University of Zurich[UZH]

MASTER THESIS – Communication Systems Group, Prof. Dr. Burkhard Stiller

Targeted Synthetic BLE Packet Data Generation Approach

Keisuke Yokota
Zurich, Switzerland
Student ID: 22-738-165

Supervisor: Katharina O. E. Müller, Weijie Niu, Prof. Dr. Burkhard Stiller
Date of Submission: 05.03.2025

University of Zurich
Department of Informatics (IFI)
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland

ifi

Master Thesis
Communication Systems Group (CSG)
Department of Informatics (IFI)
University of Zurich
Binzmuehlestrasse 14, CH-8050 Zurich, Switzerland
URL: http://www.csg.uzh.ch/

Declaration of Independence

I hereby declare that I have composed this work independently and without the use of any aids other than those declared (including generative AI such as ChatGPT). I am aware that I take full responsibility for the scientific character of the submitted text myself, even if AI aids were used and declared (after written confirmation by the supervising professor). All passages taken verbatim or in sense from published or unpublished writings are identified as such. The work has not yet been submitted in the same or similar form or in excerpts as part of another examination.

Zürich, 05.03.2025 Keisuke Gokota

Signature of student

Abstract

Bluetooth Low Energy (BLE) wird in einer Vielzahl von Geräten verwendet, darunter Smartphones und Smartwatches, wobei Ortungsgeräte wie AirTag und SmartTag besonders beliebt sind. Diese Geräte decken jedoch auch Risiken in Bezug auf die Verletzung der Privatsphäre der Nutzer und Stalking auf. Als Reaktion darauf haben auf Machine Learning basierende Methoden zur kollektiven Erkennung von BLE-Ortungsgeräten verschiedener Hersteller an Aufmerksamkeit gewonnen. Diese Masterarbeit konzentriert sich dementsprechend auf das Problem, dass die begrenzte Verfügbarkeit und Vielfalt von BLE-Paketen die Leistung von Machine Learning einschränkt, und untersucht die Erstellung synthetischer Daten, um dieses Problem zu lösen. Es wird eine Methode zur Erzeugung synthetischer BLE-Paketdaten unter Verwendung eines Markov-Modells untersucht, das sich für die Verarbeitung strukturierter und zeitlicher Datenreihen eignet. Konkret werden synthetische Daten für Samsungs SmartTag (in der Nähe) mit Hilfe eines Markov-Modells erzeugt. Die generierten synthetischen Daten erhöhen die Vielfalt des Trainingsdatensatzes, was auf das Potenzial zur Verbesserung der Modellgenauigkeit hindeutet.

Bluetooth Low Energy (BLE) is used in many devices, including smartphones and smartwatches, with tracking devices such as AirTag and SmartTag being particularly popular. However, these devices also pose risks related to user privacy violations and stalking. In response, machine learning-based methods for collectively detecting BLE tracking devices from multiple vendors have gained attention. This study focuses on the issue of the limited availability and diversity of BLE packets constraining the performance of machine learning models. It explores the creation of synthetic data to address this challenge. A method for generating synthetic BLE packet data using a Markov model, suitable for handling structured and time-series data, is investigated. Specifically, synthetic data for Samsung's SmartTag (nearby) is generated using a Markov model. The generated synthetic data enhances the diversity of the training dataset, suggesting the potential for improving model accuracy.

Acknowledgments

I would like to express my deep gratitude to many individuals who supported and advised me throughout this research. First and foremost, I extend my heartfelt thanks to my supervisor, Katharina O. E. Müller, for her persistent and precise guidance from planning this study to experimental design and thesis writing through our weekly meetings. Her clear feedback and encouragement significantly contributed to maintaining my motivation. Without her support, completing this research would not have been possible.

Additionally, I am profoundly grateful to Prof. Dr. Burkhard Stiller, Weijie Niu, and the members of the Communication Systems Group for their valuable advice, which allowed me to approach my research from a multifaceted perspective.

I also want to thank my friends in Japan and those I met in Switzerland for their various forms of support. Their companionship made daily life enjoyable and provided essential breaks from my research, helping me maintain a healthy balance between work and relaxation.

Finally, I sincerely thank my family for their unwavering support from my admission to the University of Zurich to my daily life. Their encouragement and understanding gave me the strength to overcome challenges and stay focused on my goals.

Contents

Chapter 1

Introduction

Bluetooth Low Energy (BLE) is a low-power wireless communication technology. Apple introduced BLE for the first time in their smartphone, the iPhone 4s, in 2011 [1]. Currently, many companies use BLE to connect many devices simultaneously, including smartphones, smartwatches, wireless speakers, and fitness watches. Among them, the use of personal tracking devices such as Apple's AirTag and Samsung's SmartTag is growing rapidly [2, 3, 4]. While these devices are useful for tracking lost items and obtaining location information, they can also pose security risks [5, 6], such as privacy breaches and unauthorized tracking [7, 8]. In recent years, research into the detection and classification of BLE tracking devices has been conducted [9, 10, 11], and methods have been proposed to effectively identify these devices using machine learning models [9, 10].

Stefan Richard Saxer [9] created the dataset and built the machine-learning model based on the vast amount of BLE data collected over 600 hours from tracking devices such as AirTag, SmartTag, and Tile Mate. The model was able to classify devices with a high degree of accuracy, but challenges remain regarding the diversity of the data and the generalization performance of the model.

1.1 Motivation and Thesis Goals

When creating a machine learning model for classifying BLE tracking devices, if there is not enough data, it may not be possible to extract enough meaningful features, leading to a decrease in the accuracy of the model [12]. In particular, in the case of Stefan's study, there was not enough data that matched real-world environments and conditions, so while the accuracy was high for test data, it was low for real-world data [9]. In addition, data collection involves enormous time and monetary costs. Furthermore, collecting sufficient data from all devices is difficult because of the bias in the devices actually used. There are also privacy issues, such as the ability to track the user's location and activities based on the data held by the BLE device [6]. Therefore, it is important to create synthetic data.

This study investigates the use of the Markov model among the methods for creating synthetic data. The Markov model is excellent for generating time series data because it learns the transition probabilities between states of data and predicts the next state based on those probabilities [13]. It is considered suitable for reproducing the continuity and patterns of BLE packets. Generating synthetic data using a Markov model improves the quantity and diversity of data. This allows the machine learning model to extract meaningful features and increase accuracy. Moreover, by increasing the diversity of the data, the problem of noise in the data, which often occurs in real-world data, can be addressed. This will make it possible to detect devices using the model not only in limited environments but also in a variety of situations.

This thesis aims to investigate ways to generate high-quality synthetic BLE data, with a special focus on investigating the applicability of the Markov model in this use case.

Therefore, the goals of this thesis are:

1. Clarify the appropriateness of the Markov model for creating BLE data.
2. Create synthetic data of BLE using a Markov model.
3. Evaluate the synthetic data

A literature review was conducted to achieve the first goal. Comparisons were made between GAN and a Markov model because GAN has recently received much attention for creating synthetic data. Also, related works that addressed the synthesis of BLE packet data and examples of synthetic data creation using the Markov model were examined. Based on these studies, the Markov model was explained as appropriate for this research. Then, the synthetic data was created using the Markov model to accomplish the second goal. Lastly, the generated data was evaluated.

1.2 Thesis Outline

The following chapters of this thesis cover roughly the following topics.

Chapter 2 introduces the basic theoretical concepts and previous research essential for understanding this study's content.

Chapter 3 explains the procedure for generating synthetic data using the Markov model, the reason why the Markov model is selected to create synthetic data for BLE packets, and for which devices the synthetic data was created.

Chapter 4 describes details of the data to which the Markov model was applied. Moreover, the results of evaluating the created synthetic data and the considerations that can be drawn from them are shown.

Chapter 5 summarizes this thesis, answers the research questions, and indicates future work.

Chapter 2

Fundamentals

This chapter introduces the basic theoretical concepts and previous research essential for understanding this study's content. In particular, the basic concepts of BLE and synthetic data are explained. Moreover, Stefan's work is explained because it is this study's most critical prior work. In addition, some studies that tried creating synthetic data for BLE packet data and synthetic data using the Markov model are introduced.

2.1 Background

This section introduces the fundamental theoretical ideas that are helpful to understanding this research. There are two subsections. First, the concept of the BLE is explained. The way of communication of BLE and the structure of packet data used in communication are explained in relation to the basic idea of BLE. Then, the basic matters regarding synthetic data are explained. The advantages of synthetic data and the methods used to create synthetic data are introduced. GAN, a method using deep learning that has been attracting attention recently, and the Markov model used in this research are explained in the second subsection. These subsections are loosely connected.

2.1.1 Bluetooth Low Energy (BLE)

This chapter describes Bluetooth Low Energy (BLE), including its basic concepts, communication methods, and packet structure. The BLE description is based on the Bluetooth Core Specification [14] and Nordic Developer Academy's description of Bluetooth Low Energy [15].

Basic Concepts and Communication Methods

Bluetooth Low Energy (BLE) is one of the Bluetooth standards. BLE features lower power consumption than Bluetooth Classic, a conventional communication method.

In BLE communication, there are two roles: Central and Peripheral, and communication takes place between the two. In general, PCs and smartphones play the central role, while smartwatches and lost-and-found tags such as AirTag play the peripheral role. First, the peripheral advertises. The advertising is performed when the peripheral waits for a connection, and data is sent to an unspecified number of parties rather than one-to-one. By receiving these advertisements, the center knows what peripheral devices are around it. The central selects the party to which it wants to connect from the advertisements it finds and sends a connection request. When the peripheral device receives the connection request, it stops advertising and switches to a one-to-one connection.

In other words, the Peripheral device periodically advertises to an unspecified number of devices until it establishes one-to-one communication with a specific central (Figure 2.1). Therefore, by acquiring and analyzing this advertisement, peripheral devices, in this case, BLE trackers, can be classified.

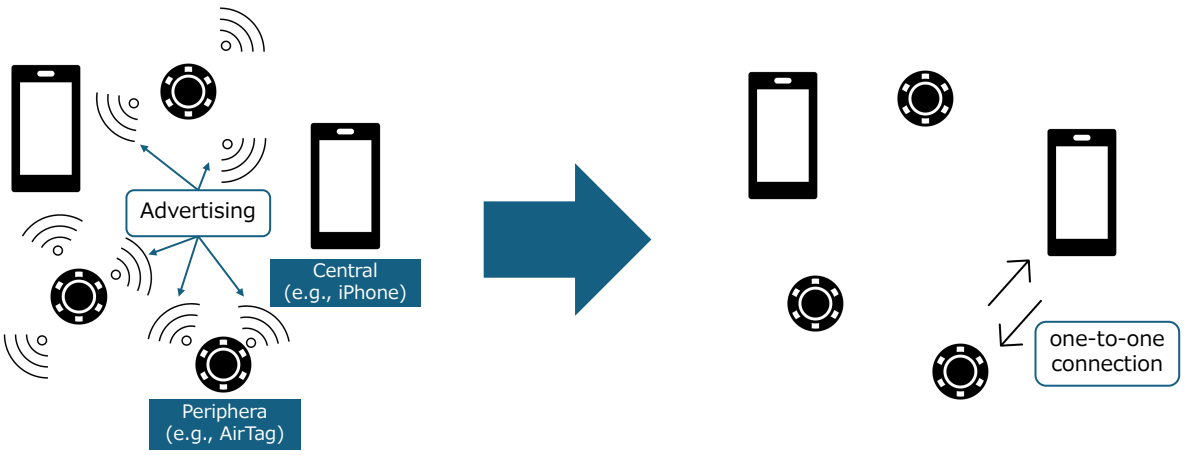


Figure 2.1: BLE Advertising and Connection Establishment Process

Packet Structure

Bluetooth Low Energy packets follow a certain structure (Figure 2.2). As mentioned earlier, advertising needs to be analyzed to classify BLE trackers. Among the Preamble, Access Address, Protocol Data Unit (PDU), and Cyclic Redundancy Check (CRC) that make up a packet (Figure 2.2), the Protocol Data Unit (PDU) is involved in advertising. So, it is sufficient to focus on the PDU. Although not shown in Figure 2.2, there are two types of PDUs: Advertising Physical Channel PDUs, which are used for advertising, and Data Physical Channel PDUs, which are used for one-to-one communication with the central. In this research, only the Advertising Physical Channel PDUs are relevant. Depending on the type of PDU, the structure of the payload, which will be described later, may differ slightly. Furthermore, Figure 2.3 shows the example of the packet structure of Apple's AirTag.

