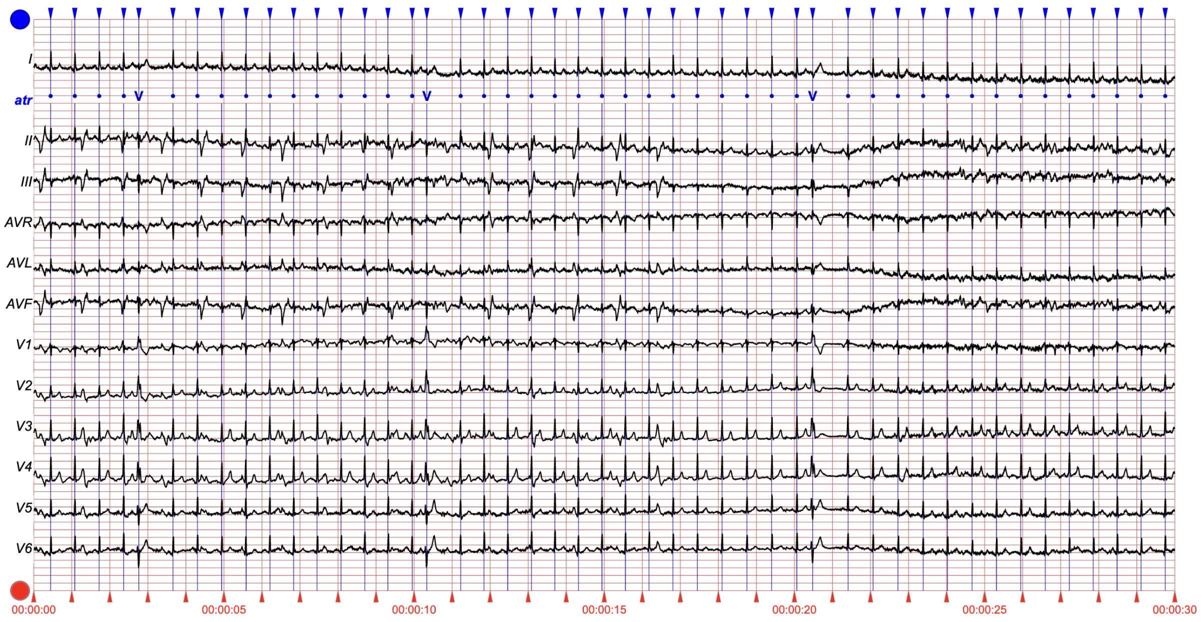


Figure 4.2: Example record from the St. Petersburg dataset as obtained with the LightWAVE tool [9], [10]

4.1.3 Brno University of Technology ECG Quality Database (BUT QDB)

The BUT QDB dataset [9, 13] was another potential option. The data was collected with a single lead which is beneficial for our purposes, but it does not have normal or AF classes. Instead, the three classes in the dataset distinguished ECG signals of different noise levels. Additionally, the shortest record is 24 hours long, and there are only 18 records. An example record from the dataset can be seen in Figure 4.3.

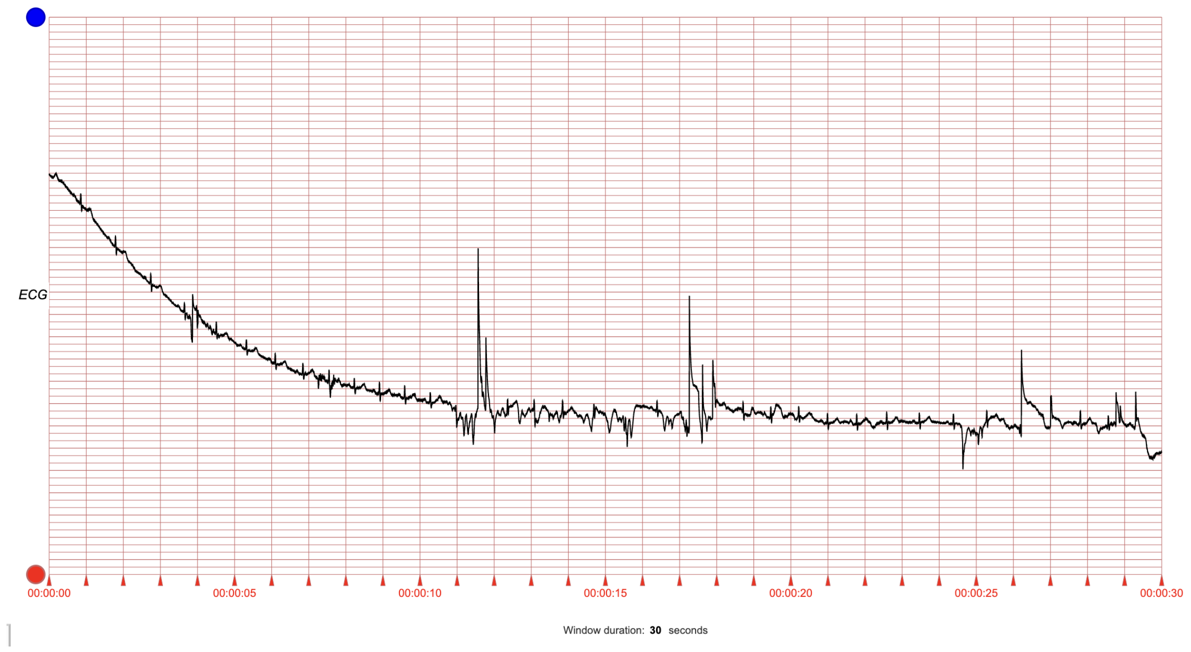


Figure 4.3: 30 seconds of an example record from the BUT QDB dataset as obtained with the LightWAVE tool [9], [10], [13]

4.1.4 PhysioNet/Computing in Cardiology Challenge 2017 Database

The PhysioNet/Computing in Cardiology Challenge 2017 [9, 14] dataset was captured with a single lead and contains a larger number of classified records for both normal and AF patients. Moreover, it has a short sample length. Two example records of the dataset shown originally in [9, 14] are shown in Figure 4.4.

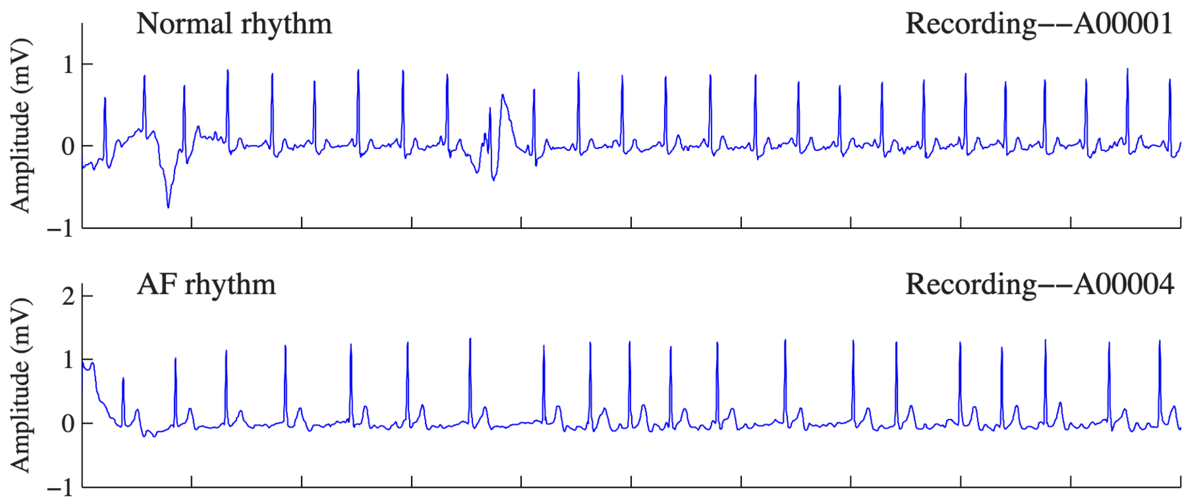


Figure 4.4: Example of a normal and AF record from the PhysioNet/Computing in Cardiology Challenge 2017 dataset [14]

In an attempt to mimic the readings obtained by a smartwatch, one approach would be to capitalize on the pluralism of ECG examples contained in the 12-lead datasets. In this case, we should isolate the readings of the single lead of interest from the 12-lead configuration and then add artificial noise. However, in our experiments, we chose to rely on a dataset that naturally bears the highest similarity to smartwatch readings namely, the PhysioNet/Computing in Cardiology Challenge 2017 dataset published by PhysioNet [9, 14]. The dataset is composed of 8,528 training records and 3,658 testing records. The data was captured using AliveCor's single-channel ECG device. Each user held one of the two electrodes in each hand which created a lead I ECG reading [14]. Although the device was not worn on the user's wrist as a smartwatch would be, it was still beneficial for us to use this dataset because of the realistic noise in it. Each record was classified into one of 4 categories: normal, AF, other rhythm, and noisy. Each record was sampled at 300 Hz and between 9 and 61 seconds with most records being just over 30 seconds long. Since our main goal is the detection of AF condition only, we isolated only the normal (5224) and AF (805) records from the original dataset. As we will see in Section 4.7, for the purposes of our evaluation, this dataset was further split into training and testing sets according to a specific evaluation strategy e.g., 10-fold cross-validation.

4.2 Experimental Setup

All the experiments were done on an HP Pavilion laptop with an 11th Gen Intel(R) Core(TM) i7-1165G7 processor and 12 GB of RAM. Our experiments were all coded using Python. For both the traditional and secure models, the client and server were run locally on the machine by running two separate processes. We exchanged data between the client and server by using Python's socket class.

To perform the preprocessing, we relied on Python's *BioSPPY* library [23]. Specifically, we used their *filter_signal* function with the parameters ftype=FIR, band=bandpass, order=90, frequency=[3,45], and sampling_rate=300. To detect the R-peaks, we used BioSPPY's *ecg* function, using the filtered signal and a sample rate of 300. For feature extraction, we used numpy to determine the minimum, maximum, mean, median, and standard deviation of the RR intervals and scipy to determine their skew and kurtosis. To perform 10-fold cross-validation, we used sklearn's StratifiedKFold class with the parameters n_splits=10, shuffle=True, and random_state=86.

4.2.1 Traditional Model

For the traditional model, we relied on sklearn's KNeighborsClassifier model with the parameters n_neighbors=9, p=2, weights=uniform, and algorithm=auto. We created a client class that would take a feature vector and sent it to the server class. The server put the feature vector into its pre-trained KNN model. Then, the server sent the classification prediction back to the client.

4.2.2 Secure Model

For the secure model, the client and server both had a component that allowed them to supply a feature vector and get a set of randomly generated, zero-free coefficient polynomials such that $f(0)$ for each polynomial was the feature's value. The client would then select three x-values between 0 and a maximum x-value of 100,000 for each feature. These points were used to get the three shares to be used in Lagrange interpolation. Then, the client used its TCP socket to send a message that contained the three x-values and for each feature, the second and third shares. The message was formatted as a JSON array.

When the server received the message, it generated polynomials and shares for each of its data points in the same manner as the client such that $g(0)$ for each polynomial was the feature's value. For each data point, it also computed an $h(n)$ polynomial such that $h(0) = 0$. Then, the server computed $S(n) = f(n) + g(n) + h(n)$ for the second and third shares.

Then, the server sends a message that contains, for each record, the first $g(n)$ and $h(n)$ shares for each of its features, the two $S(n)$ points that it calculated, and the classification label for that record.

Upon receiving the server's message, the client uses the server's share to calculate, for each feature for each record, the last point on $S(n)$. Then, for each feature for each data point, the client uses scipy's "lagrange" function with three x-values that it selected and the three $S(n)$ points to determine $S(n)$ and then $S(0)$. Now the client has an $S(0)$ value for each of the features for each of the server's records, it can take the squared sum of the $S(0)$ values for each feature to determine the distance to each record. Once the client knows the distance from itself to all of the server's data points, it orders these distances from smallest to largest. Then it takes the $k = 9$ smallest distances and their corresponding classification classes. Then, the client takes the class that appears the most in $k = 9$ list and uses that class as its classification prediction.

4.3 Data Preprocessing

From visual inspection of several example signals in the dataset, we identified that several issues exist in the ECG readings that may lead to sub-optimal results. For example, we observed that there are some signals that contain portions that are highly noisy. For this reason, we relied on Python's *BioSPPY* module [23] and adopted the FIR filter with a band-pass filter of 3 to 45 to apply noise reduction. In the past, similar approaches have been adopted by other researchers [24] with the same aim. Figure 4.5 shows an example of a noisy record before and after being passed through the filter.

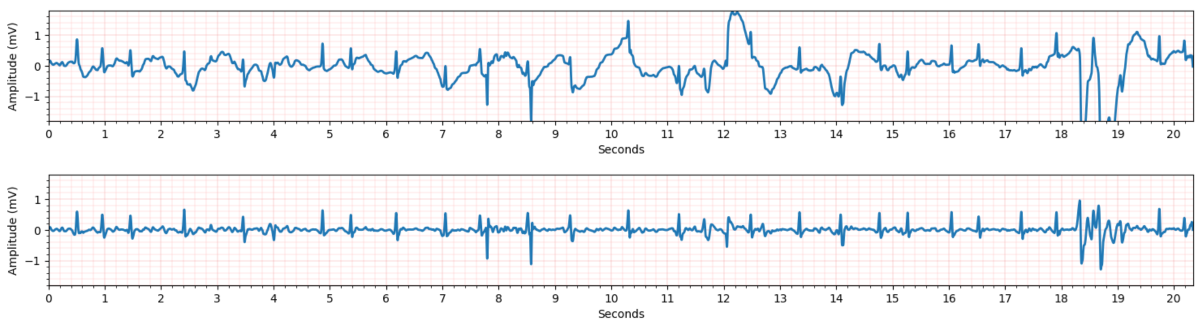


Figure 4.5: ECG record A00015 before (top) and after (bottom) applying band-pass filtering

4.4 Feature Extraction

A significant number of computations are expected to take place inside the limited resources of the clients i.e., the smartwatches. For this reason, we engaged in feature engineering primarily in an attempt to reduce the total number of features thus, significantly reducing the computational overhead and the size of messages to be sent over the network. It is well-known that not all raw features are equally important for the detection of certain conditions. For example, the work of Goodfellow et al. [24] showed that heart rate is one of the strongest biomarkers used to detect AF. The heart rate can be extracted by calculating the time intervals between consecutive R-peaks, called the RR interval.

In our implementation, we relied on Python's BioSPPY library [23] to calculate the R-peaks' locations similar to Goodfellow et al. [24]. Each patient's ECG signal was further simplified to a set of the following seven features for the RR intervals for the sample: minimum, maximum, mean, median, standard deviation, skew, and kurtosis. In all subsequent sections, when we refer to the patient's data, we are referring to this feature vector.

Figure 4.6 shows a normal and AF pre-processed record and the location of the R-peaks that were identified for them, and Table 4.2 shows the feature vectors that were calculated using the RR intervals for each record. The RR interval lengths for both records are also shown in the histogram in Figure 4.7. Note that all the calculations were done relative to our sample rate of 300 Hz. So a distance of 50 between two R-peaks would be equivalent to 10 seconds.

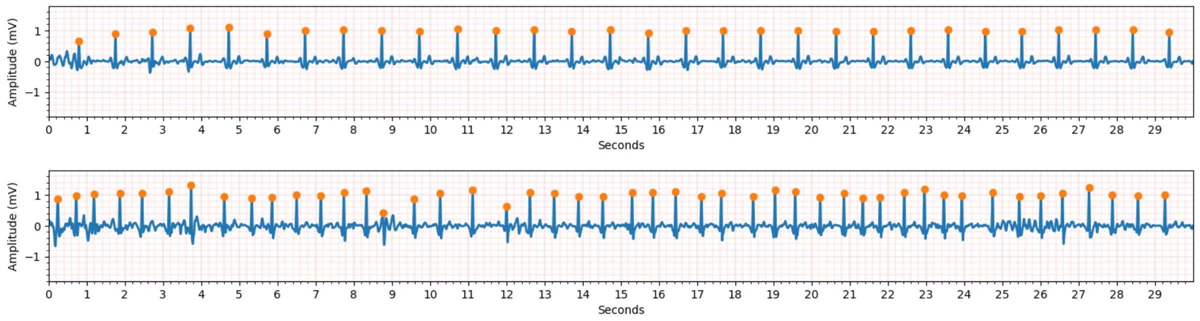


Figure 4.6: Pre-processed normal ECG record A00025 (top) and AF record A00009 (bottom) with R-peaks marked

Table 4.2: Example RR interval feature vectors for normal record (A00025) and AF record (A00009)

Feature	Normal record	AF record
Minimum distance	59.4	67.4
Maximum distance	62.7	140
Average distance	60.9	98.2
Median Distance	60.8	95.5
Standard deviation of distances	1.03	17.6
Skew of distance	0.282	0.318
Kurtosis of distances	-1.05	-0.395

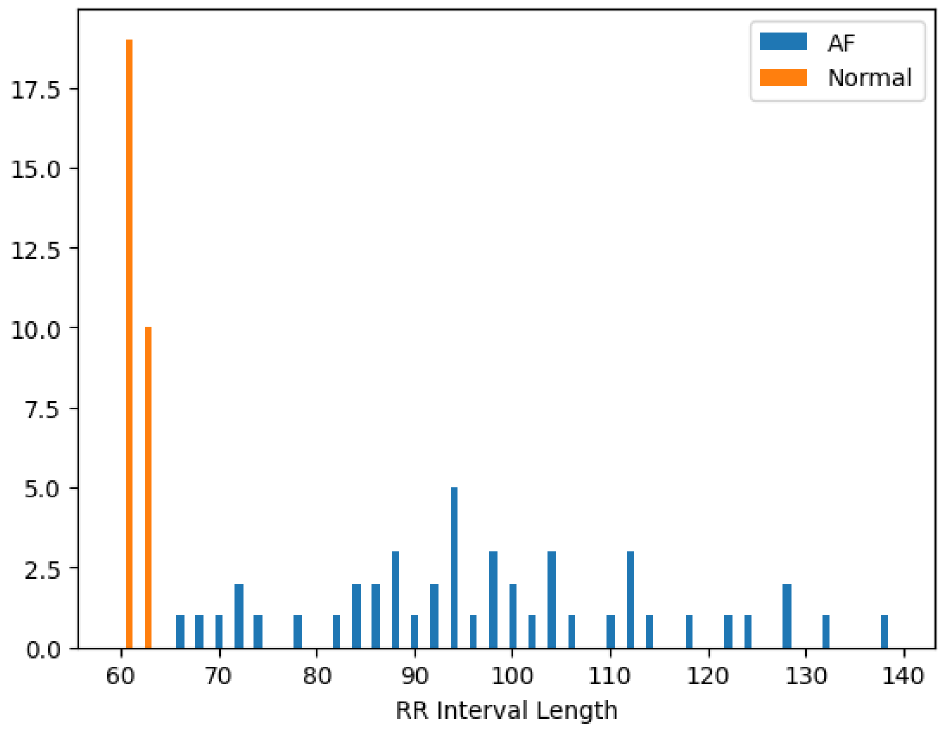


Figure 4.7: Histogram of RR intervals for a normal (A00025) and AF (A00009) record

4.5 Experiment 1: Comparing Distance Calculations

An integral component of the adopted classification algorithms is the calculation of Euclidean distance. Due to Lagrange interpolation which lies at the core of the adopted secret sharing scheme, all mathematical operations including the calculation of Euclidean distance, may not be precise. This in turn may cause severe degradation of the classification accuracy of the model.

Assumption: *The calculation of the Euclidean distance in our adopted secure scheme, may be imprecise, which may lead to the degradation of predictive accuracy.*

Our first experiment aims to quantify the difference between the Euclidean distance obtained through the traditional approach versus the one calculated by our adopted distributed, secure scheme.

Towards this end, we compared (a) each normal record to the rest of the normal records, (b) each normal record to each of the abnormal (AF) records, (c) each AF record to the rest of the AF records, and finally (d) each AF record to each normal record, utilizing both the traditional and the secure schemes to calculate the Euclidean distances.

For each point in the first set, the distance from that point to each of the other points in the second set was computed. Then, the distances for that point were ordered from smallest to largest, and we took the average of the shortest 9 distances for each point, according to equation 4.1.

$$\hat{d}(x_j) = \frac{\sum_{i=1}^{k} d_i}{k} \tag{4.1}$$

where $k = 9$ is the number of neighbors, D is the sorted list of distances between signal x_j and each other signal in the dataset X, and $d_i \in D$ are distance of x_j with x_i, $\forall i \neq j$.

We plotted the average distance for each of the points on a box plot to visually infer their distribution and compare the distances of points across different classes. The results are given in Figures 4.8 and 4.9. The reader may notice that the distances computed via the traditional and secure schemes are almost identical. Table 4.3 shows the mean, maximum, minimum, and median of the absolute differences between the traditionally and securely calculated distances.

Table 4.3: Absolute differences in Euclidean distance calculation between traditional and secure computations

Comparison Set	Mean	Max	Min	Median
N-N	5.89×10^{-5}	3.17×10^{-4}	8.23×10^{-9}	5.02×10^{-5}
N-AF	5.57×10^{-5}	2.71×10^{-4}	9.45×10^{-9}	4.71×10^{-5}
AF-AF	6.10×10^{-5}	2.57×10^{-4}	5.13×10^{-7}	5.09×10^{-5}
AF-N	6.33×10^{-5}	2.51×10^{-4}	1.04×10^{-7}	5.50×10^{-5}

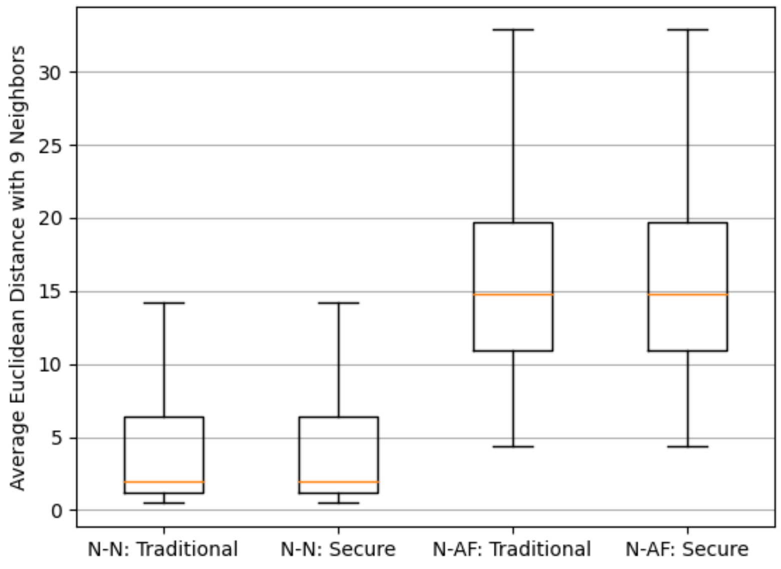


Figure 4.8: Distances between (a) normal vs normal and (b) normal vs AF data points (signals) for the traditional and secure scheme.

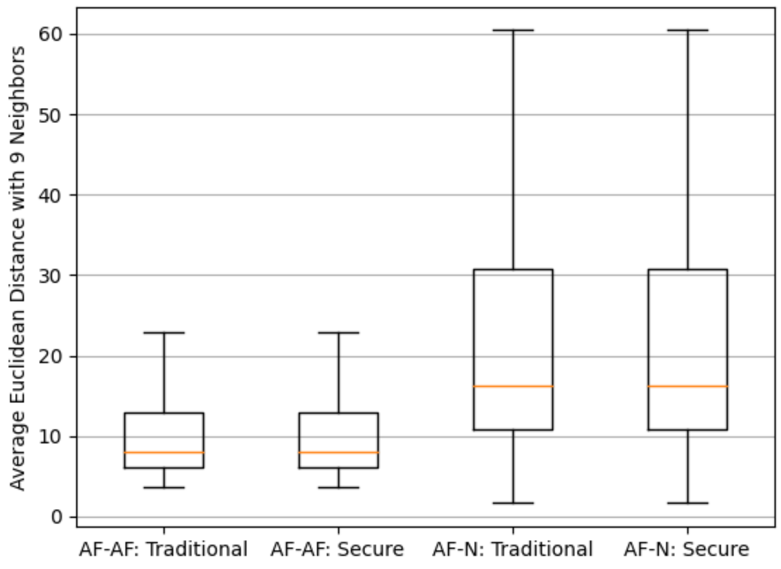


Figure 4.9: Distances between (a) AF vs AF and (b) AF vs normal data points (signals) for the traditional and secure scheme.