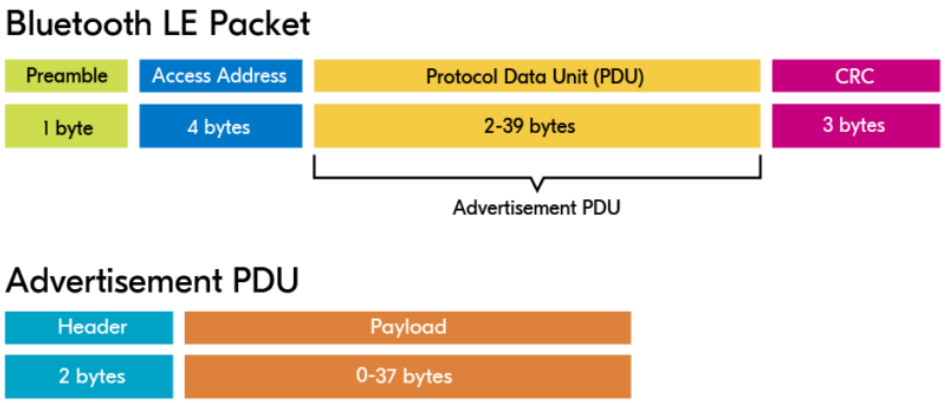


Figure 2.2: Structure of BLE Packets and Advertisement PDUs [15]

```
Bluetooth Low Energy Link Layer
    Access Address: 0x8e89bed6
  > Packet Header: 0x2560 (PDU Type: ADV_IND, ChSel: #2, TxAdd: Random)
    Advertising Address: fd:e5:68:1f:9f:9c
  v Advertising Data
    v Manufacturer Specific
        Length: 30
        Type: Manufacturer Specific (0xff)
        Company ID: Apple, Inc. (0x004c)
      > Data: 121910bc3904307fa14c973cb6aefd08ba26426c0a15fb2a4c022c
    CRC: 0x0f0801
```

Figure 2.3: Example of a BLE Advertising Packet (AirTag) [16]

The PDU comprises a Header and a Payload (Figure 2.4). The Payload is divided into Advertising Address (AdvA) and Advertising Data (AdvData). AdvA is located in the first 6 bytes of a BLE Advertisement Packet (PDU) payload portion and serves as the source address. A source address is an address that uniquely identifies the source device and is essential for identifying the packet sender in BLE communications. However, a single BLE device may use multiple source addresses. In addition, there are some BLE devices that change their source addresses periodically to protect privacy.

AdvData contains an Advertisement data packet. For example, Advertising data packets may include BT_DATA_NAME_COMPLETE or BT_DATA_MANUFACTURER_DATA. BT_DATA_NAME_COMPLETE is the name of the device, which is recognized by humans through smartphones. BT_DATA_MANUFACTURER_DATA is a manufacturer-specific number, and each company is assigned a unique number. Such data serves as an important feature for classification.

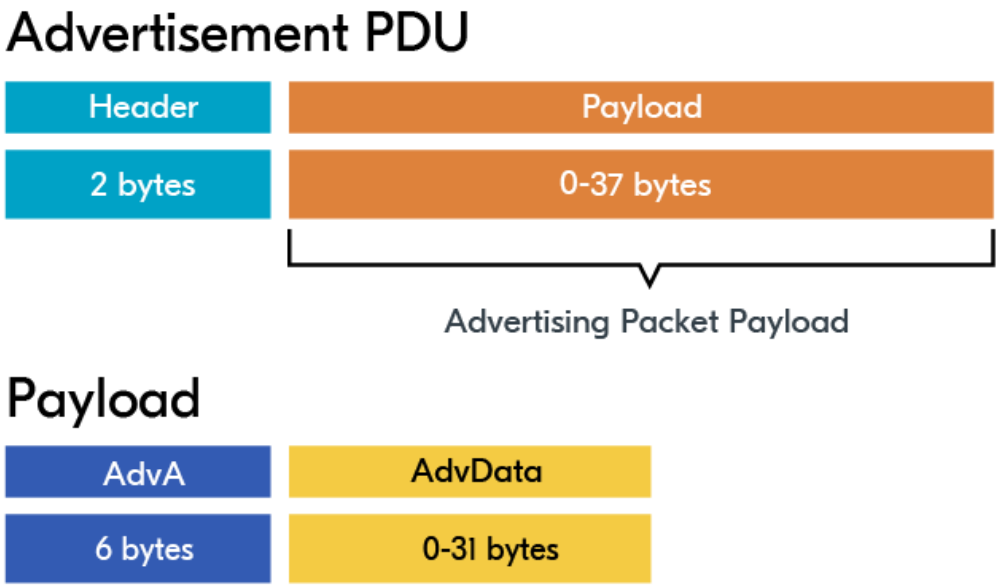


Figure 2.4: Structure of Advertisement PDU and Payload in BLE [15].

Moreover, the Advertising Data packet can be broken down into three parts: AD length, AD type, and AD data (Figure 2.5). AD length is the length of the entire structure (including AD type and AD data). AD type indicates the type of data. AD data is the actual data. These are also important features to classify BLE devices.

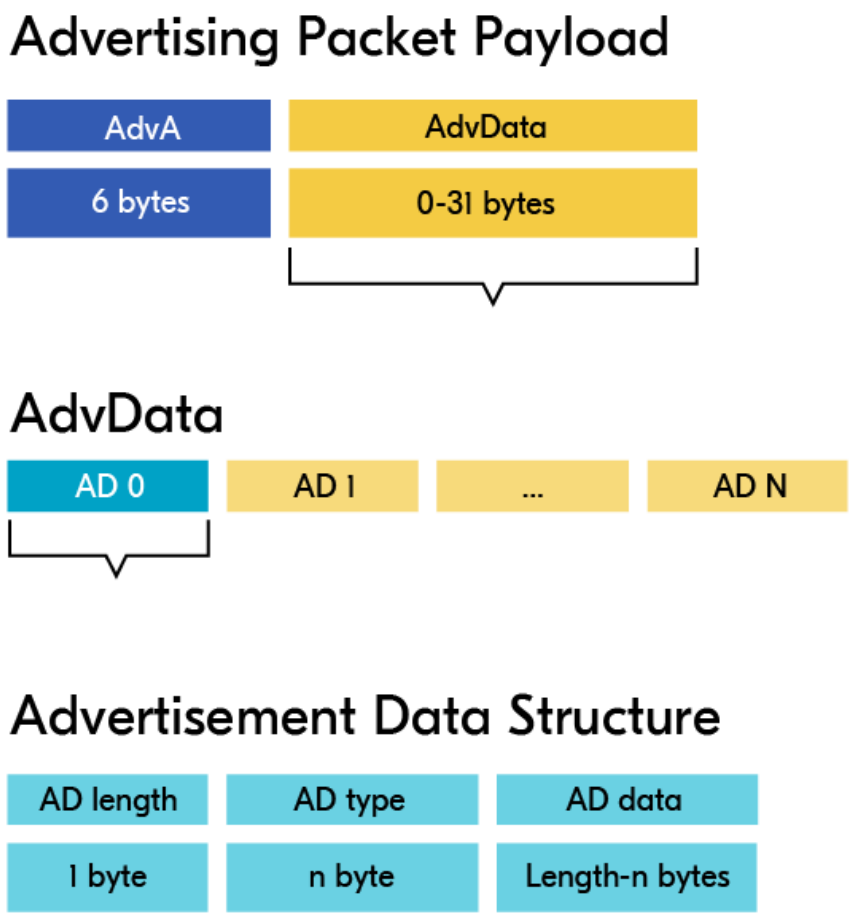


Figure 2.5: Advertisement Data Structure in BLE [15].

To summarize, when classifying BLE devices, it is necessary to focus on their advertising. BLE packets can also be broken down into smaller pieces. PDU is focused on a deep-dive into advertising. The PDU is divided into a Header and a Payload. Moreover, the Payload can be split into AdvA and AdvData. AdvData can be further divided into AD length, AD type, and AD data, which are important features for classifying BLE devices. These are important features for classifying BLE devices.

2.1.2 Synthetic Data

This section provides an explanation of synthetic data, including a description of the basic concepts of synthetic data and the general methods to generate it. In addition, an

explanation of the Markov model, which is the focus of this paper, is provided.

Basic Concepts

Synthetic data is artificial data created to mimic real-world data characteristics and structures [17]. Synthetic data has the advantage of protecting privacy, eliminating data shortages, and reducing bias. Synthetic data can reduce the risk of privacy violations because it retains the structure and statistical characteristics of the original data but is not the actual data [18, 19]. There is no limit to the amount of data because synthetic data can be created on a computer [20]. Moreover, real-world data may contain more specific categories or attributes than others, causing bias. Synthetic data can help eliminate bias by increasing the data for attributes with fewer data [21, 22].

BLE packet data contains personal information such as user location and activity history and may violate privacy. There are also time and financial costs associated with collecting the data. Furthermore, the data may be biased toward AirTag and SmartTag, which have the largest market share as BLE trackers. Synthetic data could be a solution to these problems. However, ensuring that the synthetic data created is a good reflection of the real world is necessary.

Methods for Creating Synthetic Data

There are three main methods for creating synthetic data: statistical methods, machine learning model methods, and deep learning methods [23]. Statistical methods analyze data distribution and create synthetic data based on that distribution. For example, Bayesian inference is used to model the distribution and structure of the data. The synthetic data are generated by combining prior and posterior probabilities [24]. In machine learning model methods, data is created by having the machine learning model learn the characteristics of the real-world data. For example, decision trees or random forests are used to learn the characteristics of the data to produce synthetic data [25]. Methods that use deep learning create data using advanced deep learning models such as Generative Adversarial Network (GAN) [26, 27]. GAN is good at enriching the diversity of data because it can produce data based on random noise [28, 29]. GAN is used in a variety of situations, but especially in image recognition [30] and speech data synthesis [31].

Generative Adversarial Network (GAN)

GAN consists of two neural networks, a Generator, and a Discriminator, that iteratively train each other to produce data. Both the generator and discriminator individually analyze the training data and its attributes. The Generator creates new data by adding the random noise to the properties of the trained data. The discriminator takes the data created by the Generator and real data. Then, the discriminator checks whether the data belongs to the original dataset. This is how GAN creates the data. By repeating this process, the Generator learns to generate data that the Discriminator judges as real data,

and the Discriminator learns to discriminate more correctly. In this way, the Generator and Discriminator evolve in competition with each other to generate more realistic data [32]. Figure 2.6 shows the comprehensive overview of the GAN Structure.

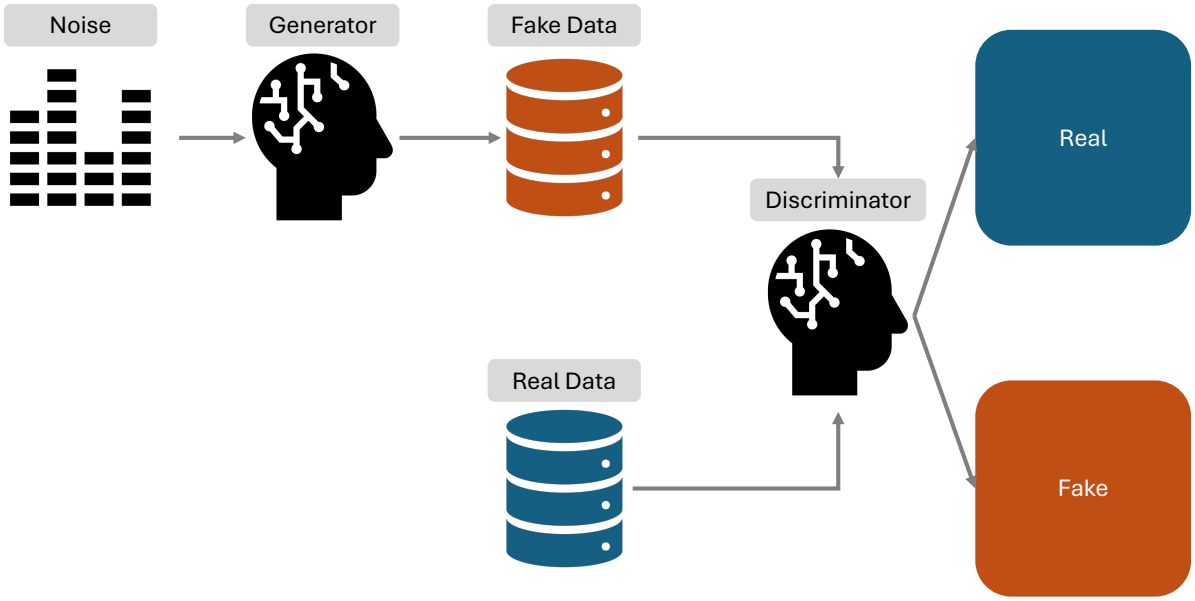


Figure 2.6: Comprehensive Overview of the GAN Structure

Algorithm 1 is based on the GAN training algorithm proposed by [32]. As the algorithm indicates, the generator takes noise as input and generates new data from it. This noise provides randomness to the generated data. This allows the generator to generate diverse data. On the other hand, if the structure of the data is fixed, such as BLE packet data, this randomness can be a disadvantage.

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets [32]

for number of training iterations **do**
  for k steps **do**
    Sample minibatch of m noise samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_g(z)$.
    Sample minibatch of m examples $\{x^{(1)}, \ldots, x^{(m)}\}$ from data generating distribution $p_{\text{data}}(x)$.
    Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D(x^{(i)}) + \log \left(1 - D(G(z^{(i)})) \right) \right].$$

  end for
  Sample minibatch of m noise samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_g(z)$.
  Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log \left(1 - D(G(z^{(i)})) \right).$$

end for

Markov Model

A Markov model is one statistical method for creating synthetic data. This study investigates the Markov model to generate synthetic data for BLE.

A stochastic process that is "Markovian" in that only the current state depends on the following state and not on past states is called a Markov model. That is, the property that the next state is determined from the current state and has nothing to do with the past state [33]. For example, the weather forecasting model is shown in Figure 2.7. If the current weather is cloudy, there is a 50% chance of rain, a 20% chance of sunny, and a 30% chance of cloudy again. Currently, the following weather conditions depend only on the current weather conditions: cloudy. The background that led to the current cloudy weather is not taken into account.

These properties make a Markov model suitable for time series data. It is also suitable for structured data because it is based solely on state transitions known from existing data and does not add random noise. For example, it is used to create synthetic data for weather forecasts and financial markets [34].