A secondary conclusion of this experiment is that on average, and with the exception of some outliers, there is a clear separation between the normal and AF classes. This implies that the classification accuracy is expected to be high on average.

To further quantify the difference between the distances obtained through the traditional and secure schemes we calculated a second metric. More specifically, for the two schemes, we computed the Euclidean distance from each point to all different points in the dataset and calculated the absolute value of the difference between the two. Then, we took the differences and normalized them for better visualization. Our results show that the difference between the two schemes is constantly kept at an extremely low level. The average normalized difference was 1.07%, and the maximum normalized difference was 5.01%. Our results for just one example are given in Figure 4.10.

Conclusion: *The calculation of Euclidean distances using our proposed distributed and privacy-preserving scheme yields values that are nearly identical to the traditional centralized approach for the same data. On average, the difference is restricted to 1.07% and never exceeds 5.01%.*

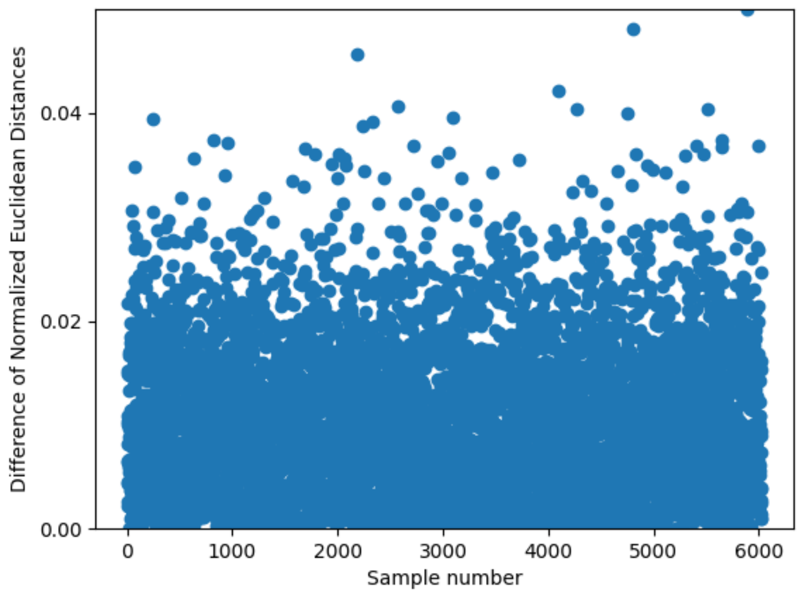


Figure 4.10: Absolute difference of normalized Euclidean distance of an example signal in the dataset with all the rest of the examples in the same dataset.

4.6 Experiment 2: Comparing Maximum X-coordinates in Lagrange Interpolation

With Lagrange Interpolation, there can be some error in calculating the secret polynomial S(n) which may be carried over to the calculation of the secret values S(0). The reader may recall that in this context, S(0) is the distance between the two points we are comparing.

Assumption: *The range of the possible x-values used for Lagrange interpolation can result in accuracy errors when calculating the distributed, secure Euclidian distance.*

Having more potential different x-coordinates that could be used to create the shares makes it harder for an attacker to guess what points are being sent and for them to try to determine what components are going into the Lagrange function, so we want to measure the trade-off in accuracy that we get from using a broader range of possible x-values.

In our experiment, the client had the data for one record, and the server had the data for all the other records. We allowed the client to pick 3 random x-values between 0 and a maximum x-coordinate. Then, the client and server exchanged shares so that the client could compute the S(n) function and the S(0) value for each pair of points. This tells us the distances between the client's point and all the points that the server has. We completed this step multiple times, increasing the maximum x-coordinate for each trial.

Next, we computed the distances using the traditional Euclidean distance formula. The traditional distances served as our ground truth, and ideally, the S(0) value for each point of a trial should match the ground truth. To compare the securely-computed distances to the traditional, centrally-computed distances for each trial, we calculated the sum of the squared distance differences in each trial. Our results are shown in Figure 4.11.

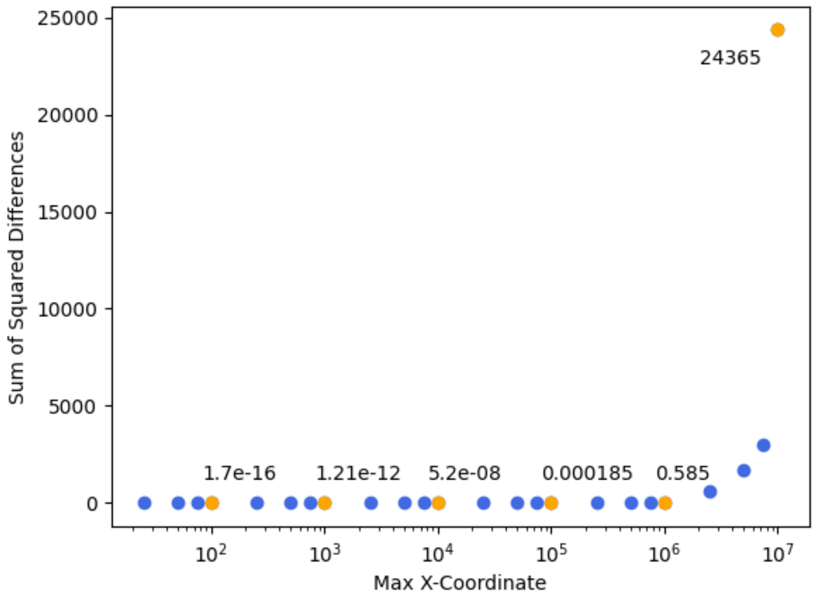


Figure 4.11: Sum squared difference between traditional and secure distance calculation for a point as the maximum x-value for Lagrange interpolation increases.

In Figure 4.11, we can see that as the maximum x-value used for Lagrange goes up, the difference in the distances between the traditional and secure models increases which is detrimental to the calculation of accuracy. This happens because the Lagrange interpolation function sometimes may make some small errors. When the x-coordinates are closer to S(0), the small errors don't make much of a difference in computing S(0), but when x=1,000,000, having some small errors in the function estimate can make a big difference at S(0). Based on empirical observations, for our application, a value ranging between 0 and 10^5 had almost no difference. At 10^6, the sum of the squared distances was only 0.585. Since the sum is squared and it is for over 6,000 differences, 0.585 is an acceptable error. Between 10^6 and 10^7, we start to see a significant increase in the error. At 10^7, the sum of squared distances is 24,365. An error rate this large can result in inaccurate distance computations. As a result, we elected to use a maximum x-value of 10^6 for our experiments.

Conclusion: *Increasing the maximum x-value used in Lagrange interpolation can decrease the accuracy of the Euclidean distance calculation. There are very minimal differences between the* *traditional and secure calculations up to an x-value of 100,000. Beyond that, the error rate of the Euclidean distance starts to increase exponentially.*

4.7 Experiment 3: Comparing Accuracy with KNN Classification

In the next experiment, we wanted to compare the accuracy and F1 scores for AF classification in the traditional environment compared to the secure setup that uses SMPC.

Assumption: *The calculation of the Euclidean distance in our adopted secure scheme may be imprecise which may lead to a lower accuracy for the secure KNN model compared to the traditional KNN model.*

For our experiment, we used 10-fold cross-validation upon the features extracted from the dataset. We used KNN with the parameter k=9 neighbors. In the secure model, we set the maximum x-coordinate for the Lagrange interpolation to 100,000.

To compare the models, we looks at the confusion matrices for both models and used them to calculate each model's accuracy, precision, recall, and f1-scores according to equations 4.2, 4.3, 4.4, and 4.5 where TP is the true positives, TN is the true negatives, FP is the false positives, and FN is the false negatives.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.2}$$

$$Precision = \frac{TP}{TP + FP} \tag{4.3}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.4}$$

$$F1Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{4.5}$$

Ideally, a classifier should maximize the TP and TN and minimize the FP and FN. Since our dataset is imbalanced and there are significantly more normal records than AF records, it's important to look at metrics like the recall and F1 score instead of simply looking at the model's accuracy.

The confusion matrices for the traditional and secure models are shown in Figures 4.12, 4.13. The matrices are identical. Both models correctly classify 5130 normal records, misclassify 94 normal records, correctly classify 623 AF records, and misclassify 182 AF records. They both had a recall value of 77.39% and an F1-score of 81.87%. A full summary of the metrics for both models can be seen in Table 4.4.

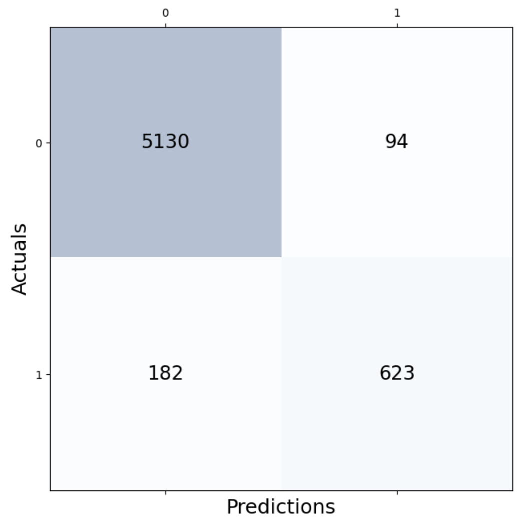


Figure 4.12: Confusion Matrix - Traditional Model

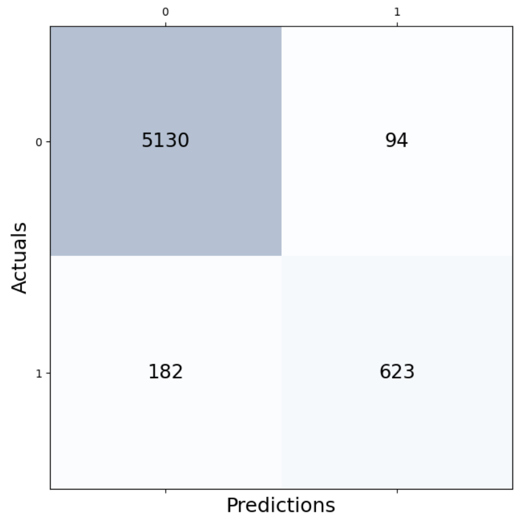


Figure 4.13: Confusion Matrix - Secure Model

Table 4.4: Comparing Statistics for KNN Models in the Traditional and Secure Approaches

Model	F1 Score	Recall	Precision	Accuracy	TP	FP	TN	FN
Traditional	81.87%	77.39%	86.89%	95.42%	5130	94	623	182
Secure	81.87%	77.39%	86.89%	95.42%	5130	94	623	182

Conclusion: *Using the secure, decentralized approach resulted in no penalty in our accuracy, F1-score, precision, or recall metrics for our KNN model.*

4.8 Experiment 4: Comparing Classification Time Efficient

In the final experiment, we wanted to compare the amount of time needed to run an AF classification in the traditional environment compared to the secure setup that uses SMPC.

Assumption: *The calculation of the Euclidean distance and the use of Lagrange interpolation in our adopted secure scheme are more computationally heavy and may lead to a longer AF classification time than the secure KNN model compared to the traditional KNN model. The amount of data to be sent over the network is also larger which could result in more networking time for the secure model.*

For our experiment, we gave the client one record from the dataset and gave the server the full dataset. Then, the client submitted the record to the server for classification. We completed this trial 100 times for both the traditional and secure models and took the average times for all of the steps in the computations and communications. Our timing results are shown in Table 4.5, and the size of the messages passed between the client and server for both scenarios are shown in Table 4.6.