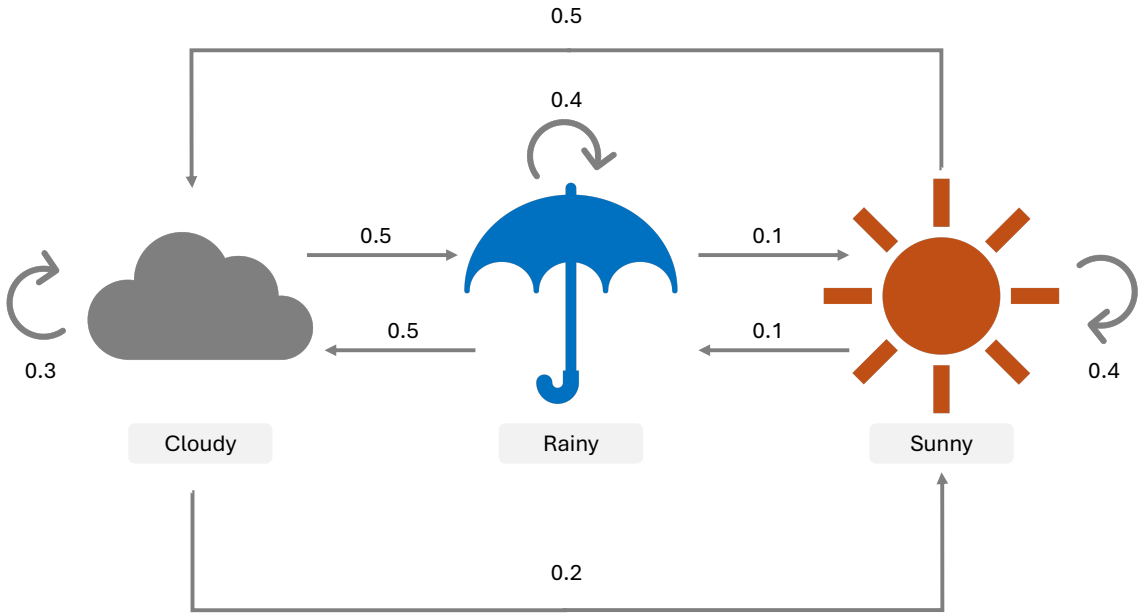


Figure 2.7: Example of the Markov Model

There is also a hidden Markov model, an extension of the Markov model. A hidden Markov model has both hidden state and observable data. The hidden state cannot be observed directly, so the state is inferred from observable data. The observable data follows a probability distribution for each state [35]. For example, as shown in Figure 2.8, a model estimates the weather of a friend's location based on the friend's behavior. In this case, the weather is a hidden state and cannot be observed directly. On the other hand, the friend's behavior is observable data and can be observed. Then, the observable data are walk, clean, shop, shop, clean, ... The goal is to estimate the change in weather that cannot be observed directly from these data.

In this way, a hidden Markov model can be used for more complex problems because it can include dependencies between hidden states and observable data [36]. Hidden Markov models are used in speech recognition, gene finding, and profiling [37].

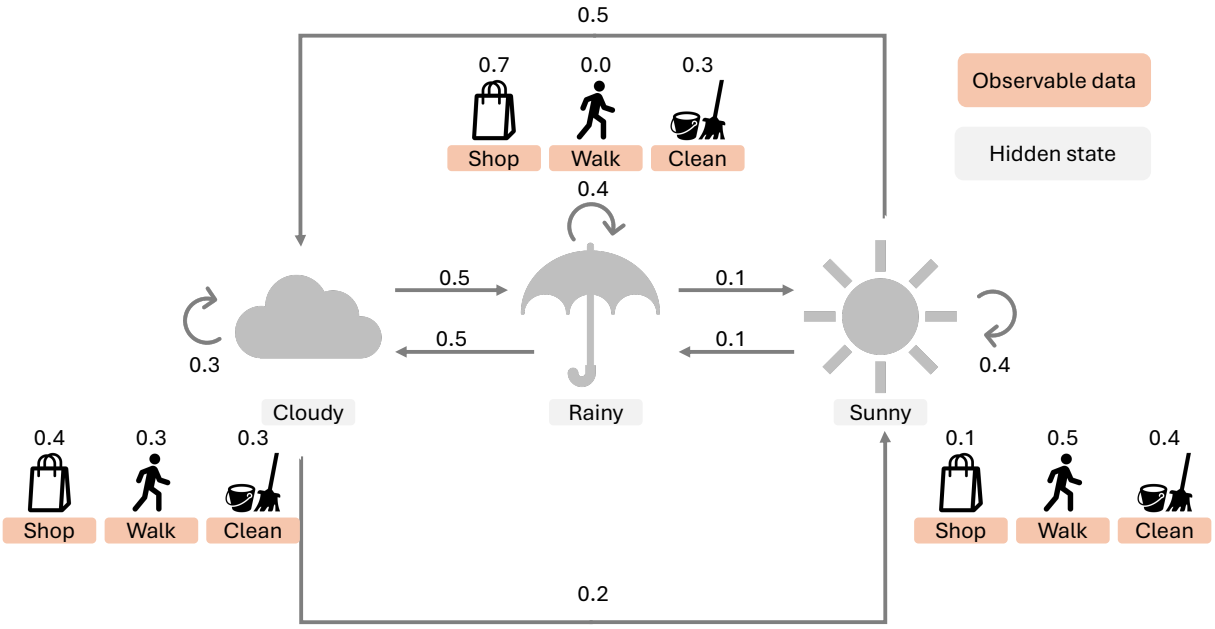


Figure 2.8: Example of the Hidden Markov Model

Since the interest of this study is the generation of synthetic data itself, the main topic is not to investigate the events behind the data. Hence, the Markov model is appropriate rather than the Hidden Markov model. More details of the method selection are explained in the Design chapter (Chapter 3).

2.1.3 Previous Research

This study extends Stefan's work [9]. While Bluetooth-based tracking devices help track lost items, they are also used for the nefarious purpose of stalking. In response, Bluetooth tracker vendors like Apple have created applications to detect privacy violations using their trackers. However, such applications can only detect their device; for example, the applications by Apple can only detect the device of Apple. Therefore, to create a machine-learning model that can detect trackers from multiple vendors at once, Stefan collected Bluetooth data from Bluetooth devices, created a machine-learning model using the collected data, and used the created machine-learning model in a real-world environment to perform Bluetooth tracker detection.

Stefan first collected data in a controlled, laboratory-like environment to train and test the machine-learning model. In this environment, a metal box (Faraday cage) was used to block outside radio interference, and packets were collected. Specifically, as shown in Figure 2.9, the Faraday cage contains only an nRF 52840 DK logic board and one BLE device. The nRF 52840 DK logic board captures the packets sent by the BLE and sends them to the PC connected to it with a USB micro B cable. Since there was only one BLE device in the cage, the packets could be captured without interference from other devices. Incidentally, when collecting BLE packets in the nearby state, in addition to the BLE devices, the paired devices were also included together. For example, a SmartTag

(nearby) and a Samsung Galaxy S23 Ultra were included in the Faraday cage together (Figure 2.10). This is because a device in a nearby state will go to a lost state if it cannot find a paired device. For efficiency, as shown in Figure 2.11, Stefan also collected data using two BLE devices in the Faraday Cage. In this case, he first turned on only one device and captured packets from that device to identify the source address. Then, he turned on another device and simultaneously captured data from both devices. In this way, packets with an address different from the known source address of the first device can be distinguished and accurately labeled as coming from the second device. Because the Faraday cage blocks outside interference, only data from these two devices is reliably captured, making it easy to identify each device.

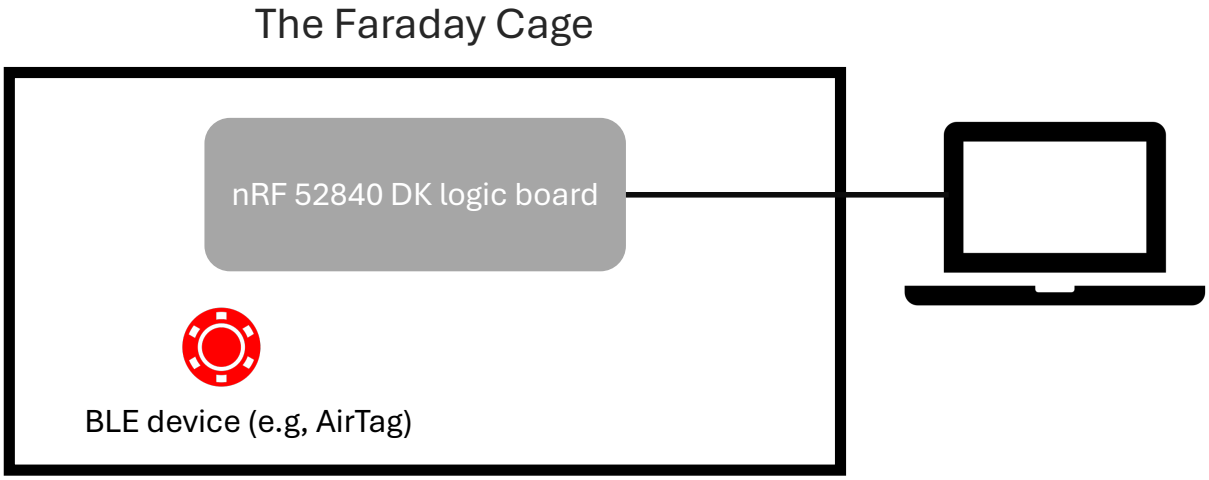


Figure 2.9: The Faraday Cage with One BLE Device and the nRF 52840 DK Logic Board

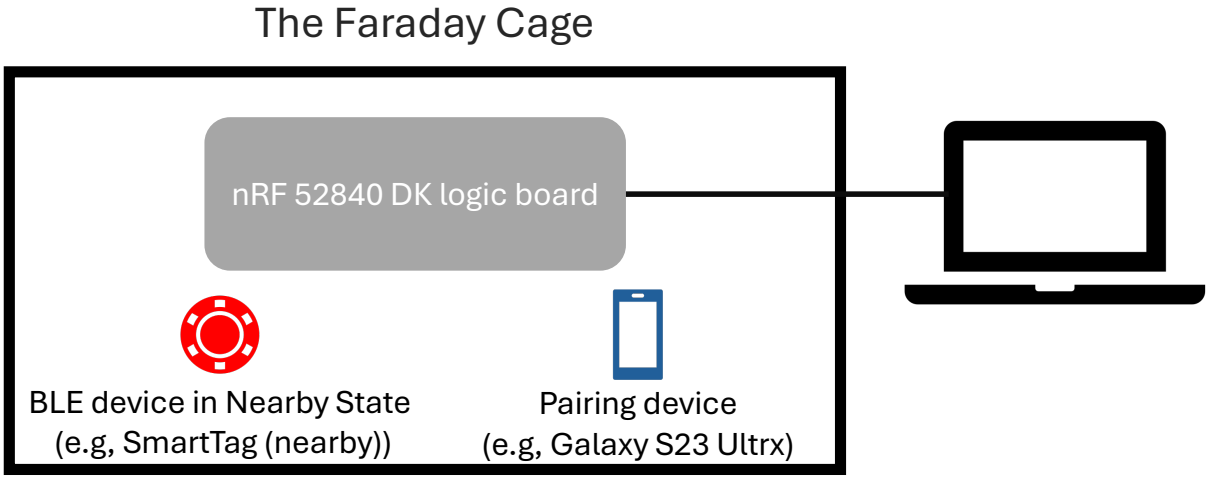


Figure 2.10: The Faraday Cage with One BLE Device in a Nearby State and One Pairing Device

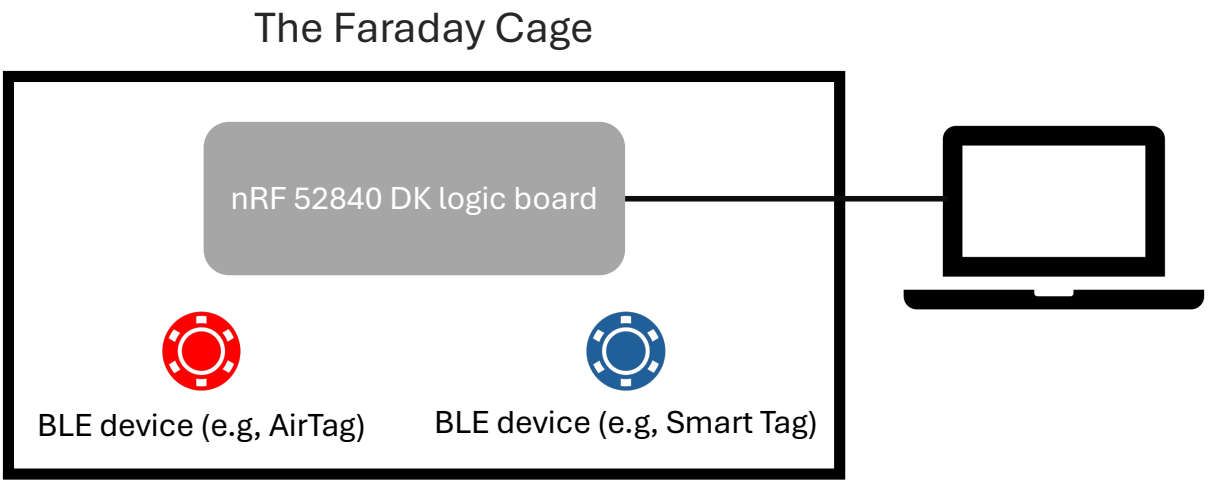


Figure 2.11: The Faraday Cage with Two BLE Devices and the nRF 52840 DK Logic Board

As a result, approximately 30 million packets of data were collected from a wide variety of devices, ranging from traditional BLE trackers such as AirTag and SmartTag to BLE tracker-enabled devices such as the iPhone and MacBook, and a few devices such as the Lenovo Yoga Laptop. In addition, Stefan labeled the collected data. Table 2.1 shows the examples of the dataset created by Stefan.