Table 4.5: Average time (in ms) for each networking and computation operation over 100 trials

	Traditional	Secure
Client Setup	0.783	0.706
Client to Server Networking	0.092	0.300
Server Computation	1.01	1618
Server to Client Networking	0.038	706
Client Computation	0	14,667
Total Time	1.92	16,992

Table 4.6: Average message sizes (in bytes) over 100 trials

	Traditional	Secure
Client to Server	150	343
Server to Client	17	3,153,403

Table 4.5 shows that the traditional model is significantly faster than the secure model. The client setup is a little slower because the secure client needs to generate a polynomial and shares for each of its features.

The client-to-server networking is slower because the client's message is larger in the secure setup. The client needs to send two shares for each feature and the set of 3 x-values for the secure model, while in the traditional model, it just needs to send the 7 features. For server computation, the traditional approach is faster because the server just needs to plug the client's features into the KNN model that it has already trained. In the secure setup, the server needs to generate two polynomials and two shares on each polynomial for each of the features for each of the data points that it has. Then, it computes two of the points on the secret polynomial S(n) for each of its features compared with the client's features. Sending the server-to-client message takes more time because the server needs to send a message that contains the S(n) points that it calculated, 2 shares on g(n), and two shares on h(n) for each of the features for each of its records. The server has the whole data set (6,029 records), so the resulting message is large.

The bulk of the timing difference happens in the client computation section. In the traditional model, the client does not need to perform any computations. In the secure framework, the client needs to calculate the final S(n) point for each f(n) along with the pair of g(n) and h(n) shares received from the server. Then, it needs to use the two S(n) points that it received plus the additional S(n) point that it calculated to perform Lagrange interpolation for each feature for each data record. Then, the client needs to take the square root of the sum of the S(0) values it calculates. This will give the client the distance between itself and one of the server's data points. After completing this for all points, the client then needs to order these distances and find the most common class in the top k records to determine what class its own record is.

Conclusion: *Using the secure, decentralized approach resulted in a time penalty when compared to the traditional approach. The size of the packets sent was also larger.*

Chapter 5

Discussion

One challenge that we faced was choosing the optimal preprocessing procedures for noise reduction. Through analysis of the datasets, we observed that the data collected from the single lead can be at times highly noisy. Notice, this is expected to happen in realistic scenarios since the data in our dataset (much like in real life) was not collected in a hospital or other controlled environment. We wanted to choose a preprocessing option that was not too computationally heavy since the client does the signal preprocessing. Using the BioSPPY Python module [23] sufficiently removed noise in our dataset without being too computationally heavy.

Another challenge we faced was choosing the best features for feature extraction. Since the client needs to perform Lagrange interpolation for each feature, it is important to keep the number of features small. Since there are numerous biomarkers other than the RR interval, we considered extracting other parts of the QRS complex and examining them. Options of biomarkers to look for included the presence of P-waves and the length of the QRS complex itself. Although these options could have increased our accuracy, we decided to focus only on the RR interval features because they were simple for the client to extract. The added increase in accuracy from adding more features would have also increased the size of the packets sent over the network and the time needed to complete a prediction.

Chapter 6

Related Work

In this chapter, we will look at the research done with AF detection using 12-lead and single-lead configurations. Then, we will examine different privacy-preserving methodologies, specifically homomorphic encryption, differential privacy, federated learning, and secret sharing.

6.1 Atrial Fibrillation Detection

6.1.1 Traditional, 12-lead ECG detection

One commonly referenced standard for 12-lead ECG AF detection, called the SAFE trial, was developed by Hobbs et al. in 2005 [25, 26]. They specifically were concerned with AF detection in patients aged 65 years and older, and the objective of the study was to compare different active AF screening intervention practices to a control group where no active screening was done. The two active screening methods they examined were systematic and opportunistic screening. In systematic screening, the entire target population was invited to do an ECG. In opportunistic screening, a healthcare professional takes the patient's pulse during a consultation, and if the pulse is irregular, the patient is invited to do an ECG. In the control group, the detection rate of new cases of AF was 1.63% a year, and with the intervention practices, it was 1.63%. The systematic screening detected 1.62%, and opportunistic screening detected 1.64%. One of the main takeaways from the study is that actively screening for AF results in more AF detections than the current practices.

Another paper, published by Poon et al. in 2005 [27], examined the accuracy of the rhythm interpretations from the GE Healthcare Technologies MUSE software 005C. They compared the computer-based interpretation with physician-confirmed classifications for 4297 records taken in a university hospital setting. They found that 13.2% (565/4297) of the computer-based diagnoses needed revision; however, if patients with pacemakers were excluded, the revision rate dropped to 7.8% (307/3954). The predictions yield a high false-positive identification rate for the detection of various non-normal sinus rhythms which resulted in a specificity of 90.1% for normal sinus rhythms. They also found that the false negative rate for AF was 9.2%. The study concludes that *computer diagnosis of paced rhythms remains problematic* and that physicians may need to read over the computer-generated diagnoses to verify them.

In 2017, Acharya et al. [28] developed a 9-layer deep convolutional neural network (CNN) to classify heartbeats into one of the five following categories: non-ectopic, supraventricular ectopic, ventricular ectopic, fusion, and unknown. For their dataset, they used MIT's BIH Arrhythmia Database [9, 29]. The dataset is collected using a two-channel ambulatory ECG, but the study focuses only on the readings from Lead II. They start by pre-processing the ECG signal to remove noise by using denoising and removing the baseline wander. Then, they perform ECG heartbeat segmentation and R-peak detection. Their model achieved 94.03% classification accuracy when noise removal was applied and 93.47% accuracy without noise removal.

6.1.2 Smartwatch Detection

In 2019, Perez et al. [17] conducted a large-scale study to test the ability of smartwatches to identify AF. This work is often referred to as the *Apple Heart Study.* The study's rationale and methods are illustrated in [16]. In the study, participants without AF used a smartwatch to record their heart rates. An individual's heart rate was recorded using photoplethysmographic (PPG) technology on smartwatches and gives a reading that is similar to a single-lead ECG. If the smartwatch recorded irregular pulses that could be AF, the participant was mailed a single-lead ambulatory ECG patch that they wore for up to 7 days. The patch was used to test if the patient actually suffered from AF. The study had 419,297 participants. 2,161 participants received an irregular pulse notification. 84% of these notifications were concordant with AF. Some patients were excluded from subsequent testing because did not schedule an appointment to pick up their ECG patch, they had previous AF, or they had urgent symptoms. 658 of the 2,161 participants were mailed an ECG patch for further testing. 450 of the 658 returned their testing patch and recorded data that could be analyzed. 34% of the ECG patch reading showed AF. Overall, participants had a low probability of receiving an irregular pulse notification, and the algorithm was accurate in identifying AF-like rhythms. Of the participants who were recorded as having an irregular pulse, the positive predictive value (PPV) was 84% for observing *AF on the ECG simultaneously with a subsequent irregular pulse notification* and 71% for observing *AF on the ECG simultaneously with a subsequent irregular tachogram.*

In another study from 2019, Guo et al. [30] also investigated using mobile PPG technology for AF detection. In the study, 187,912 individuals in China recorded their heart rates using wristbands or wristwatches. 424 participants received "suspected AF" notifications. These notifications were based on algorithms implemented in the wristwatches and wristbands which were both made by Huawei Technologies Co. They found that 262 of the 424 flagged individuals effectively followed up and completed subsequent testing. 227 of the 262 (87%) were confirmed as having AF, with the positive predictive value of PPG signals being 91.6%. Some critiques of the study relate to concerns regarding the accuracy of the technique used to detect AF episodes [31]. Guo et al. assumed that the suspected AF episodes were either all true or all false based on the results of the final diagnosis at the follow-up appointment. In the critique, the authors stress that all AF events count and that "electrocardiographic reference should be available when adjudicating AF episodes as being true" [31]. It is possible that the patient's smartwatch recorded a set of AF symptoms, even if the final diagnosis contradicted the findings. Since the study did not compare the PPG readings with an ECG reading, it is more difficult to verify the PPV. When compared to the Apple Heart Study's PPV of 71% [17], the PPV of 91.6% obtained by Guo et al. [30] seems like a major improvement, but it is important to consider the possible overestimation of their PPV.

Because of the amount of noise in smartwatch data, preprocessing can be an important step. To preprocess smartwatch data, there are numerous different options. As stated in the survey done by Liu et al. [32], numerous works apply FIR band-pass filters [24, 33, 34], IIR high-pass filters [35], other frequency filters [36, 37], and median filters to remove the record's baseline wander [38]. Most of these works used the PhysioNet/ Computing in Cardiology Challenge

2017 Dataset [9, 14].

After preprocessing the records, many works extracted features from the records. As illustrated in the Lie et al. survey [32], many works extract basic statistical characteristics based on RR interval, RR interval first differences, or second derivatives of RR interval [39, 40, 41, 42, 43, 44, 45, 46, 47]. Other works calculated statistics based on the heart rate variability (HRV) in the record [24, 42, 43, 46, 48, 49, 50, 51]. Some works also extracted features related to the beat waveforms [41, 44, 45, 47, 48, 52]. Additionally, many works analyzed different segments of the signal's morphology [24, 33, 41, 43, 44, 46, 52, 53, 54, 55, 56] such as the PR interval, AT interval, and many other morphology features.

We can also compare the different classifiers used for smartwatch AF detection. One commonly adopted approach was a Support Vector Machine (SVM) [35, 39, 40, 53, 57, 58, 59, 60]. Other works used decision trees like the random forest classifier [41, 52, 61, 62], decision trees with Adaboost [44, 46, 55, 63], and variations of bagged decision trees [45, 64, 65]. Some works used deep learning for classification. CNNs were used in [38, 48, 66, 67, 68, 69, 70, 71, 72, 73], and long short-term memory (LSTM) classifiers were used in [33, 74, 75].

Most of the models were tested with the data in the PhysioNet/ Computing in Cardiology Challenge 2017 Dataset [9, 14]. Some of the works using CNN models [66, 69, 70, 71, 72, 73] used other databases such as the MIT-BIH Arrhythmia Database [9, 76] and the MIT-BIH AF Database [9, 77].

Of the databases that used the PhysioNet/ Computing in Cardiology Challenge 2017 database, the highest overall F1-score with an SVM model was 81% [40, 57]. For random forest classifiers, decision trees with Adaboost, and bagged decision trees, the highest overall F1-scores were 83% [52], 86% [44], 81% [64], respectively. The highest overall F1-score for CNNs was 82% [67], and for LSTMs, it was 82% [75]

6.2 Privacy-Preserving Methodologies

In this section, we will examine different privacy-preserving methodologies that can be used to perform secure mathematical operations on data points without revealing the values of the data.

6.2.1 Homomorphic Encryption

One privacy-preserving method is homomorphic encryption (HE) which allows us to encrypt data points and perform operations on the data without decrypting them. The term homomorphism was introduced in 1978 by Rivest, et. al [78]. The authors lay out an example scenario where a bank wants to keep their customers' data encrypted but also be able to determine statistics about the group like the average account balance or the number of loans given that are over $5,000. They want the bank to be able to determine aggregate statistics on their database without compromising their customer's privacy.

For an encryption scheme to be homomorphic over an operation "$\star$", the following equation must be true:

$$E(m_1) \star E(m_2) = E(m_1 \star m_2) \forall m_1, m_2 \in M \tag{6.1}$$

where E is the encryption algorithm, and M is the set of all possible messages [79]. There are three main types of homomorphic encryption: partially homomorphic encryption (PHE), somewhat homomorphic encryption (SHE), and fully homomorphic encryption (FHE). In PHE, we can perform one type of operation an unlimited amount of times. In SHE, more than one type of operation can be performed a limited number of times. In FHE, an unlimited number of operations can be performed an unlimited number of times. Because FHE can be resource heavy, it is still common to use PHE and SHE schemes.

One of the first applications of PHE, RSA, was introduced in 1978 by Rivest et al [80]. The purpose of RSA is to generate public-private key pairings for public key cryptography. RSA relies on the difficulty of factoring the product of two large numbers. In their work [78], Rivest et al. demonstrate that RSA is homomorphic over multiplication. This means that $E(m_1) * E(m_2) = E(m_1 * m_2)$. In 1985, another multiplicative PHE scheme was introduced by ElGamal [81]. This scheme was also used for public key encryption and was an implementation of the Diffie-Hellman Key Exchange system [82]. ElGamal relies on the difficulty of computing discrete logarithms over finite fields for its security. In 1999, Paillier introduced a PHE scheme that implemented additive HE [83]. His model is based on composite degree residuosity classes and trapdoor mechanisms. For example, a class may have a degree set to a hard-to-factor number $n = pq$ where p and q are two large prime numbers

The Polly Cracker scheme was developed by Fellows and Koblitz in 1994 [84] and is one of the first examples of SHE. The scheme allows for an unlimited number of both additions and multiplications, but the ciphertext grows exponentially as more operations are performed. Additionally, the multiplication step is very expensive. In 2011, Albrecht et al. [85] introduced a Polly Cracker with Noise cryptosystem. Their goal was to improve upon the work of Fellows and Koblitz and to reduce the size of the cipher text. With the new approach, additions do not increase the size of the ciphertext, and multiplications square the size of the ciphertext. Another SHE scheme was introduced in 2005 by Boneh-Goh-Nissim (BGN) [86]. This scheme allows an arbitrary number of additions and one multiplication upon the ciphertext. They improve upon previous schemes because the ciphertext size in their approach is independent of the formula size or depth. This means that the ciphertext size will not increase as more operations are performed.

In 2009, Gentry [87] made a breakthrough in HE and developed the first feasible FHE system. FHE allows an arbitrary number of addition *and* multiplication over the encrypted data. He achieved this by starting with a somewhat homomorphic "bootstrappable" encryption scheme that works when the function f is the scheme's own decryption function. Then, they use recursive self-embedding to reach an FHE scheme. Their approach is based on performing hard problems using ideally chosen lattices. This approach is mostly theoretical and is inefficient to implement in practice. The bootstrapping part of their algorithm is especially computationally heavy.

In 2012, Brakerski et al. [88] proposed a new FHE scheme based on Gentry's work [87] that eliminates the bootstrapping procedure and is more efficient. The authors develop schemes based on learning with error (LWE) and ring-LWE (RLWE). For the LWE and RLWE scenarios, they provide two schemes, one without bootstrapping that evaluates L-level arithmetic circuits and whose security is based on RLWE for an approximation factor exponential in L and another that uses bootstrapping as an optimization whose security is based on the hardness of RLWE for quasi-polynomial factors.

In 2013, Rane and Boufounos [89] developed a privacy-preserving nearest neighbors (PPNN) scheme. They used this method to compare a set of signals to a query signal to determine which signals are more similar to the query signal. They divide their approach into two problems (a) doing a privacy-preserving distance computation and (b) doing a privacy-preserving minimum distance finding, and they develop different privacy-preserving. In one of their scenarios, they perform distance computations using an additive homomorphic system. The minimum distance finding is performed using a garbled circuit which is an application of SMPC. They based their garbled circuit on the secure millionaire problem posed by Yao [90] and discussed in Section 6.2.4. To find the minimum distance, they do a sequence of millionaire protocols. This scenario is based on the "unproven hardness of factorization, finding residues, discrete logarithms, [and], lattice problems". In their information-theoretic scenario, they use polynomial secret sharing to calculate the Euclidean distance and the same minimum distance finding using garbled circuits as in the first approach. This approach is based on the principle that an adversary will not know a secret key or shared secret. In their approach that used HE, they found that "if the ciphertext size and the complexity of the encryption and decryption operations can be made manageable", it is "feasible for the client to encrypt its data itself and send it to a cloud-based server" [89].

In 2015, Page et al. [91] proposed a cloud-based privacy-preserving remote ECG monitoring and surveillance system that used FHE. Because FHE is computationally heavy, they propose a proof-of-concept system to determine the practicality of using FHE. The authors use a prolonged QT interval as a biomarker for increased patient risk. They transfer FHE-encrypted QT and RR sampled to a server. The server then performs computations on the data samples and sends the encrypted results to the patient's doctor. The patient's doctor has the decryption key and uses it to determine the results of the computation. The transmission of the encrypted AT and RR samples took about 2 Mbps of network bandwidth per patient, and the authors found that running comparisons on the encrypted data was fast enough to be run on modest software and send alerts to the patient's doctor in real-time.

In 2016, Chillotti et al. [92] proposed a new, faster FHE scheme that performs the bootstrapping step in less than 0.1 seconds . The work improves upon the work of Ducas and Micciancio [93] who achieved a bootstrapping procedure runtime of about 0.69 seconds. Additionally, Chillotti et al. reduced the bootstrapping key size from 1 GB to 24 MB [92]. The new approach provided the same security levels as previous approaches.

In 2019, Lui et al. [94] developed a privacy-preserving classification scheme for cloud data. Their approach uses additive HE and secret sharing to complete a secure squared Euclidean distance, secure comparison, secure sorting, secure minimum and maximum number finding, and secure frequency calculations. These building blocks allow them to perform KNN securely. This approach is different than other approaches like SMPC because the data nodes say offline. The only interaction between a querying node and the server is that the node will send encrypted data and receive the encrypted results. They analyze their approach by using the University of California, Irvine's Car Evaluation Data Set. A similar concept was attempted in 2014 by Samanthula et al. where the authors also use an additive HE scheme to perform privacy-preserving KNN [95]. Compared to the work of Lui et al [94], the work of Samanthula et al. [95] is more efficient while still ensuring the same level of user privacy.

In 2020, Yang et al. [96] developed a secure and efficient KNN classification algorithm (SEED-KNN) for Industrial IoT. Their approach uses vector homomorphic encryption (VHE). They construct a key-switching matrix and use a noise matrix for data encryption. Their model is ideal for performing operations on encrypted data in large-scale scenarios on distributed servers. One application of this is in industrial control systems [96]. They demonstrate that their model has a high classification accuracy, achieves semantic security, and is very efficient. Because of these strengths, it can be used in industrial IoT.

In 2021, Vizitiu et al. [97] proposed a framework for privacy-preserving wearable health data analysis to detect AF using HE. Their study develops a privacy-preserving, ML framework, develops CipherML which is a library to help implement ML solutions on homomorphically encrypted data, and does a proof-of-concept study of AF detection from wearable devices. For the AF detection, they propose two approaches (a) a multi-layer perceptron (MLP) that receives ECG features that are computed and encrypted on the wearable device and (b) an end-to-end deep convolutional neural network (1D-CNN) that receives the entire raw ECG data encrypted. Their method with the MLP using features from the ECG achieves a higher F1 score than the 1D-CNN approach. Using HE results in a small performance drop with both approaches. This decrease in performance is caused by the limitations of using homomorphically encrypted data versus using plaintext data.

In 2021, Almalki et al. [98] developed an Efficient and Privacy-Preserving Data Aggregation

(EPPDA) scheme with authentication for IoT-based healthcare applications. In their setup, a user has one or more sensors and IoT-enabled medical devices. They want to automatically exchange data to identify the presence of different health conditions and then send these results to healthcare professionals for review. Their approach uses additive HE for data privacy and a homomorphic MAC to check data integrity. In their model, the data aggregator verifies the legitimacy of the nodes in the network when it receives a message. The nodes also verify the server before they accept the aggregation results. With their approach, the authors are able to guarantee data privacy, message authenticity, and integrity. Their communication overhead is also lightweight.

In 2022, Watkins et al. [99] proposed a privacy-preserving scheme to aggregate data in E-Health. Their approach uses additive homomorphic encryption to perform KNN using data from IoT wireless wearable devices. In their model, they have a health center that distributes encryption keys to hospitals. The hospitals use these keys to encrypt their patient's data. Then, the hospital sends the encrypted data to a third-party server. The health center can send aggregation queries to the server, get the encrypted results, and then decrypt using their key. The aggregation queries that the health center sends to the server are related to the distances between a test record and the data that the server has. The authors found that using KNN with FHE made their algorithm efficient and scalable and allowed them to preserve patient data. The data they used was 400 bytes before encryption and 1600 bytes after encryption.

One of the drawbacks of homomorphic encryption is that it can be difficult to apply with large datasets because of the mathematically heavy homomorphic operations. Additionally, although FHE allows more computations, it can sometimes take more time than approaches that use PHE or SHE. Using HE can also become slower as the size of the dataset increases.

6.2.2 Differential Privacy

The concept of differential privacy was first proposed by Dwork in 2006 [100]. With this methodology, each client adds random noise to their data to preserve their privacy. The noise is generated in such a way that attributes about each client's record cannot be determined. Additionally, differential privacy attempts to ensure that adding or removing a record to the database does not significantly change the outcome of the analysis or computations performed.

An attacker cannot determine the characteristics of a record based on the change in the analysis result when the record is added to the database. Given two databases, D_1 and D_2 that differ by only one row, their analysis outputs should be probabilistically indistinguishable as shown in Equation 6.2

$$\Pr[\mathcal{K}(D_1) \in S] \leq \exp(\epsilon) \times \Pr[\mathcal{K}(D_2) \in S] \tag{6.2}$$

where $\mathcal{K}$ is the noise randomization function, ϵ is the differential privacy leakage parameter, and all $S \subseteq Range(\mathcal{K})$. The closeness of the output of both databases is determined based on the privacy parameter ϵ. Lower values of ϵ indicate stronger privacy and closer analysis results for the two databases.

Setting an appropriate value for ϵ can be difficult. Lee and Clifton found that the value of ϵ does not correlate directly to a specific privacy value [101]. A lower value of ϵ indicates high privacy, but the value of ϵ needed to achieve a specific level of privacy will vary based on the dataset that is being used. An ϵ value of 0.1 may preserve privacy for one dataset, but a different dataset may need an ϵ values of 0.01 to preserve privacy. The authors found that when discussing using a differential privacy mechanism, it is also important to find the appropriate value of ϵ and that determining a correct ϵ value may be as hard or harder than implementing the differential privacy.

In 2019, Beaulieu-Jones et al. used differential privacy to train deep neural networks to be used in clinical data sharing [102]. Their findings suggest that synthetic data with noise added to it can be shared with other parties and allow them to perform data analyses as if they had the original patient data. In their approach, they used an auxiliary classifier generative adversarial network (AC-GAN) to determine a patient's systolic blood pressure over time. Their approach works best with low-dimensional time series data. They also compared training different machine learning models with the real data compared to the synthetic data. In all cases, the models trained on the synthetic data performed worse than with the real data. This occurs because of the trade-off between privacy and accuracy in differential privacy scenarios.

Another option with differential privacy is to use local differential privacy (LDP). In the work of Bebensee [103], the author provides an overview of how to apply LDP. Unlike traditional differential privacy, LDP allows noise to be added to user inputs locally instead of potentially relying on a third party. This approach improves data privacy, but the total variance is higher and increases as the number of participants increases. One example of using LDP in smartwatches is demonstrated by Kim et al. [104]. In this approach, LDP was used to calculate population statistics about the step count of users. The paper evaluated different values of ϵ and compared the results to the aggregation result without applying differential privacy. With a low value for ϵ, the actual and estimated results were significantly different. As the value of ϵ was increased, the users compromised some privacy, and the accuracy increased.

Ghazarian [105] applied differential privacy methods to ECG data to determine heart conditions in 2021. In the work, he trains a CNN on samples from about 81,000 patients. The overall accuracy of the model was 95.69%, but the classification rate for patients with AF and complete right bundle branch block was 49%.

6.2.3 Federated Learning

Another method for preserving private data is federated learning. This approach allows nodes to work collaboratively to create a shared, global prediction model. At the same time, the nodes keep all the data that they contribute to training the model on their own device. Federated learning allows for the development of a machine learning model without compromising nodes' private data.

The concept of federated learning was developed in 2016 by Google researchers [106]. The global, server model gets sent the all of the client nodes. Then, the client nodes use their private data to continue training the model. The client nodes then send this updated model back to the server where all of the model updates are aggregated into the global prediction model. The updated global model is then sent to the client nodes again. This process occurs in an iterative fashion so that the global model is frequently updated without accessing the client nodes' private data. One of the drawbacks of federated learning is that the information communicated from the client to the centralized party may possibly reveal sensitive, private information about the client [106]. This means that the client must still place some trust in the server that is coordinating all of the training [107]. One weakness of this approach is that it favors clients who hold a larger portion of the data.