Table 2.1: Example of the Dataset Created by Stefan (SmartTag: Lost) [38]

Number	Time	Source	Destination	Protocol	Channel	Length Packet	Length Header	AD Type	Company ID	MS Data	UUID	Service Data	PDU
1	1712778478	6d:40:e6:f1:49:01	ff:ff:ff:ff:ff:ff	LE LL	37	63	37	Flags, 16-bit Service Class UUIDs (incomplete), Service Data - 16 bit UUID	Samsung Electronics Co., Ltd.			130f050217cfebfab1c2e0deb7000000558045e8	ADV_IND
2	1712778480	6d:40:e6:f1:49:01	ff:ff:ff:ff:ff:ff	LE LL	38	63	37	Flags, 16-bit Service Class UUIDs (incomplete), Service Data - 16 bit UUID	Samsung Electronics Co., Ltd.			130f050217cfebfab1c2e0deb7000000558045e8	ADV_IND
3	1712778482	6d:40:e6:f1:49:01	ff:ff:ff:ff:ff:ff	LE LL	39	63	37	Flags, 16-bit Service Class UUIDs (incomplete), Service Data - 16 bit UUID	Samsung Electronics Co., Ltd.			130f050217cfebfab1c2e0deb7000000558045e8	ADV_IND
4	1712778482	6d:40:e6:f1:49:01	ff:ff:ff:ff:ff:ff	LE LL	37	63	37	Flags, 16-bit Service Class UUIDs (incomplete), Service Data - 16 bit UUID	Samsung Electronics Co., Ltd.			130f050217cfebfab1c2e0deb7000000558045e8	ADV_IND
5	1712778484	6d:40:e6:f1:49:01	ff:ff:ff:ff:ff:ff	LE LL	38	63	37	Flags, 16-bit Service Class UUIDs (incomplete), Service Data - 16 bit UUID	Samsung Electronics Co., Ltd.			130f050217cfebfab1c2e0deb7000000558045e8	ADV_IND
6	1712778484	6d:40:e6:f1:49:01	ff:ff:ff:ff:ff:ff	LE LL	39	63	37	Flags, 16-bit Service Class UUIDs (incomplete), Service Data - 16 bit UUID	Samsung Electronics Co., Ltd.			130f050217cfebfab1c2e0deb7000000558045e8	ADV_IND

Additionally, data was collected and tested at Zurich Central Station in a real-world setting to evaluate how the machine learning model could perform. Because of the difficulty of labeling this data, real-world data for inference is not labeled.

The data collected by Stefan is used in this research.

Limitations

Stefan built a machine-learning model to classify BLE trackers using data collected in a controlled environment. The model showed more than 99% accuracy on test data. On the other hand, it was only 80% accurate for data collected in the real world. The assumption is that real-world data is not labeled and, therefore, requires using a softmax confidence threshold for inference, which is likely less accurate than the labeled test data. Softmax confidence thresholds determine whether classification results are reliable based on the confidence the model predicts for each class. Expressly, a classification result is accepted only if the confidence level indicated by the model for the classification result for a given

sample exceeds the threshold value. Conversely, the classification result is rejected if the confidence level does not reach the threshold.

The other reason for the lower accuracy of real-world data is the difference between test and real-world data. Test data is separate from the kernel used during training in relatively consistent environments and conditions. This leads to models predicting high accuracy against test data because the properties align with features that models can easily interpret. In contrast, real-world data is gathered in unpredictable conditions, with background noise and variety. This includes noise and interference from different devices, making the data erratic and increasing the chance of model misclassification.

Potential Solutions

Stefan suggested increasing the amount of data to address the lack of accuracy for real-world data. By increasing the amount and type of data that can be used for training, it is thought that models can be improved to withstand unexpected noise and interferences between devices, as is the case with real-world data.

Alternatively, increasing the amount of data is not the only step to improve model accuracy against real-world data; improving feature extraction and creating more nuanced models are also options. Stefan believes that everything necessary for classification has already been extracted as features at inference time, and misclassification is not due to a lack of features. Additionally, since the current model already performs well on the test set, upgrading it, for example, by tuning hyperparameters, is not expected to yield significant improvements.

Given the high accuracy already maintained for the test data, I agree with Stefan's idea. Therefore, hyperparameter optimization would be less effective. Instead, increasing the amount and type of data is most reasonable since the lack of real-world data is considered a bottleneck. For these reasons, this research aims to create synthetic data.

2.2 Related Work

In this section, the related works are introduced. The related works were searched from two aspects. First, research related to BLE was reviewed. While studies addressing challenges related to BLE were identified, no studies specifically focusing on the generation of synthetic data for BLE packet data were found. Second, the studies that focused on generating synthetic data using Markov models were searched for. There are studies that tackled producing synthetic data using a Markov model, mainly time series data. Therefore, using a Markov model, it is significant to create synthetic data for BLE packet data, which is time series data.

2.2.1 Research Related to BLE

[39, 40] conducted a study on indoor location tracking using BLE beacons. In this work, they generated synthetic data for RSSI used for BLE beacon location tracking, which measures the strength of the received signal in wireless communications. The Wasserstein interpolation method was used to create the synthetic data. However, the targeted data was RSSI data, not BLE packet data. [41] performed simulations to study the effects of multiple BLE advertising packets colliding. Moreover, [42] developed a MATLAB Simulink library to simulate BLE wireless sensor networks.

The studies that generated synthetic data for BLE packet data could not be identified. The studies presented so far have dealt with topics related to BLE. However, they did not create synthetic data for BLE packet data.

Moreover, research has been done to generate synthetic data of IP packets and TCP/UDP flow. [43] worked on generating synthetic data of real network traffic that can be used in developing and evaluating network security and intrusion detection systems (IDS). Network data has the same challenges as BLE packet data, such as privacy protection and the high cost of data collection. The synthetic data was designed to realistically mimic the network's packet flow, reproducing the original data's statistical characteristics and distribution. However, this study also did not produce synthetic data for BLE packet data.

To sum up, no studies were found in which researchers created synthetic data for BLE packet data itself. However, some studies have focused on generating synthetic data related to BLE packets, simulating BLE networks, or creating synthetic data for network traffic.

2.2.2 Creation of Synthetic Data Using the Markov Model

[44] used a Markov model to develop synthetic data for minute-by-minute high-resolution solar radiation data. The data generated was achieved by modeling the time series of the "clarity index" of the solar radiation. The synthetic data were statistically consistent with the actual data. Moreover, [45] also designed synthetic data for wind speed data using a Markov model. Specifically, wind speed data at 5-minute intervals were created based on wind speed data at 30-minute intervals. Synthetic data were created using a first-order Markov model, which considers that the current state depends only on the immediately preceding state, and a second-order Markov model, which considers that the current state depends on the two immediately. The synthetic data were statistically similar to the actual data. Additionally, [46] worked on synthetic data for wind power generation.

There are studies that have used a hidden Markov model. For example, [47] used a hidden Markov model to build synthetic data for financial time series data. The created data were found to have characteristics similar to those of the actual data. [48] also fabricated synthetic data for disease incidence data using a hidden Markov model. [49] used a hidden Markov model to construct synthetic data for population data. The synthetic data has a very similar structure to the original data. Hidden Markov models were used

because of the interdependencies among attributes such as age, gender, education level, and occupation in the population data, which are difficult to model with simple Markov models.

There are researchers who tried to create synthetic data using the Markov model [44, 45, 46] or the hidden Markov model [47, 48, 49]. Both models were used to produce synthetic data for time series data, suggesting a good compatibility between time series data and a Markov model. Also, simple Markov models are better suited for relatively simple syntheses, such as wind speed. In contrast, hidden Markov models are better suited when one is interested in transitions in the state behind the observed data, such as disease incidence, based on previous research. Additionally, no study has used the Markov model to create synthetic data for BLE packet data.

To sum up, no study has developed synthetic data for BLE, as Table 2.2 indicates. Therefore, it is significant to generate synthetic data for BLE packets. Also, prior studies show that Markov models have been used to create synthetic data for time series data. Since BLE packet data is also time series data, it is reasonable to use Markov models. In addition, the objective of this study is to produce synthetic data for BLE packet data, and simulation is out of scope.

Table 2.2: Overview of the Related Works

Research	Synthesizing BLE Packet Data	Related		Markov Model		Time Series Data	Can It Be Used to Create Synthetic BLE Packets?	
		Synthesizing Data	Simulation	Not Hidden	Hidden		Method	Data
[39]	No	Yes	No	No	No	Yes	No	Partially
[40]	No	Yes	No	No	No	Yes	No	Partially
[41]	No	No	Yes	No	No	Yes	No	Partially
[42]	No	No	Yes	No	No	Yes	No	Partially
[43]	No	Yes	No	No	No	Yes	No	Partially
[44]	No	Yes	No	Yes	No	Yes	Yes	No
[45]	No	Yes	No	Yes	No	Yes	Yes	No
[46]	No	Yes	No	Yes	No	Yes	Yes	No
[47]	No	Yes	No	No	Yes	Yes	No	No
[48]	No	Yes	No	No	Yes	No	No	No
[49]	No	Yes	No	No	Yes	No	No	No
This research	Yes	Yes	No	Yes	No	Yes	Yes	Yes

Chapter 3

Design

This chapter provides an overview of the methods to create and evaluate synthetic data. Moreover, it explains why the Markov model was chosen to synthesize the data, comparing it to other methods such as GAN. It also describes which BLE packet data was focused on. In addition, the structure of the data from which the data was synthesized is explained.

3.1 Overview of the Method

A Markov model was employed among several methods to create the synthetic data. Also, SmartTag (nearby) was focused on several devices. The details of the reasons for selecting the Markov model and SmartTag (nearby) are explained in Sections 3.2 and 3.3.

As Figure 3.1 shows, synthetic data creation is divided into five steps.

1. Preprocess the raw data.
2. Separate the preprocessed data into individual columns.
3. Examine each column: Columns with only a single unique value were excluded from the Markov model processing because they did not require any synthetic data generation. Columns with two or more unique values that were highly correlated were grouped and treated as a single-state transition. For instance, columns CH 37, CH 38, and CH 39 were grouped.
4. Generating synthetic data:
    - 4-1 Columns with only a single unique value were duplicated without applying a Markov model.
    - 4-2 For columns with multiple unique values, state transition probabilities were computed for each grouped column. Using these probabilities, a Markov model was applied to generate synthetic data.

5. Combine the columns created in steps 4-1 and 4-2 to construct a single synthetic dataset. The synthetic data produced is 600,000 rows × 30 columns.

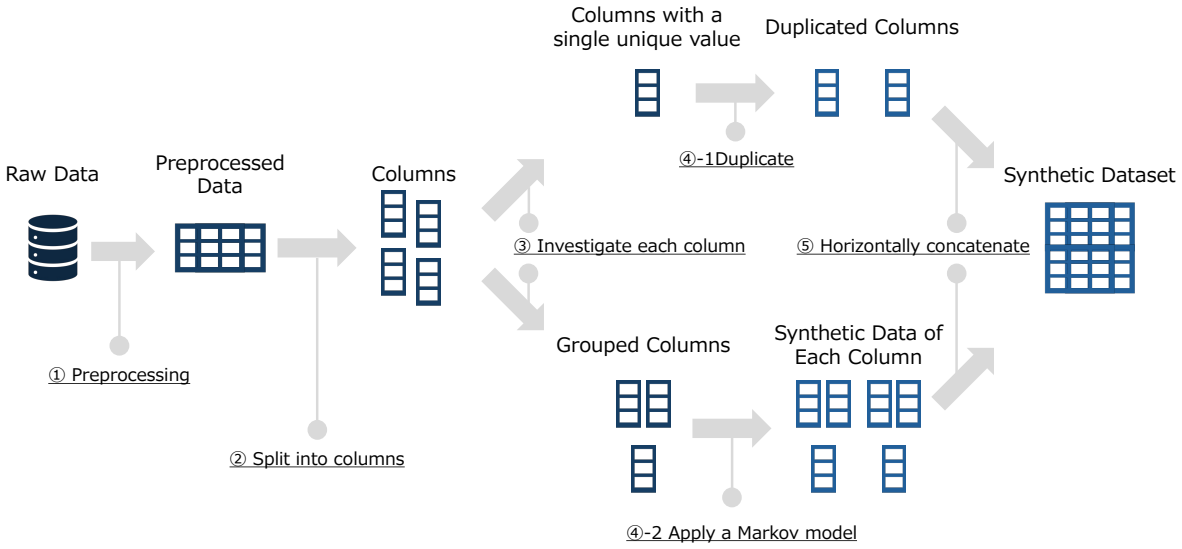


Figure 3.1: Pipeline for Synthetic BLE Packet Data Generation

The following columns had only one unique value, so the Markov model was not used. Each column had only the values shown in parentheses. Therefore, data with only the values in parentheses were created for each column by duplicating the data.

- AD Manufacturer Specific (0)
- AD Tx Power Level (0)
- AD 16-bit Service Class UUIDs (0)
- COMP Apple (0)
- Comp Other (0)
- UUID Tile (0)
- PDU ADV_NONCONN_IND (0)
- PDU ADV_SCAN_IND (0)
- CT 07 (0)
- CT 12 (0)
- CT Other (0)
- ST 3 (0)
- Label (SmartTag (nearby))

Since the columns below had multiple values, synthetic data was created using a Markov model. Columns highly related to each other were grouped and treated as one state. The columns to which the Markov model was applied are described for each group.