In 2018, Brisimi et al. [108] proposed the use of federated learning predictive models with Electronic Health Records. Their goal was to perform binary supervised classification to predict hospitalizations for cardiac events. For the classification, they used a soft-margin l_1-regularized sparse Support Vector Machine (sSVM) classifier. They found that using cluster Primal Dual Splitting (cPDS) in their sSVM model was slightly more accurate and faster than other methods but required more network overhead. They also determined that their model improved prediction accuracy over existing risk metrics.

In 2021, Can et al. [109] applied federated learning to biomedically monitor heart activity collected from smart bands. To perform their classification for a signal, they use a Multilayer Perceptron (MLP) classifier for both the traditional and federated learning scenarios. They achieve similar levels of accuracy in their federated learning model compared to the traditional model where the user sends their data features to the server. A use case for this work is performing PPG-based mental stress detection.

In 2021, Şahinbaş et al. [110] demonstrated another application of federated learning in healthcare. In their work, they assumed that each client node was a hospital with various patient records. Their approach consisted of two phases. In Phase I, the model was built at the individual hospital, and in Phase II, the separate hospital models were combined by a trusted third-party authority. In their experiments, they found that the predictive accuracy of their deep learning model decreased as the number of IoT clients at the hospital increased.

In 2022, Tanzir et al. [111] developed a federated learning model for healthcare data. In their approach, the data owners decide on a list of features to use in the comparison, train their local models, and send the local models to be aggregated at the central server. Some applications of their protocol are performing privacy-preserving detection of heart failure and cancer diseases.

6.2.4 Secret Sharing

Shamir published the idea of secret sharing in 1979 [21]. A similar concept was also developed independently by Blakley the same year [112]. In Shamir's work, the author developed a (k, n) threshold scheme [21]. Given a private data point D, D is divided into n pieces $D_1, \ldots, D_n$ such that knowing k or more D_i pieces allows us to compute D but knowing $k - 1$ or fewer points does not allow the computation of D. This simple (k, n) threshold scheme is achieved by using polynomial interpolation. To encode D, a random polynomial $q(x)$ of degree $k-1$ is chosen such that $q(0) = D$. The n shares of D are calculated by computing $q(1), \ldots, q(n)$. Since a polynomial function of degree $k-1$ can be determined using k points on the function, we can use k shares to determine the equation for $q(x)$. Then, we can compute $q(0) = D$ and determine the value of the private data point.

In 1982, Yao [113] extended the concept of secret sharing and developed a two-party computation (2PC) example. The work uses a scenario where two millionaires want to know who is richer without either of them revealing their wealth. The millionaires are able to encrypt their data in such a way that they can determine the result of comparing their data without revealing the data. They will know who is wealthier but not by how much. They do not have any information about the other's wealth. In 1987, Goldreich et al. [114] extended the 2PC scenario to multiparty computation (MPC). In this scenario, the shares for each party are encoded using a polynomial t of n-degree where $t(0)$ is the private value. To determine the private value or comparison, $n+1$ parties need to collaborate and sent their shares to each other.

One of the first large-scale real-world uses of secure multiparty computation (SMPC) was at the Danish sugar beet auction in 2008 [115]. They used SMPC to find the market clearing price which is a price that the sugar beets should sell at. By using SMPC, they ensured that "each bid submitted to the auction was kept encrypted from the time it left the bidder's computer" and that "no single party had access to the bids at any time". The system was able to efficiently compute the price to trade contracts

In 2008, Bogdanov et al. [116] developed Sharmind which is a framework for fast privacy-preserving computations. The goal of their approach was to make large-scale share computations more feasible. They provide a virtual machine (VM) that users can download to implement SMPC in a plug-and-play fashion. This VM creates a runtime environment where users can test private data processing with numerous different privacy-preserving algorithms. Although their work decreased the computation time, it only works in a scenario with three computing parties and only one semi-honest adversary.

In 2014, Turban [117] developed an SMPC protocol suite inspired by Shamir's secret sharing scheme. The author added a new protocol suite to Sharemind that used Shamir's secret sharing and compared it to the existing protocols already implemented in Sharemind. They found that their implementation was about three times slower than Sharemind's additive protocol and that thier multiplication performance was about the same.

In 2018, Park et al. [118] developed a privacy-preserving KNN (PPKNN) scheme to be used for medical diagnosis in e-Health cloud data. Their scenario uses multiparty computation based on Shamir's secret sharing. In their PPKNN scenario, an inquiring patient sends their symptoms to a server. Then, the server will use PPKNN to compute the similarities between the query and each data in the full dataset and convert these similarities into bitwise shares. The full dataset consists of various data from different hospitals. Finally, the server will select the k data with the highest similarities to the client's query record. The diagnosis result for the patient query will be the same as the most common classification of these k data points. The server will send the resulting diagnosis back to the client.

Another approach that applies SMPC to machine learning was demonstrated by Wei et al. in 2022 [119]. In their approach, they develop an SMPC model that uses K-Means clustering with three semi-honest computing servers. The goal of their approach is to achieve full data privacy while also considering the efficiency and practicality of their approach. The authors used replicated secret sharing (RSS) and achieve the same accuracy as the centralized, plaintext K-Means clustering algorithm. Additionally, they found that their privacy-preserving scheme can handle datasets with millions of points in an acceptable amount of time.

In 2022, Yang et. al [120] demonstrated a privacy-preserving scheme that uses KNN and secret sharing. Performing KNN in a privacy-preserving manner can be computationally heavy because each query record needs to be compared to the entire dataset. Their approach uses a decision tree structure to calculate the dataset partition which reduces their computational and communication burden. Their scheme took less than half an hour on average amount 12960 instances compared to privacy-preserving KNN (PPKNN) which takes about 6 hours. When compared to traditional KNN, their approach was less accurate on all of the datasets that they tested with. To achieve faster and lighter computations, they pay a penalty in terms of accuracy.

Chapter 7

Conclusion

In this work, we introduced a framework for classifying ECG signals obtained via a smartwatch device into healthy or AF types. This was done in a manner that respects the privacy of the user and their data. Through a proposed network protocol, it becomes possible for a client and a server to exchange secret shares rather than exchanging raw data. With knowledge of the shares, they can then securely compute the Euclidean distance between signals. This in turn enables to execution of a privacy-preserving version of the KNN algorithm for inference (classification).

In our work, we ran numerous experiments to compare the traditional and secure Euclidean distance computation frameworks. The Euclidean distances obtained by both frameworks were almost identical. On average, the normalized difference between the traditional and secure Euclidean distance was averaged to 1.07% and never exceeded 5.01%.

We also tested the impact of the range of x-values used for Lagrange interpolation in the security framework. We found that there were only minimal differences between the traditional and secure Euclidean distance calculations up to an x-value of 100,000. After that, the difference between the calculations starts to increase exponentially.

To compare the penalty of our framework in terms of the predictive accuracy for a lazy classifier such as the KNN classifier, we compared both the accuracy and computation time to the traditional framework. Using the secure, decentralized approach resulted in no penalty in our accuracy, F1-score, precision, or recall metrics for our KNN model. However, there was a penalty in terms of the computation time and size of the packets sent over the network.

Although the secure computation was significantly slower than the traditional computation, a single prediction only takes about 17 seconds on average which is still highly practical in real-life scenarios.

7.1 Future Work

In the future, we plan on conducting experiments to reduce the prediction time necessary for the secure model. One option is to explore performing different feature extraction so that fewer features need to be sent from the client to the server. The number of features sent directly impacts the size of the message sent from the server back to the client and the number of Lagrange interpolations that a client needs to do for each record-to-record comparison. We also plan on exploring the use of different classifiers. Since KNN needs access to the entire dataset, the client needs to compute the Euclidean distance from itself to *all* of the records in the dataset. Reducing the number of data points that the client needs to perform comparisons between is also likely to reduce the computation time. Towards this end, we plan to explore sending centroids or descriptions of highly dense areas to the client in a privacy-preserving manner rather than sending the actual data points. In this way, a neighborhood of 100 or more data points could be described through just one point. Another option would be the use of Voronoi diagrams.

Bibliography

[1] M. Hasan, "State of iot 2022: Number of connected iot devices growing 18% to 14.4 billion globally," Internet, IoT Analytics, Hamburg, Germany, MAY 2022. [Online]. Available: https://iot-analytics.com/number-connected-iot-devices/

[2] D. S. Lear, "Ready to zap those problem heart cells: Getting on with my ablation," Internet, January 2019. [Online]. Available: https://drscottlear.com/2019/01/30/ready-to-zap-those-problem-heart-cells-getting-on-with-my-ablation/

[3] "What is an ecg?" Internet, AliveCor. [Online]. Available: https://drscottlear.com/2019/01/30/ready-to-zap-those-problem-heart-cells-getting-on-with-my-ablation/

[4] J. Furst, "Recording a 12 lead ecg/ekg," Internet, First Aid For Free, February 2017. [Online]. Available: https://www.firstaidforfree.com/recording-a-12-lead-ecgekg/

[5] E. Blakeway, R. Jabbour, J. Baksi, N. Peters, and R. Touquet, "Ecgs: Colour-coding for initial training," *Resuscitation*, vol. 83, no. 5, pp. e115–e116, 2012.

[6] "Understanding the normal 12-lead ecg/ekg," Internet, Fast Learn ECG, June 2013. [Online]. Available: https://www.youtube.com/watch?v=_nBZqwwoa-U

[7] "Atrial fibrillation," Internet, Mayo Clinic. [Online]. Available: https://www.mayoclinic.org/diseases-conditions/atrial-fibrillation/symptoms-causes/syc-20350624

[8] F. Castells, C. Mora, J. Millet, J. J. Rieta, C. Sánchez, and J. M. Sanchis, "Multidimensional ica for the separation of atrial and ventricular activities from single lead ecgs in paroxysmal atrial fibrillation episodes," in *Independent Component Analysis and Blind Signal Separation: Fifth International Conference, ICA 2004, Granada, Spain, September 22-24, 2004. Proceedings 5.* Springer, 2004, pp. 1229–1236.

[9] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, "Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals," *circulation*, vol. 101, no. 23, pp. e215–e220, 2000.

[10] "Lightwave 0.71." [Online]. Available: https://github.com/PIA-Group/BioSPPy/

[11] P. Wagner, N. Strodthoff, R.-D. Bousseljot, D. Kreiseler, F. I. Lunze, W. Samek, and T. Schaeffter, "Ptb-xl, a large publicly available electrocardiography dataset," *Scientific data*, vol. 7, no. 1, p. 154, 2020.

[12] P. Wagner, N. Strodthoff, R.-D. Bousseljot, W. Samek, and T. Schaeffter, "Ptb-xl, a large publicly available electrocardiography dataset (version 1.0.3)," 2022.

[13] A. Nemcova, R. Smisek, K. Opravilová, M. Vitek, L. Smital, and L. Maršánová, "Brno university of technology ecg quality database (but qdb)," *PhysioNet*, 2020.

[14] G. D. Clifford, C. Liu, B. Moody, H. L. Li-wei, I. Silva, Q. Li, A. Johnson, and R. G. Mark, "Af classification from a short single lead ecg recording: The physionet/computing in cardiology challenge 2017," in *2017 Computing in Cardiology (CinC)*. IEEE, 2017, pp. 1–4.

[15] M. Clinic, "Atrial fibrillation," Internet, Mayo Clinic. [Online]. Available: https://www.mayoclinic.org/diseases-conditions/atrial-fibrillation/symptoms-causes/syc-20350624

[16] M. P. Turakhia, M. Desai, H. Hedlin, A. Rajmane, N. Talati, T. Ferris, S. Desai, D. Nag, M. Patel, P. Kowey *et al.*, "Rationale and design of a large-scale, app-based study to identify cardiac arrhythmias using a smartwatch: The apple heart study," *American heart journal*, vol. 207, pp. 66–75, 2019.

[17] M. V. Perez, K. W. Mahaffey, H. Hedlin, J. S. Rumsfeld, A. Garcia, T. Ferris, V. Balasubramanian, A. M. Russo, A. Rajmane, L. Cheung *et al.*, "Large-scale assessment of a smartwatch to identify atrial fibrillation," *New England Journal of Medicine*, vol. 381, no. 20, pp. 1909–1917, 2019.

[18] "Cardiology teaching package," Internet, Nottingham School of Health Sciences. [Online]. Available: https://www.nottingham.ac.uk/nursing/practice/resources/cardiology/function/normal_duration.php

[19] "12 lead ekg: #1 howell q6," Internet, Quizlet. [Online]. Available: https://quizlet.com/116751872/12-lead-ekg-1-howell-q6-flash-cards/

[20] B. J. Drew, R. M. Califf, M. Funk, E. S. Kaufman, M. W. Krucoff, M. M. Laks, P. W. Macfarlane, C. Sommargren, S. Swiryn, and G. F. Van Hare, "Practice standards for electrocardiographic monitoring in hospital settings: an american heart association scientific statement from the councils on cardiovascular nursing, clinical cardiology, and cardiovascular disease in the young: endorsed by the international society of computerized electrocardiology and the american association of critical-care nurses," *Circulation*, vol. 110, no. 17, pp. 2721–2746, 2004.

[21] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, p. 612–613, Nov. 1979. [Online]. Available: https://doi.org/10.1145/359168.359176

[22] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation," in *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, ser. STOC '88. New York, NY, USA: Association for Computing Machinery, 1988, p. 1–10. [Online]. Available: https://doi.org/10.1145/62212.62213

[23] C. Carreiras, A. P. Alves, A. Lourenço, F. Canento, H. Silva, A. Fred *et al.*, "BioSPPy: Biosignal processing in Python," 2015–. [Online]. Available: https://github.com/PIA-Group/BioSPPy/

[24] S. D. Goodfellow, A. Goodwin, R. Greer, P. C. Laussen, M. Mazwi, and D. Eytan, "Classification of atrial fibrillation using multidisciplinary features and gradient boosting," 2017, pp. 1–4.

[25] F. R. Hobbs, D. Fitzmaurice, J. Mant, E. Murray, S. Jowett, S. Bryan, J. Raftery, M. Davies, and G. Lip, "A randomised controlled trial and cost-effectiveness study of systematic screening (targeted and total population screening) versus routine practice for the detection of atrial fibrillation in people aged 65 and over. the safe study." 2005.

[26] D. A. Fitzmaurice, F. R. Hobbs, S. Jowett, J. Mant, E. T. Murray, R. Holder, J. Raftery, S. Bryan, M. Davies, G. Y. Lip *et al.*, "Screening versus routine practice in detection of atrial fibrillation in patients aged 65 or over: cluster randomised controlled trial," *Bmj*, vol. 335, no. 7616, p. 383, 2007.

[27] K. Poon, P. M. Okin, and P. Kligfield, "Diagnostic performance of a computer-based ecg rhythm algorithm," *Journal of electrocardiology*, vol. 38, no. 3, pp. 235–238, 2005.

[28] U. R. Acharya, S. L. Oh, Y. Hagiwara, J. H. Tan, M. Adam, A. Gertych, and R. San Tan, "A deep convolutional neural network model to classify heartbeats," *Computers in biology and medicine*, vol. 89, pp. 389–396, 2017.

[29] G. B. Moody and R. G. Mark, "The impact of the mit-bih arrhythmia database," *IEEE engineering in medicine and biology magazine*, vol. 20, no. 3, pp. 45–50, 2001.

[30] Y. Guo, H. Wang, H. Zhang, T. Liu, Z. Liang, Y. Xia, L. Yan, Y. Xing, H. Shi, S. Li *et al.*, "Mobile photoplethysmographic technology to detect atrial fibrillation," *Journal of the American College of Cardiology*, vol. 74, no. 19, pp. 2365–2375, 2019.

[31] L. M. Eerikäinen, A. G. Bonomi, L. R. Dekker, and R. M. Aarts, "Atrial fibrillation episodes detected using photoplethysmography: Do we know which are true?" *Journal of the American College of Cardiology*, vol. 75, no. 11, pp. 1365–1365, 2020.

[32] Y. Liu, J. Chen, N. Bao, B. B. Gupta, and Z. Lv, "Survey on atrial fibrillation detection from a single-lead ecg wave for internet of medical things," *Computer Communications*, vol. 178, pp. 245–258, 2021.

[33] V. Maknickas and A. Maknickas, "Atrial fibrillation classification using qrs complex features and lstm," in *2017 Computing in Cardiology (CinC)*. IEEE, 2017, pp. 1–4.

[34] D. Sopic, E. De Giovanni, A. Aminifar, and D. Atienza, "Hierarchical cardiac-rhythm classification based on electrocardiogram morphology," in *2017 Computing in Cardiology (CinC)*. IEEE, 2017, pp. 1–4.

[35] M. García, J. Ródenas, R. Alcaraz, and J. J. Rieta, "Atrial fibrillation screening through combined timing features of short single-lead electrocardiograms," in *2017 Computing in Cardiology (CinC)*. IEEE, 2017, pp. 1–4.

[36] D. Smoleń, "Atrial fibrillation detection using boosting and stacking ensemble," in *2017 Computing in Cardiology (CinC)*. IEEE, 2017, pp. 1–4.

[37] F. Andreotti, O. Carr, M. A. Pimentel, A. Mahdi, and M. De Vos, "Comparing feature-based classifiers and convolutional neural networks to detect arrhythmia from short segments of ecg," in *2017 Computing in Cardiology (CinC)*. IEEE, 2017, pp. 1–4.

[38] B. S. Chandra, C. S. Sastry, S. Jana, and S. Patidar, "Atrial fibrillation detection using convolutional neural networks," in *2017 Computing in Cardiology (CinC)*. IEEE, 2017, pp. 1–4.

[39] B. M. Whitaker, M. Rizwan, V. B. Aydemir, J. M. Rehg, and D. V. Anderson, "Af classification from ecg recording using feature ensemble and sparse coding," in *2017 Computing in Cardiology (CinC)*. IEEE, 2017, pp. 1–4.

[40] L. Billeci, F. Chiarugi, M. Costi, D. Lombardi, and M. Varanini, "Detection of af and other rhythms using rr variability and ecg spectral measures," in *2017 Computing in Cardiology (CinC)*. IEEE, 2017, pp. 1–4.

[41] C. H. Antink, S. Leonhardt, and M. Walter, "Fusing qrs detection and robust interval estimation with a random forest to classify atrial fibrillation," in *2017 Computing in Cardiology (CinC)*. IEEE, 2017, pp. 1–4.

[42] I. Christov, V. Krasteva, I. Simova, T. Neycheva, and R. Schmid, "Multi-parametric analysis for atrial fibrillation classification in ecg," in *2017 Computing in Cardiology (CinC)*. IEEE, 2017, pp. 1–4.

[43] T. Teijeiro, C. A. García, D. Castro, and P. Félix, "Arrhythmia classification from the abductive interpretation of short single-lead ecg records," in *2017 Computing in cardiology (cinc)*. IEEE, 2017, pp. 1–4.

[44] G. Bin, M. Shao, G. Bin, J. Huang, D. Zheng, and S. Wu, "Detection of atrial fibrillation using decision tree ensemble," in *2017 Computing in Cardiology (CinC)*. IEEE, 2017, pp. 1–4.

[45] Y. Liu, K. Wang, Q. Li, R. He, Y. Xia, Z. Li, H. Liu, and H. Zhang, "Diagnosis of af based on time and frequency features by using a hierarchical classifier," in *2017 Computing in Cardiology (CinC)*. IEEE, 2017, pp. 1–4.

[46] S. Datta, C. Puri, A. Mukherjee, R. Banerjee, A. D. Choudhury, R. Singh, A. Ukil, S. Bandyopadhyay, A. Pal, and S. Khandelwal, "Identifying normal, af and other abnormal ecg rhythms using a cascaded binary classifier," in *2017 Computing in cardiology (cinc)*. IEEE, 2017, pp. 1–4.

[47] S. Jiménez-Serrano, J. Yagüe-Mayans, E. Simarro-Mondéjar, C. J. Calvo, F. Castells, and J. Millet, "Atrial fibrillation detection using feedforward neural networks and automatically extracted signal features," in *2017 Computing in Cardiology (CinC)*. IEEE, 2017, pp. 1–4.

[48] S. Ghiasi, M. Abdollahpur, N. Madani, K. Kiani, and A. Ghaffari, "Atrial fibrillation detection using feature based algorithm and deep convolutional neural network," in *2017 Computing in Cardiology (CinC)*. IEEE, 2017, pp. 1–4.

[49] L. Salahuddin, M. G. Jeong, and D. Kim, "Ultra short term analysis of heart rate variability using normal sinus rhythm and atrial fibrillation ecg data," in *2007 9th International Conference on e-Health Networking, Application and Services*. IEEE, 2007, pp. 240–243.

[50] J. Park, S. Lee, and M. Jeon, "Atrial fibrillation detection by heart rate variability in poincare plot," *Biomedical engineering online*, vol. 8, no. 1, pp. 1–12, 2009.

[51] J. Mietus, C. Peng, I. Henry, R. Goldsmith, and A. Goldberger, "The pnnx files: re-examining a widely used heart rate variability measure," *Heart*, vol. 88, no. 4, pp. 378–380, 2002.

[52] M. Zabihi, A. B. Rad, A. K. Katsaggelos, S. Kiranyaz, S. Narkilahti, and M. Gabbouj, "Detection of atrial fibrillation in ecg hand-held devices using a random forest classifier," in *2017 Computing in Cardiology (CinC)*. IEEE, 2017, pp. 1–4.

[53] C. Liu, Q. Li, P. B. Suresh, A. Vest, and G. D. Clifford, "Multi-source features and support vector machine for heart rhythm classification," in *2017 Computing in Cardiology (CinC)*. IEEE, 2017, pp. 1–4.

[54] P. Schwab, G. C. Scebba, J. Zhang, M. Delai, and W. Karlen, "Beat by beat: Classifying cardiac arrhythmias with recurrent neural networks," in *2017 Computing in Cardiology (CinC)*. IEEE, 2017, pp. 1–4.

[55] M. Da Silva-Filarder and F. Marzbanrad, "Combining template-based and feature-based classification to detect atrial fibrillation from a short single lead ecg recording," in *2017 Computing in Cardiology (CinC)*. IEEE, 2017, pp. 1–4.

[56] J. Behar, J. Oster, Q. Li, and G. D. Clifford, "Ecg signal quality during arrhythmia and its application to false alarm reduction," *IEEE transactions on biomedical engineering*, vol. 60, no. 6, pp. 1660–1666, 2013.

[57] R. Smíšek, J. Hejč, M. Ronzhina, A. Němcová, L. Maršánová, J. Chmelík, J. Kolářová, I. Provazník, L. Smital, and M. Vítek, "Svm based ecg classification using rhythm and morphology features, cluster analysis and multilevel noise estimation," in *2017 Computing in Cardiology (CinC)*. IEEE, 2017, pp. 1–4.

[58] S. Yazdani, P. Laub, A. Luca, and J.-M. Vesin, "Heart rhythm classification using short-term ecg atrial and ventricular activity analysis," in *2017 Computing in Cardiology (CinC)*. IEEE, 2017, pp. 1–4.

[59] V. Gliner and Y. Yaniv, “Identification of features for machine learning analysis for automatic arrhythmogenic event classification,” in *2017 Computing in Cardiology (CinC)*. IEEE, 2017, pp. 1–4.

[60] J. A. Behar, A. A. Rosenberg, Y. Yaniv, and J. Oster, “Rhythm and quality classification from short ecgs recorded using a mobile device,” in *2017 Computing in Cardiology (CinC)*. IEEE, 2017, pp. 1–4.

[61] R. Mahajan, R. Kamaleswaran, J. A. Howe, and O. Akbilgic, “Cardiac rhythm classification from a short single lead ecg recording via random forest,” in *2017 Computing in Cardiology (CinC)*. IEEE, 2017, pp. 1–4.

[62] M. Kropf, D. Hayn, and G. Schreier, “Ecg classification based on time and frequency domain features using random forests,” in *2017 Computing in Cardiology (CinC)*. IEEE, 2017, pp. 1–4.