- Length Packet, Length Header, Length MS Data, Length Service Data: Since these columns are about packet length and are highly relevant, they were handled together.
- CH 37, CH38, CH 39: Since these columns are about the channel, they were handled together.
- AD Flags, AD Service Data - 16 bit UUID, AD 16-bit Service Class UUIDs (incomplete), AD Other: Since all of these columns are about AD type and are highly relevant, they were handled together.
- UUID Samsung, UUID Other: Since all of these columns are about UUID and are highly relevant, they were handled together.
- PDU ADV_IND, PDU Other: Since all of these columns are about PDU type and are highly relevant, they were handled together.
- ST 5, ST Other: Since all of these columns are about SmartTag type and are highly relevant, they were handled together.

The method of setting initial values is an issue when using the Markov model to generate synthetic data. This study adopted a technique to address this issue, where all states were designated as initial values and an equal amount of data was generated for each state. For example, if there are three types of channel states (CH 37, CH 38, and CH 39) and the total number of synthetic data is set to 600,000, each state is assigned 200,000 data points. This is calculated by dividing 600,000 equally by 3, the number of states. If the final number of data to be created cannot be divided by the number of states, the remainder is allocated equally to each state, one case per state. At this time, the difference in the number of data produced is either 1 or 0, which is not a significant problem. This method makes it possible to obtain a certain number of data, even from states that are observed infrequently. It contributes to increasing the diversity of synthetic data.

Next, each column in the SmartTag (nearby) dataset was examined individually. First, columns containing only one value were separated from those with multiple values; columns with a single value were assigned that value in the synthetic data, eliminating the need to apply a Markov model. Subsequently, columns with multiple values that are strongly related were grouped and treated as a single state. For instance, the columns CH 37, CH 38, and CH 39, which represent channels, were combined into a single state. State transition probabilities were then calculated for each summarized state. Based on the obtained state transition probabilities, synthetic data were generated for the same number of steps, considering all states as initial values.

Finally, a single data set was created by horizontally merging the data produced by duplicating it with the data created by the Markov model.

The evaluation of the synthetic data was conducted from two perspectives. One perspective involved examining the differences in the distribution of values between the original data and the synthetic data to assess the consistency of their data structures. Another perspective focused on comparing the inference results of two neural network models: one trained exclusively on the original data and the other trained on a combination of the original and synthetic data. This comparison aimed to determine whether incorporating synthetic data impacted the model's performance. Details of the evaluation are explained in Section 4.2.

3.2 Method Selection

This research used the Markov model to create synthetic data for the BLE packet. BLE packet data is time-series data transmitted at regular intervals and is either structured or data with fixed fields, as described in Section 2.1. Markov models are well suited for creating synthetic data with these characteristics.

Another option is a method using deep learning, such as GAN. However, as described in Section 2.1, BLE packets have specific fields such as "Preamble," "Access Address," "Protocol Data Unit (PDU)," and "Cyclic Redundancy Check (CRC)," and the size and role of each field are strictly defined. This makes generative models that make large changes to the data structure, especially deep learning methods such as GAN, unsuitable for fixed-format data such as BLE. GAN is good for creating diverse and complex structures because it produces data by adding random noise. Still, these characteristics make it less suitable for data with a consistent structure, such as BLE packet data.

A hidden Markov model was also considered. However, this study aims to generate the BLE packet data itself, which does not require the estimation of hidden background information or states. Hidden Markov models are suitable for generating data with complex internal structures because they consider "hidden states" behind observable data. Still, they are redundant for data that requires only simple structures and explicit state transitions, such as BLE packet data. Therefore, a hidden Markov model is overly complex for generating BLE packet data.

The Markov model is suitable for time series data because it has a transition mechanism in which the next state is determined based on the current state. In addition, Markov models do not destroy existing structures, making them suitable for creating synthetic data for BLE packet data, which is structural data. Moreover, the Markov model is very simple, making it easy to implement and interpret. For these reasons, this study used a Markov model to create synthetic data for BLE packet data.

3.3 Target Data

In generating synthetic data, packet data from Samsung's SmartTag in the nearby state was selected among several BLE devices. The rationale for choosing SmartTag (nearby) and its data structure are subsequently explained.

3.3.1 Reason for Data Selection

Data in the nearby state of Samsung's SmartTag was synthesized. Not raw data but the preprocessed data, which are ready for model training, were targeted. There are three reasons for synthesizing SmartTag (nearby) data. First, as shown in Figure 3.2, this is the smallest amount of data among the packets collected by Stefan. Second, SmartTag (nearby) was not classified correctly, comparing other devices. Figure 3.3 shows the confidence level for real-world data (Bahnhof_V2), which was also collected by Stefan and stored in Kaggle [38]. The neural network used for inference outputs the probability that an input belongs to a certain label as a probability. Specifically, the confidence level of each data point is plotted based on percentiles. The X-axis is the percentile of data points (% of Packets). It shows the position of the data points in order of confidence. The y-axis is the Confidence. It indicates the confidence level of each data point. Thus, the higher the value on the y-axis, the higher the confidence in the data classification. Comparing these graphs, the SmarTag (nearby) has a lower confidence level than the other devices. Synthesizing SmartTag (nearby) can increase confidence levels. Lastly, Stefan has already worked on improving the accuracy of SmartTag (nearby) classification [50], and it is possible to compare the results of that work with the results of the method using a Markov model to generate synthetic data. Stefan aimed to improve the classification accuracy by creating synthetic data for non-Samsung products misclassified as SmartTag (nearby). This method can be compared to the results of increasing the data for SmartTag (nearby) itself using a Markov model.

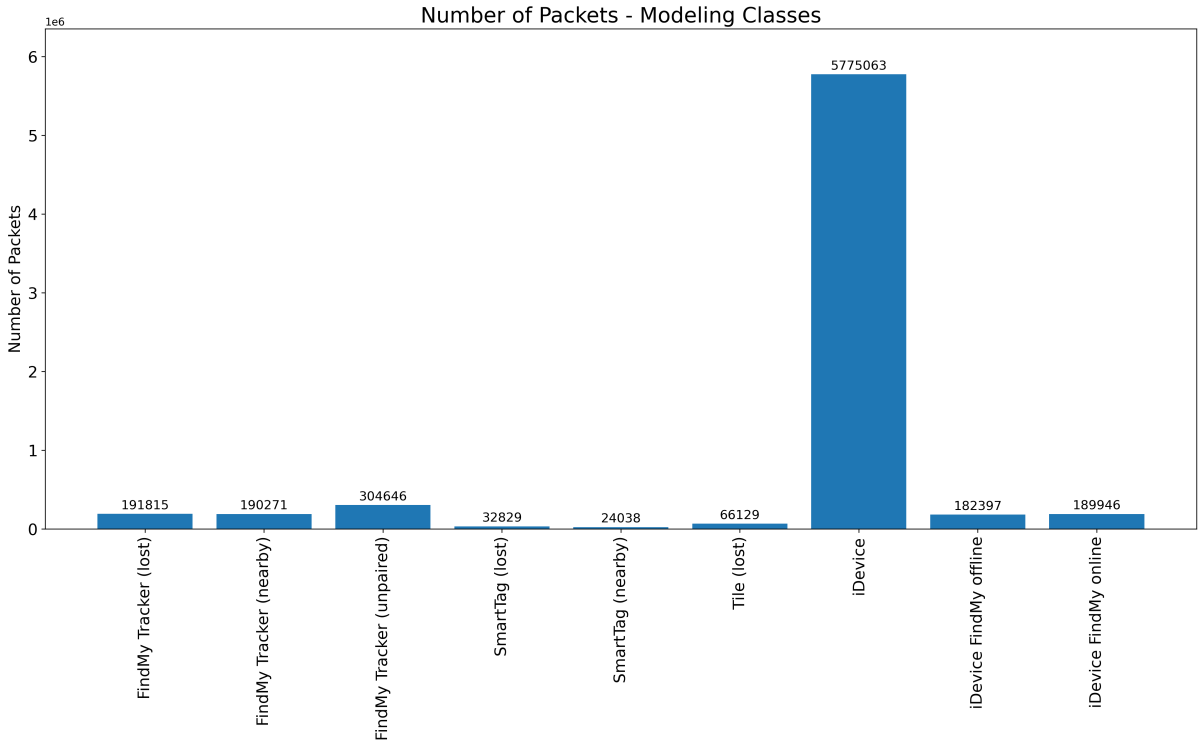
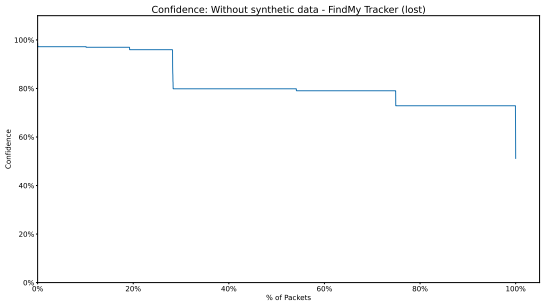


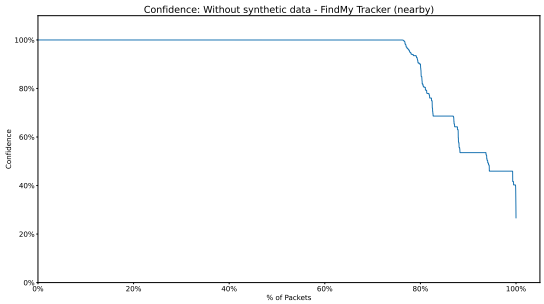
Figure 3.2: Number of Packets [9]

The data was synthesized in a preprocessed state, ready for training machine learning

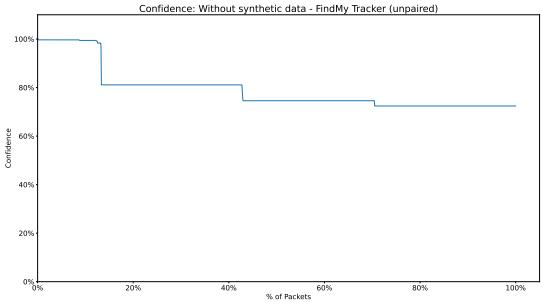
models, rather than using raw data. The reason is that the models created by Stefan were used to evaluate the synthesized data. Preprocessed data has had noise and missing values removed, and essential features of the BLE packet have been extracted, so variations can be added by synthesizing data for features that affect the model.



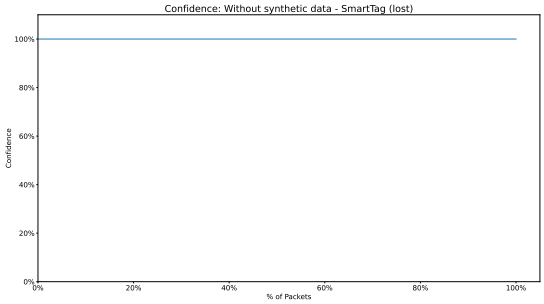
(a) Confidence - FindMy Tracker (lost)



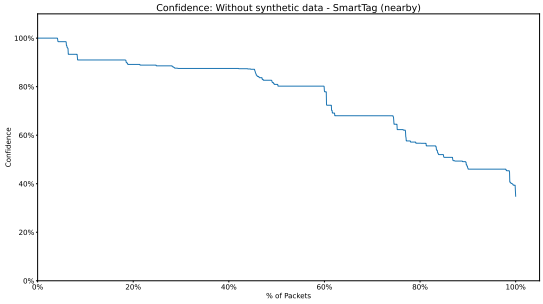
(b) Confidence - FindMy Tracker (nearby)



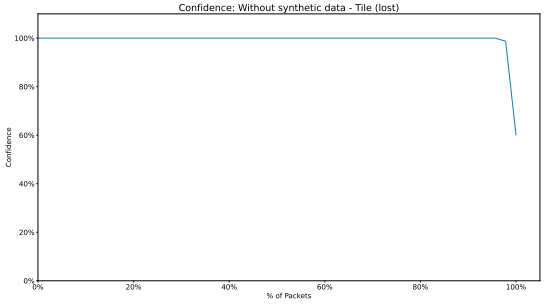
(c) Confidence - FindMy Tracker (unpaired)



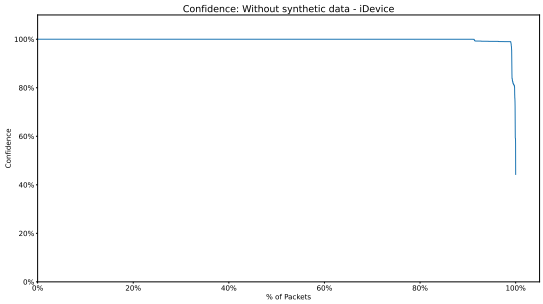
(d) Confidence - SmartTag (lost)



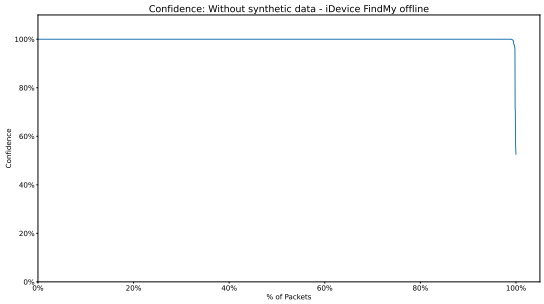
(e) Confidence - SmartTag (nearby)



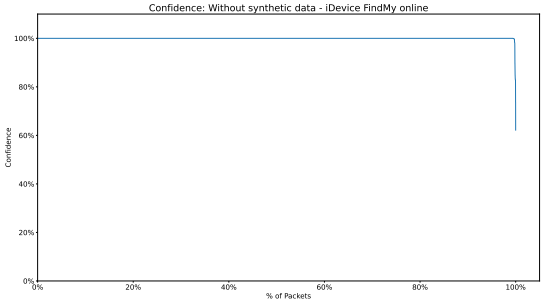
(f) Confidence - Tile (lost)



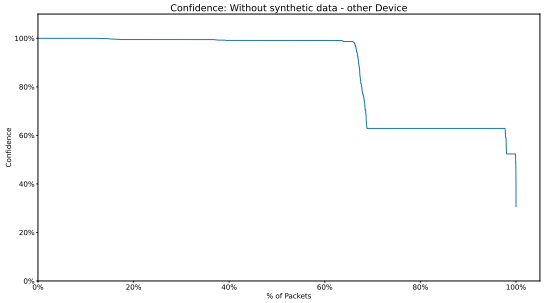
(g) Confidence - iDevice



(h) Confidence - iDevice FindMy offline



(i) Confidence - iDevice FindMy online



(j) Confidence - other Device

Figure 3.3: Confidence - Without Synthetic Data [50]

3.3.2 Data Structure

In this study, synthetic data was created from preprocessed data. The preprocessing method followed the pipeline created by Stefan. For details of the preprocessing, please refer to Stefan's paper [9]. The general flow of the process is as follows. First, features that were considered essential for BLE device categorization were extracted. Next, one-hot encoding was applied to the categorical data. After these preprocessing steps, the SmartTag (nearby) columns are as follows.

- **Length Packet:** The length of the packet in the layer used by the nRF 58420 DK to transmit to the host device when collected by Stefan in the Faraday cage.
- **Length Header:** The length of the packet in Bluetooth Low Energy link layer.
- **Length MS Data:** The length of the Manufacturer Specific Data (MS Data)contained in the payload of the BLE packet. MS Data is a field into which companies can embed their own data.
- **Length Service Data:** The length of the Service Data. Service Data is used to identify the features and services and to provide additional information.
- **CH 37, CH 38, CH 39:** Dummy variable columns representing the Advertising Channel.
- **AD Manufacturer Specific, AD Flags, AD Tx Power Level, AD Service Data - 16 bit UUID, AD 16-bit Service Class UUIDs, AD 16-bit Service Class UUIDs (incomplete):** Dummy variable columns representing the AD type.
- **AD Other:** The number of occurrences of AD types other than those shown above.
- **COMP Apple, COMP Other:** Dummy variable columns representing the company ID found in Manufacturer Specific Data, with all zeros in SmartTag(nearby) dataset since SmartTag does not correspond to either Comp Apple or Comp Other.
- **UUID Samsung, UUID Tile:** An extended category sequence expressing the number of occurrences of each UUID. A UUID is a unique identifier used in BLE (Bluetooth Low Energy) and other communication protocols. In the context of BLE, it is primarily used to distinguish devices, services, or specific attributes.
- **UUID Other:** UUIDs representing company names other than Apple, Samsun, and Tile.
- **PDU ADV_IND, PDU ADV_NO NCONN_IND, PDU ADV_SCAN_IND, PDU Other:** Dummy variable columns representing the PDU type.
- **CT 07, CT 12, CT Other:** Dummy variable columns representing the Continuity type. However, Continuity is used between Apple devices and has nothing to do with Samsung's SmartTag. Therefore, all values are set to 0.

- **ST3, ST5, ST Other:** Dummy variable columns representing the SmartTag type. In Samsung's SmarTag, bits 5∼7 of the Service Data represent the state of the SmartTag (Figure 3.4). This part is called SmartTag Type. It is useful for classifying SmartTag states, such as nearby, lost, and searching states.
- **Label** Label for each data. All values are SmartTag (nearby) in the SmartTag(nearby) dataset.

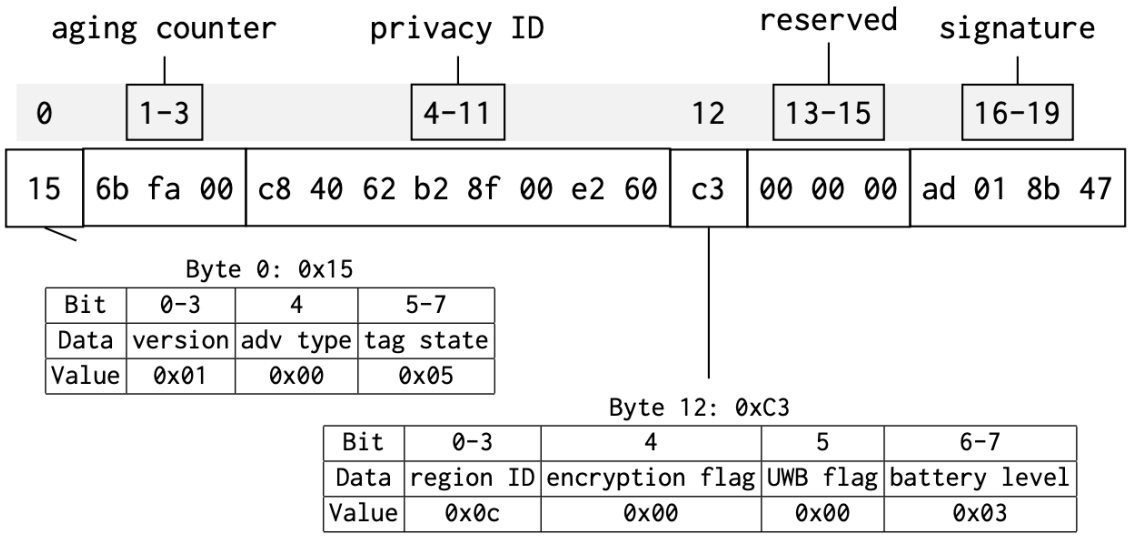


Figure 3.4: Structure of the Service Data Used by the Samsung SmartTag [51]

Chapter 4

Results and Evaluation

This chapter describes the state transition probabilities obtained during the creation of synthetic data using the Markov model. Python (version 3.12) was used for this process. The program can be found on GitHub. The details about GitHub are written in Appendix A.

Moreover, the results of the evaluation of the created synthetic data are explained. The synthetic data were evaluated from two perspectives. First, the structure of the data was compared. Second, the impact of the synthetic data on the model was examined by comparing a model that used the synthetic data as training data in addition to the original data with a model that used only the original data as training data.

In addition to evaluating the synthetic data, the considerations of the results were discussed.

4.1 State Transition Definition

Each feature to which the Markov model was applied was explained using the State Transition Model and the Transition Probability Matrix.

4.1.1 Length (Packet, Header, MS Data, Service Data)

Length Packet, Length Header, Length MS Data, and Length Service Data—each pertaining to packet length—were treated collectively as a single state. Length Packet had four values: 63, 38, 36, and 32; Length Header had three values: 37, 12, and 6; and Length MS Data had two values: 176 and 0; and Length Service Data also had two values: 160, and 0.

Basically, the Length Packet is longer than the Length Header by 26. Therefore, if the Length Header has only three values (37, 12, and 6), the number 11 in the Length Packet is unnatural. In addition, the number of occurrences is extremely low, 11 out of 24,038

data points. Incidentally, the number of occurrences of 63 is 23,837, while both 38 and 32 appear 95 times. Moreover, the number of occurrences of 176 among Length MS Data is extremely low, only 2 out of 24,038. These unnatural numbers may be because Stefan included not only SmartTag but also Galaxy in the Faraday cage when collecting SmartTag (nearby) data. However, the exact cause remains unknown. Therefore, synthetic data was generated, including this data.

The transition diagram of data related to packet length is shown in Figure 4.1. This diagram illustrates the state transition probabilities between different packet length configurations, including Length Packet, Length Header, Length MS Data, and Length Service Data, from left to right. For example, 63_37_0_160 means that the Length of the Packet is 63, the Length of the Header is 37, the Length of MS Data is 0, and the Length of Service Data is 160.

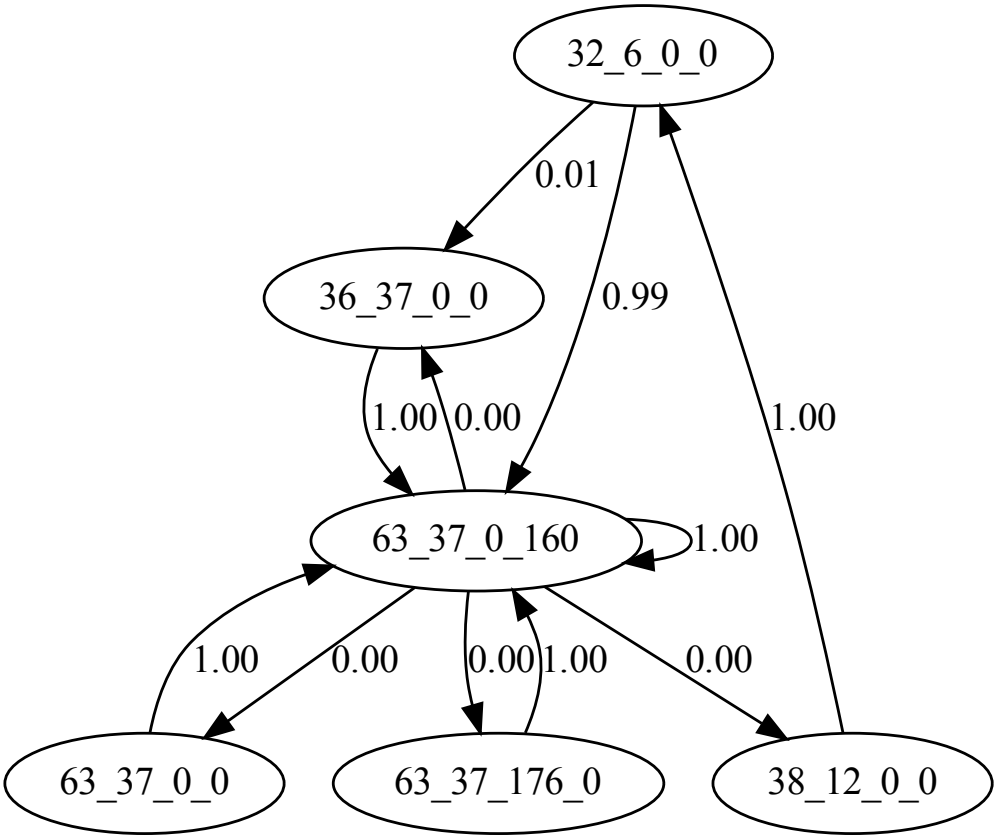


Figure 4.1: State Transition Model for Packet Length

As shown in Figure 4.1, the transition probability is heavily concentrated on specific states, particularly 63_37_0_160. Table 4.1 provides a numerical representation of these transitions, detailing the probabilities of state changes.

Table 4.1: Transition Probability Matrix for Packet Length

	(32, 6, 0, 0)	(36, 37, 0, 0)	(38, 12, 0, 0)	(63, 37, 0, 0)	(63, 37, 0, 160)	(63, 37, 176, 0)
(32, 6, 0, 0)	0.000000	0.010526	0.000000	0.000000	0.989474	0.000000
(36, 37, 0, 0)	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000
(38, 12, 0, 0)	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
(63, 37, 0, 0)	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000
(63, 37, 0, 160)	0.000000	0.000420	0.003986	0.000126	0.995384	0.000084
(63, 37, 176, 0)	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000

4.1.2 Chanel

SmartTag (nearby) always used one of the 37, 38, or 39 channels. There were no cases where multiple channels were used at the same time or no channels were used.

Figure 4.2 shows the channel state transition diagram. CH 37 was the most used. The probability of transitioning to CH 37 from either CH 38 or CH 39 is over 60%. The probability of transitioning from CH 37 to CH 37 again is over 50%. CH 38 is the next most frequently used, followed by CH 39.

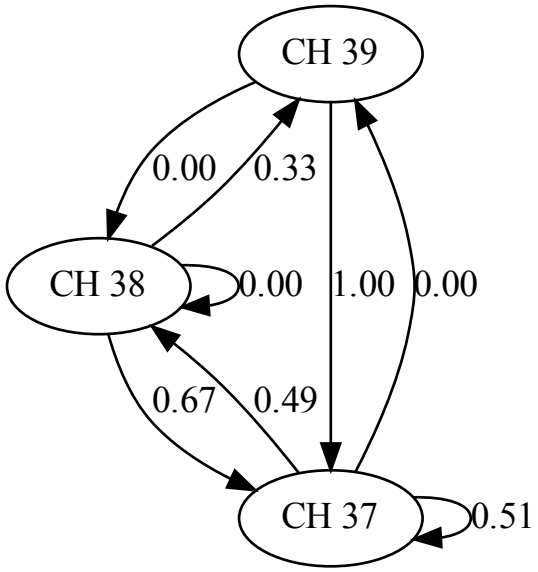


Figure 4.2: State Transition Model for Channel

The numerical transition probabilities corresponding to this state transition model are presented in Table 4.2. This table quantitatively describes the likelihood of moving between different channels. It confirms that CH 37 is the dominant state, with a self-transition probability of approximately 50

Table 4.2: Transition Probability Matrix for Channel

	CH 39	CH 38	CH 37
CH 39	0.000000	0.000419	0.999581
CH 38	0.333054	0.001398	0.665549
CH 37	0.000069	0.492758	0.507173

4.1.3 Advertising Data Type

SmartTag (nearby) used three main types of Advertising Data: AD Flags, AD Service Data - 16 bit UUID, and AD 16-bit Service Class UUIDs (incomplete). Advertising Data other than these were also used, but very rarely (5 times out of 24,038). Therefore, it is thought that Advertising Data other than AD Flags, AD Service Data - 16 bit UUIDs, and AD 16-bit Service Class UUIDs (incomplete) originated from Galaxy, but there is no proof of this. Therefore, the synthetic data was created, including Advertising Data other than these three.