[63] E. E. Coppola, P. K. Gyawali, N. Vanjara, D. Giaime, and L. Wang, “Atrial fibrillation classification from a short single lead ecg recording using hierarchical classifier,” in *2017 Computing in Cardiology (CinC)*. IEEE, 2017, pp. 1–4.

[64] S. Patidar, A. Sharma, and N. Garg, “Automated detection of atrial fibrillation using fourier-bessel expansion and teager energy operator from electrocardiogram signals,” in *2017 Computing in Cardiology (CinC)*. IEEE, 2017, pp. 1–4.

[65] K. Stepien and I. Grzegorczyk, “Classification of ecg recordings with neural networks based on specific morphological features and regularity of the signal,” in *2017 Computing in Cardiology (CinC)*. IEEE, 2017, pp. 1–4.

[66] Z. Wu, T. Lan, C. Yang, and Z. Nie, “A novel method to detect multiple arrhythmias based on time-frequency analysis and convolutional neural networks,” *IEEE Access*, vol. 7, pp. 170 820–170 830, 2019.

[67] S. Parvaneh, J. Rubin, A. Rahman, B. Conroy, and S. Babaeizadeh, "Analyzing single-lead short ecg recordings using dense convolutional neural networks and feature-based post-processing to detect atrial fibrillation," *Physiological measurement*, vol. 39, no. 8, p. 084003, 2018.

[68] M. Limam and F. Precioso, "Atrial fibrillation detection and ecg classification based on convolutional recurrent neural network," in *2017 Computing in Cardiology (CinC)*. IEEE, 2017, pp. 1–4.

[69] X. Zhai and C. Tin, "Automated ecg classification using dual heartbeat coupling based on convolutional neural network," *IEEE Access*, vol. 6, pp. 27 465–27 472, 2018.

[70] S. Kiranyaz, T. Ince, and M. Gabbouj, "Real-time patient-specific ecg classification by 1-d convolutional neural networks," *IEEE Transactions on Biomedical Engineering*, vol. 63, no. 3, pp. 664–675, 2015.

[71] J. Huang, B. Chen, B. Yao, and W. He, "Ecg arrhythmia classification using stft-based spectrogram and convolutional neural network," *IEEE access*, vol. 7, pp. 92 871–92 880, 2019.

[72] A. Ullah, S. M. Anwar, M. Bilal, and R. M. Mehmood, "Classification of arrhythmia by using deep learning with 2-d ecg spectral image representation," *Remote Sensing*, vol. 12, no. 10, p. 1685, 2020.

[73] Y. Xia, N. Wulan, K. Wang, and H. Zhang, "Detecting atrial fibrillation by deep convolutional neural networks," *Computers in biology and medicine*, vol. 93, pp. 84–92, 2018.

[74] P. Warrick and M. N. Homsi, "Cardiac arrhythmia detection from ecg combining convolutional and long short-term memory networks," in *2017 Computing in Cardiology (CinC)*. IEEE, 2017, pp. 1–4.

[75] M. Zihlmann, D. Perekrestenko, and M. Tschannen, "Convolutional recurrent neural networks for electrocardiogram classification," in *2017 Computing in Cardiology (CinC)*. IEEE, 2017, pp. 1–4.

[76] G. B. Moody and R. G. Mark, "The impact of the mit-bih arrhythmia database," *IEEE engineering in medicine and biology magazine*, vol. 20, no. 3, pp. 45–50, 2001.

[77] G. Moody, "A new method for detecting atrial fibrillation using rr intervals," *Proc. Comput. Cardiol.*, vol. 10, pp. 227–230, 1983.

[78] R. L. Rivest, L. Adleman, M. L. Dertouzos *et al.*, "On data banks and privacy homomorphisms," *Foundations of secure computation*, vol. 4, no. 11, pp. 169–180, 1978.

[79] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: Theory and implementation," *ACM Computing Surveys (Csur)*, vol. 51, no. 4, pp. 1–35, 2018.

[80] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[81] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE transactions on information theory*, vol. 31, no. 4, pp. 469–472, 1985.

[82] W. Diffie and M. E. Hellman, "New directions in cryptography," in *Democratizing Cryptography: The Work of Whitfield Diffie and Martin Hellman*, 2022, pp. 365–390.

[83] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology—EUROCRYPT'99: International Conference on the Theory and Application of Cryptographic Techniques Prague, Czech Republic, May 2–6, 1999 Proceedings 18*. Springer, 1999, pp. 223–238.

[84] M. Fellows and N. Koblitz, "Combinatorial cryptosystems galore!" *Contemporary Mathematics*, vol. 168, pp. 51–51, 1994.

[85] M. R. Albrecht, P. Farshim, J.-C. Faugere, and L. Perret, "Polly cracker, revisited," in *Advances in Cryptology–ASIACRYPT 2011: 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings 17*. Springer, 2011, pp. 179–196.

[86] D. Boneh, E.-J. Goh, and K. Nissim, “Evaluating 2-dnf formulas on ciphertexts.” in *TCC*, vol. 3378. Springer, 2005, pp. 325–341.

[87] C. Gentry, *A fully homomorphic encryption scheme.* Stanford university, 2009.

[88] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, “(leveled) fully homomorphic encryption without bootstrapping,” *ACM Transactions on Computation Theory (TOCT)*, vol. 6, no. 3, pp. 1–36, 2014.

[89] S. Rane and P. T. Boufounos, “Privacy-preserving nearest neighbor methods: comparing signals without revealing them,” *IEEE Signal Processing Magazine*, vol. 30, pp. 18–28, 2013.

[90] A. C.-C. Yao, “How to generate and exchange secrets,” in *27th annual symposium on foundations of computer science (Sfcs 1986).* IEEE, 1986, pp. 162–167.

[91] A. Page, O. Kocabas, T. Soyata, M. Aktas, and J.-P. Couderc, “Cloud-based privacy-preserving remote ecg monitoring and surveillance,” *Annals of Noninvasive Electrocardiology*, vol. 20, no. 4, pp. 328–337, 2015.

[92] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachene, “Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds,” in *Advances in Cryptology–ASIACRYPT 2016: 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I 22.* Springer, 2016, pp. 3–33.

[93] L. Ducas and D. Micciancio, “Fhew: bootstrapping homomorphic encryption in less than a second,” in *Advances in Cryptology–EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I 34.* Springer, 2015, pp. 617–640.

[94] L. Liu, J. Su, X. Liu, R. Chen, K. Huang, R. H. Deng, and X. Wang, “Toward highly secure yet efficient knn classification scheme on outsourced cloud data,” *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 9841–9852, 2019.

[95] B. K. Samanthula, Y. Elmehdwi, and W. Jiang, "K-nearest neighbor classification over semantically secure encrypted relational data," *IEEE transactions on Knowledge and data engineering*, vol. 27, no. 5, pp. 1261–1273, 2014.

[96] H. Yang, S. Liang, J. Ni, H. Li, and X. S. Shen, "Secure and efficient knn classification for industrial internet of things," *IEEE Internet of Things Journal*, vol. 7, pp. 10 945–10 954, 2020.

[97] A. Vizitiu, C.-I. Nita, R. M. Toev, T. Suditu, C. Suciu, and L. M. Itu, "Framework for privacy-preserving wearable health data analysis: Proof-of-concept study for atrial fibrillation detection," *Applied Sciences*, vol. 11, no. 19, p. 9049, 2021.

[98] F. A. Almalki and B. O. Soufiene, "Eppda: an efficient and privacy-preserving data aggregation scheme with authentication and authorization for iot-based healthcare applications," *Wireless Communications and Mobile Computing*, vol. 2021, pp. 1–18, 2021.

[99] M. Watkins, C. Dorsey, D. Rennier, T. Polley, A. Sherif, and M. Elsersy, "Privacy-preserving data aggregation scheme for e-health," in *Proceedings of the 2nd International Conference on Emerging Technologies and Intelligent Systems: ICETIS 2022, Volume 2.* Springer, 2022, pp. 638–646.

[100] C. Dwork, "Differential privacy," in *Automata, Languages and Programming: 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II 33.* Springer, 2006, pp. 1–12.

[101] J. Lee and C. Clifton, "How much is enough? choosing ε for differential privacy," in *Information Security: 14th International Conference, ISC 2011, Xi'an, China, October 26-29, 2011. Proceedings 14.* Springer, 2011, pp. 325–340.

[102] B. K. Beaulieu-Jones, Z. S. Wu, C. Williams, R. Lee, S. P. Bhavnani, J. B. Byrd, and C. S. Greene, "Privacy-preserving generative deep neural networks support clinical data sharing," *Circulation: Cardiovascular Quality and Outcomes*, vol. 12, no. 7, p. e005122, 2019.

[103] B. Bebensee, "Local differential privacy: a tutorial," *arXiv preprint arXiv:1907.11908*, 2019.

[104] J. W. Kim, J. H. Lim, S. M. Moon, H. Yoo, and B. Jang, "Privacy-preserving data collection scheme on smartwatch platform," in *2019 IEEE International Conference on Consumer Electronics (ICCE)*. IEEE, 2019, pp. 1–4.

[105] A. Ghazarian, "Assessing the re-identification risk in ecg datasets and an application of privacy preserving techniques in ecg analysis," Ph.D. dissertation, Chapman University, 2021.

[106] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[107] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[108] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, and W. Shi, "Federated learning of predictive models from federated electronic health records," *International journal of medical informatics*, vol. 112, pp. 59–67, 2018.

[109] Y. S. Can and C. Ersoy, "Privacy-preserving federated deep learning for wearable iot-based biomedical monitoring," *ACM Transactions on Internet Technology (TOIT)*, vol. 21, no. 1, pp. 1–17, 2021.

[110] K. Şahinbaş and F. O. Catak, "Secure multi-party computation based privacy preserving data analysis in healthcare iot systems," *arXiv preprint arXiv:2109.14334*, 2021.

[111] T. U. Islam, R. Ghasemi, and N. Mohammed, "Privacy-preserving federated learning model for healthcare data," in *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2022, pp. 0281–0287.

[112] G. R. Blakley, "Safeguarding cryptographic keys," in *Managing Requirements Knowledge, International Workshop on*. IEEE Computer Society, 1979, pp. 313–313.

[113] A. C. Yao, "Protocols for secure computations," in *23rd annual symposium on foundations of computer science (sfcs 1982)*. IEEE, 1982, pp. 160–164.

[114] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation," in *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, 2019, pp. 351–371.

[115] P. Bogetoft, D. L. Christensen, I. Damgård, M. Geisler, T. Jakobsen, M. Krøigaard, J. D. Nielsen, J. B. Nielsen, K. Nielsen, J. Pagter *et al.*, "Secure multiparty computation goes live," in *Financial Cryptography and Data Security: 13th International Conference, FC 2009, Accra Beach, Barbados, February 23-26, 2009. Revised Selected Papers 13*. Springer, 2009, pp. 325–343.

[116] D. Bogdanov, S. Laur, and J. Willemson, "Sharemind: A framework for fast privacy-preserving computations," in *Computer Security-ESORICS 2008: 13th European Symposium on Research in Computer Security, Málaga, Spain, October 6-8, 2008. Proceedings 13*. Springer, 2008, pp. 192–206.

[117] T. Turban, "A secure multi-party computation protocol suite inspired by shamir's secret sharing scheme," *Norwegian University of Science and Technology*, 2014.

[118] J. Park, D. H. Lee *et al.*, "Privacy preserving k-nearest neighbor for medical diagnosis in e-health cloud," *Journal of healthcare engineering*, vol. 2018, 2018.

[119] W. Wei, C. Tang, and Y. Chen, "Efficient privacy-preserving k-means clustering from secret-sharing-based secure three-party computation," *Entropy*, vol. 24, no. 8, p. 1145, 2022.

[120] J.-K. Yang, K.-C. Huang, C.-Y. Chung, Y.-C. Chen, and T.-W. Wu, "Efficient privacy preserving nearest neighboring classification from tree structures and secret sharing," in *ICC 2022-IEEE International Conference on Communications*. IEEE, 2022, pp. 5615–5620.