The state transition diagram of Advertising Data is shown in Figure 4.3, in which AD Flags, AD Service Data - 16 bit UUID, and AD 16-bit Service Class UUIDs (incomplete) are used simultaneously in the majority of the data.

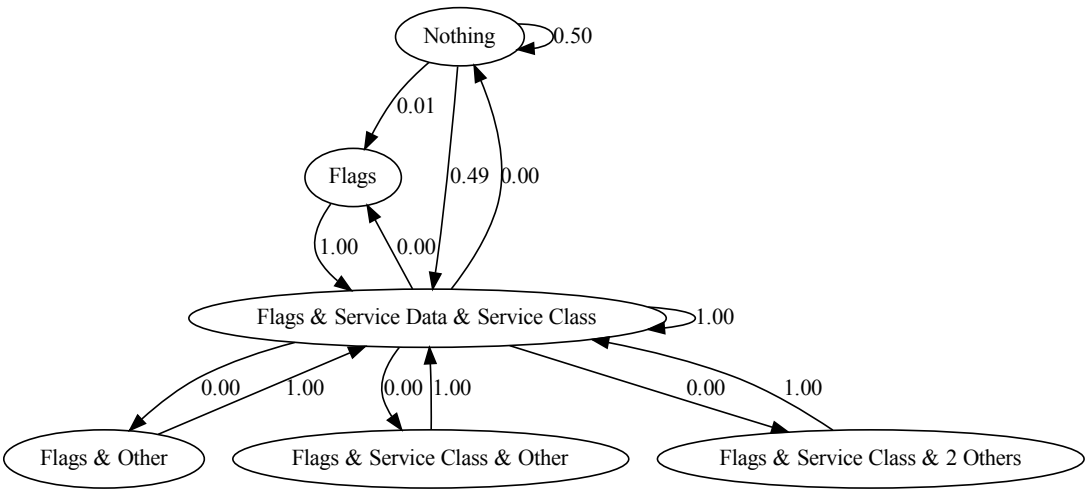


Figure 4.3: State Transition Model for Advertising Data Type

The numerical transition probabilities associated with this state transition model are provided in Table 4.3. This table presents the probability of transitioning between different advertising data types, reinforcing the observation that the combination of AD Flags, AD Service Data - 16 bit UUID, and AD 16-bit Service Class UUIDs (incomplete) is highly

prevalent. The table also highlights the rarity of other advertising data combinations, as indicated by their low transition probabilities.

Table 4.3: Transition Probability Matrix for Advertising Data Type

	Nothing	Flags	Flags & Other	Flags & Service Class & Other	Flags & Service Class & 2 Others	Flags & Service Data & Service Class
Nothing	0.500000	0.005263	0.000000	0.000000	0.000000	0.494737
Flags	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000
Flags & Other	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000
Flags & Service Class & Other	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000
Flags & Service Class & 2 Others	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000
Flags & Service Data & Service Class	0.003986	0.000420	0.000042	0.000126	0.000042	0.995384

4.1.4 UUID Type

SmartTag (nearby) often stored two UUIDs indicating Samsung, such as ”Samsung Electronics Co., Ltd.,Samsung Electronics Co., Ltd.” This is shown as 2 Samsung in Figure 4.4. There was also data in which only one UUID was stored, such as “Samsung Electronics Co. However, this was very rare, occurring only once out of 24,038 cases. There was also data containing UUIDs other than Samsung only four times out of 24,038. This data is represented as “Other” in Figure 4.4. It is puzzling that non-Samsung UUIDs are included in the data for SmartTag, which is a Samsung product. However, the synthetic data was created, including the data represented as ”Other” since there is no clear cause for this.

Figure 4.4 shows the state transition model for UUID types. It illustrates the different transitions between cases where no UUID is stored, one Samsung UUID is stored, two Samsung UUIDs are stored, or a mix of Samsung and other UUIDs is present.

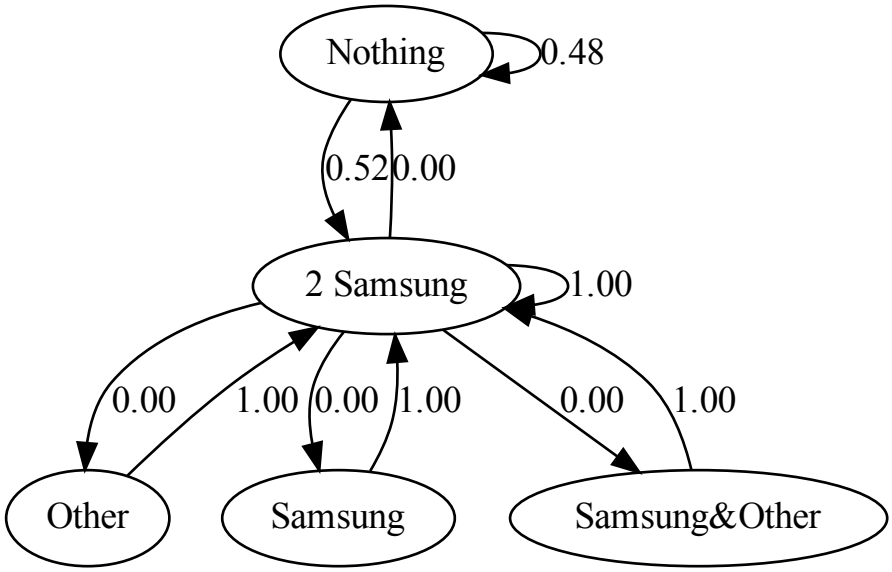


Figure 4.4: State Transition Model for UUID Type

Table 4.4 provides the transition probabilities corresponding to this state transition model. It quantifies the likelihood of transitioning between different UUID storage states. The table confirms that "2 Samsung" is the dominant state. Additionally, transitions from "Nothing" tend to lead to "2 Samsung" with a probability of approximately 52%, reflecting the prevalence of this configuration in the dataset.

Table 4.4: Transition Probability Matrix for UUID Type

	Nothing	Other	Samsung	Samsung & Other	2 Samsung
Nothing	0.475248	0.000000	0.000000	0.000000	0.524752
Other	0.000000	0.000000	0.000000	0.000000	1.000000
Samsung	0.000000	0.000000	0.000000	0.000000	1.000000
Samsung&Other	0.000000	0.000000	0.000000	0.000000	1.000000
2 Samsung	0.004448	0.000084	0.000084	0.000084	0.995300

4.1.5 PDU Type

Most of the SmartTag (nearby) PDU types were ADV_IND. It is a PDU type that indicates that the BLE device is advertising and ready to accept connection requests from the central. Therefore, it is reasonable for ADV_IND to be the most commonly used. Alternatively, there were PDU types other than ADV_IND. It is indicated as "Other" in Figure

4.5. The majority of those represented as ”Other” were considered to be SCAN_REQ, and it is usually used as packets in which the central asks for additional information on the peripheral. Thus, these packets would be expected to have been sent from the Galaxy which was placed in a Faraday cage along with the SmartTag. However, there were other PDU types classified as “Other.” Furthermore, it could not be confirmed that SCAN_REQ originated from the Galaxy device in the Faraday cage. Consequently, the synthetic data was created with these entries included rather than excluding them.

Figure 4.5 illustrates the state transition model for PDU types, showing the transitions between ADV_IND and other PDU types. The most common state is ADV_IND, with occasional transitions to and from ”Other” PDU types.

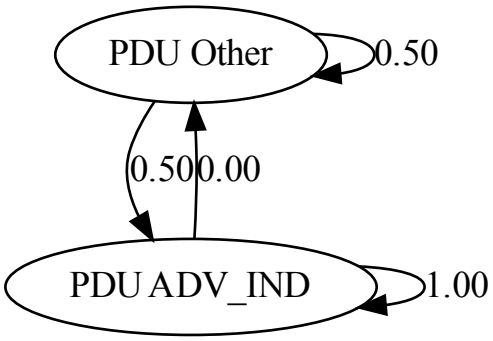


Figure 4.5: State Transition Model for PDU type

The transition probabilities corresponding to this model are shown in Table 4.5. The table quantitatively confirms that ADV_IND is the predominant PDU type, with a high self-transition probability of approximately 99.6%. In contrast, ”Other” PDU types have an equal probability of transitioning back to themselves or switching to ADV_IND, each occurring 50% of the time. These probabilities reflect the observed data distribution and indicate that non-ADV_IND PDU types occur infrequently.

Table 4.5: Transition Probability Matrix for PDU Type

	Other	ADV_IND
Other	0.500000	0.500000
ADV_IND	0.003984	0.996016

4.1.6 SmartTag Type

SmartTags in the nearby state basically use SmartTag type 5. Given this characteristic, it is reasonable that SmartTag type 5 is the most likely state for transitions. On the other

hand, data missing SmartTag type or with types other than SmartTag type 5 were also observed. The source of these data is thought to be Galaxy, which was placed together in the Faraday cage, but since the exact reason is unknown, data with SmartTag types other than 5 were also used to generate synthetic data.

Figure 4.6 illustrates the state transition model for SmartTag types, showing the transitions between cases where no SmartTag type is recorded ("Nothing"), the standard SmartTag type 5 ("ST 5"), and other types ("ST Other"). The most common state is ST 5, with occasional transitions from "Nothing" and "Other" to ST 5.

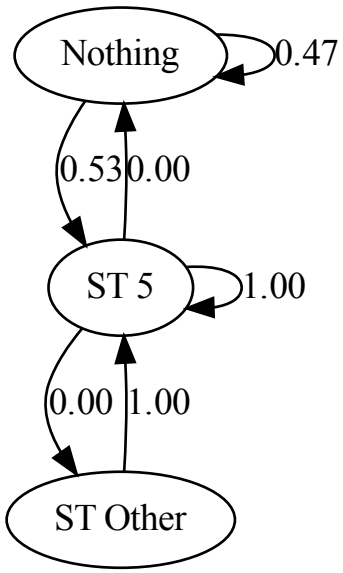


Figure 4.6: State Transition Model for SmartTag Type

The transition probabilities for this model are provided in Table 4.6. The table confirms that SmartTag type 5 is the dominant state, with a high self-transition probability of approximately 99.5%. Transitions from "Nothing" to ST 5 occur about 53% of the time, further indicating that SmartTag type 5 is the most prevalent configuration. On the other hand, SmartTag types classified as "Other" are rare but transition exclusively into ST 5.

Table 4.6: Transition Probability Matrix for SmartTag Type

	Nothing	Other	ST 5
Nothing	0.466019	0.000000	0.533981
Other	0.000000	0.000000	1.000000
ST 5	0.004616	0.000042	0.995342

4.2 Evaluation of Synthetic Data

To evaluate the synthetic data, a comparison of the structure of the data and a comparison of the model with and without synthetic data were conducted. In comparing data structures, the analysis focused on whether structural differences existed between the original data and the synthetic data generated by the Markov model. Three models were evaluated for model comparison. The first model used only the original data as training data. The second model was trained with the original and synthetic data created by Stefan. The third model used the original data and the synthetic data generated by the Markov model, all applied to the same real data. The performance results of the models were compared.

4.2.1 Comparison of Data Structures

Comparisons of data structures were made for each state in which the Markov model was applied. Specifically, comparisons were made between original and synthetic data for Length, Channel, Advertising Data type, UUID type, PDU type, and SmartTag type. The ratio of each data to the total was compared for each item.

Length (Packet, Header, MS Data, Service Data)

Figure 4.7 shows the proportions of different length combinations (Packet, Header, MS Data, and Service Data) between the original dataset and the synthetic dataset. The structure of the data is the same for both the original data and the composite data, with the majority of the combinations (63 _37_0_160) having a Packet length of 63, Header length of 37, MS data length of 0, and Service data length of 160, and only a few other combinations.

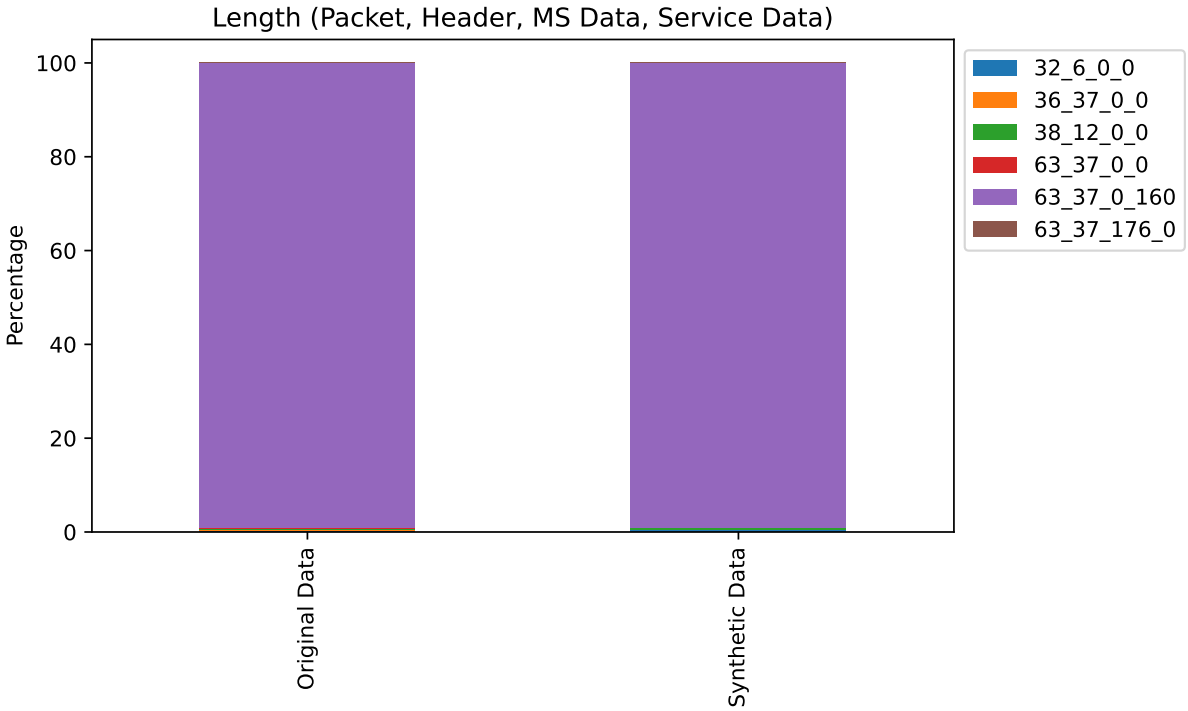


Figure 4.7: Comparison of Data Structures: Length

Table 4.7 provides the numerical breakdown of these proportions, reinforcing the findings from Figure 4.7. The values confirm that the most frequently occurring length combination (63_37_0_160) is overwhelmingly dominant in both datasets.

Table 4.7: Comparison of Proportions by Length

	32_6_0_0	36_37_0_0	38_12_0_0	63_37_0_0	63_37_0_160	63_37_176_0
Original Data	0.395208	0.045761	0.395208	0.012480	99.143024	0.008320
Synthetic Data	0.403333	0.049167	0.403167	0.007167	99.131000	0.006167

Channel

Figure 4.8 shows the proportion of different channels (CH 37, CH 38, and CH 39) in the original and synthetic datasets. No significant structural change concerning channels exists between the original and synthetic data. CH37 is the most common in both cases, followed by 38 and 39.

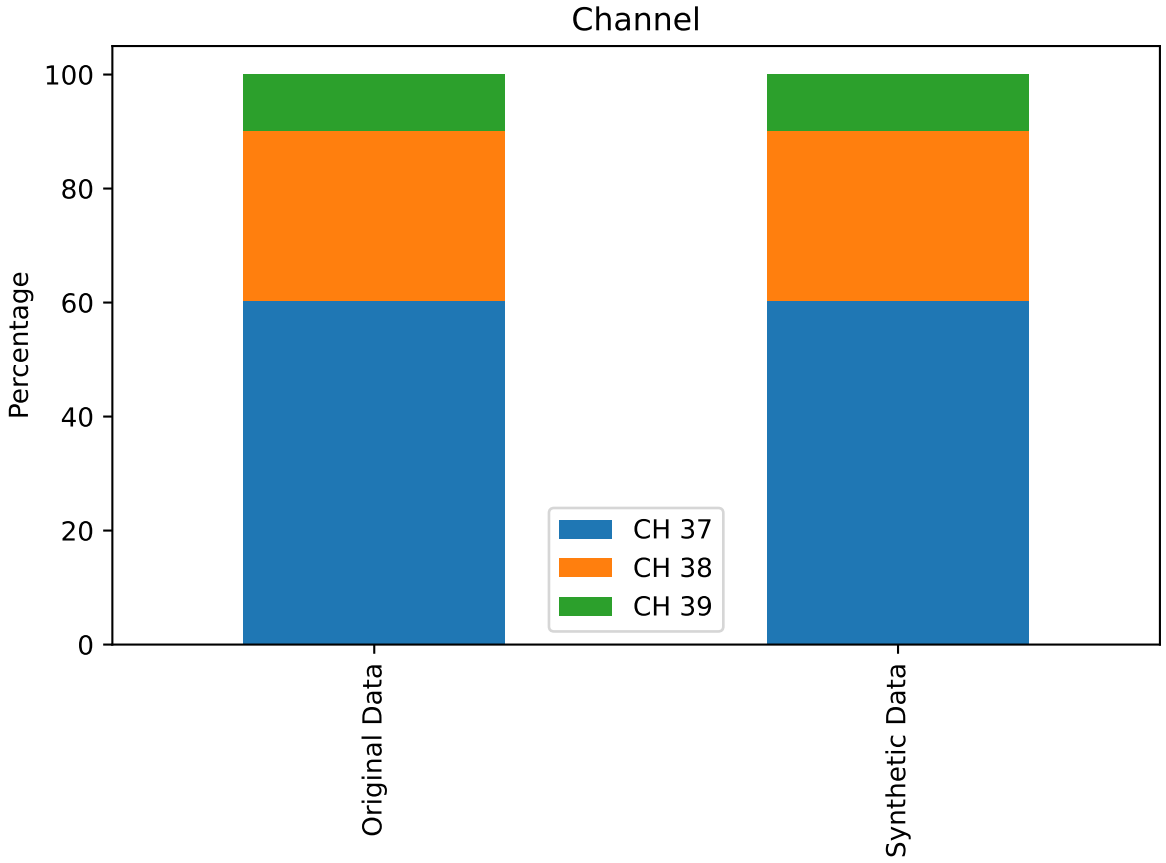


Figure 4.8: Comparison of Data Structures: Channel

Table 4.8 illustrates a numerical comparison of these proportions, confirming the findings from Figure 4.8. The values indicate that the relative distribution of channels remains nearly identical between the original and synthetic datasets, with only minor variations in CH 38 and CH 39.

Table 4.8: Comparison of Proportions by Length

	CH 37	CH 38	CH 39
Original Data	60.316998	29.765371	9.917630
Synthetic Data	60.334167	29.711667	9.954167

Advertising Data Type

Figure 4.9 provides the distribution of different advertising data types in both the original and synthetic datasets. There is no significant structural change in the Advertising data type between the original data and the synthetic data. Flags, Service Data, and Service Class Advertisement Data are often used simultaneously.

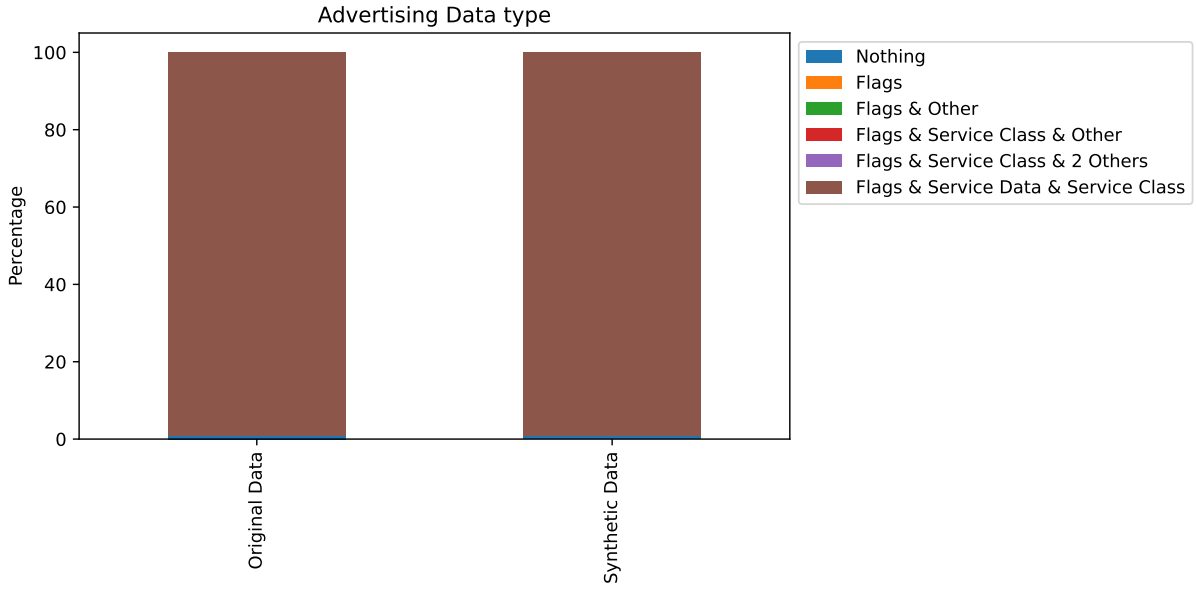


Figure 4.9: Comparison of Data Structures: Advertising Data Type

Table 4.9 gives a numerical comparison of these proportions, reinforcing the findings from Figure 4.9. The results confirm that the most frequently occurring advertising data type remains consistent across both datasets, with minor variations in the lower-frequency categories.

Table 4.9: Comparison of Proportions by Advertising Data Type

	Nothing	Flags	Flags & Other	Flags & Service Class & Other	Flags & Service Class & 2 Others	Flags & Service Data & Service Class
Original Data	0.790415	0.045761	0.004160	0.012480	0.004160	99.143024
Synthetic Data	0.821167	0.047167	0.002167	0.010167	0.001167	99.118167

UUID Type

Figure 4.10 shows the distribution of UUID types in both the original and synthetic datasets. There is no significant structural change in UUID type between the original and synthetic data. In both cases, UUIDs in which the name Samsung appears twice are primarily used, such as ”Samsung Electronics Co., Ltd.,Samsung Electronics Co., Ltd.”

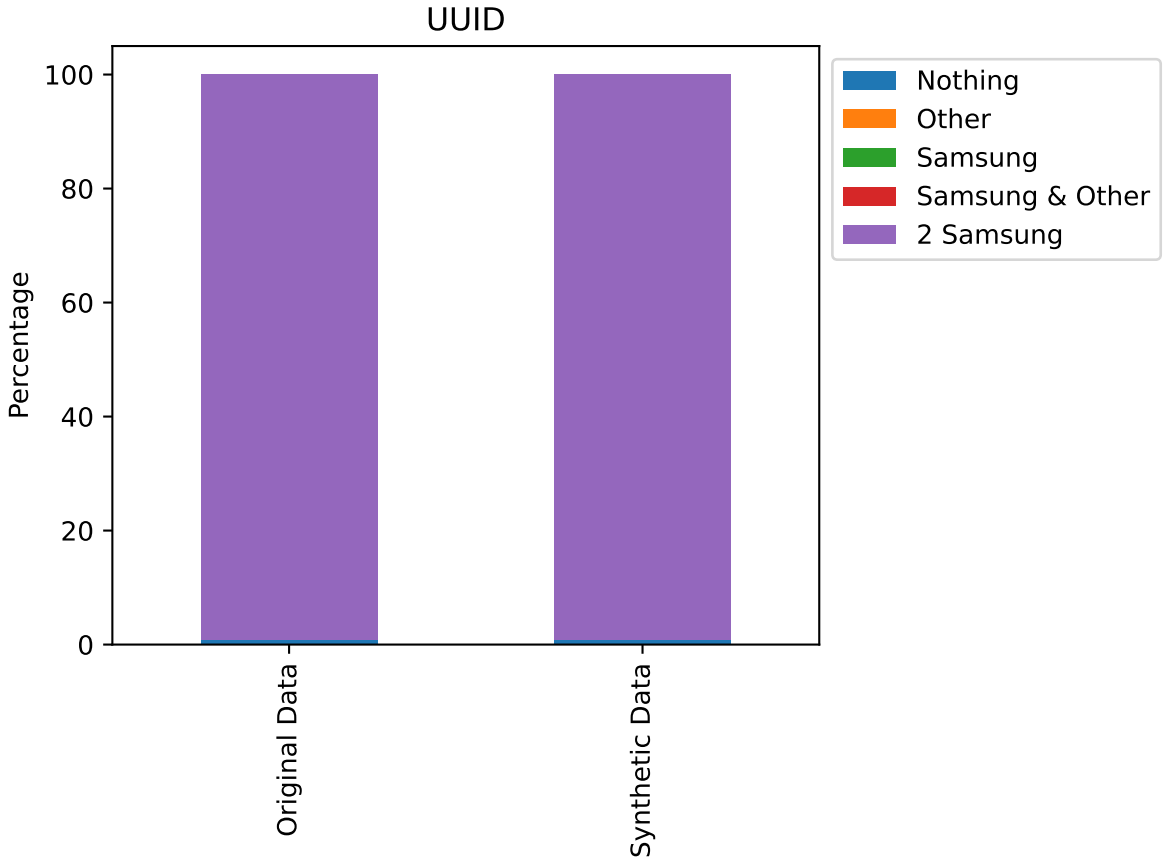


Figure 4.10: Comparison of Data Structures: UUID Type

Table 4.10 provides the numerical breakdown of these proportions, confirming the findings from Figure 4.10. The values indicate that ”2 Samsung” is the dominant UUID type, with only minor variations in the lower-frequency categories between the original and synthetic datasets.

Table 4.10: Comparison of Proportions by UUID Type

	Nothing	Other	Samsung	Samsung & Other	2 Samsung
Original Data	0.840336	0.008320	0.008320	0.008320	99.134703
Synthetic Data	0.847667	0.008500	0.006833	0.005167	99.131833

PDU Type

Figure 4.11 shows the distribution of PDU types in both the original and synthetic datasets. There is no significant structural change in PDU type between the original and synthetic data. In both cases, ADV_IND accounts for more than 99% of the data.

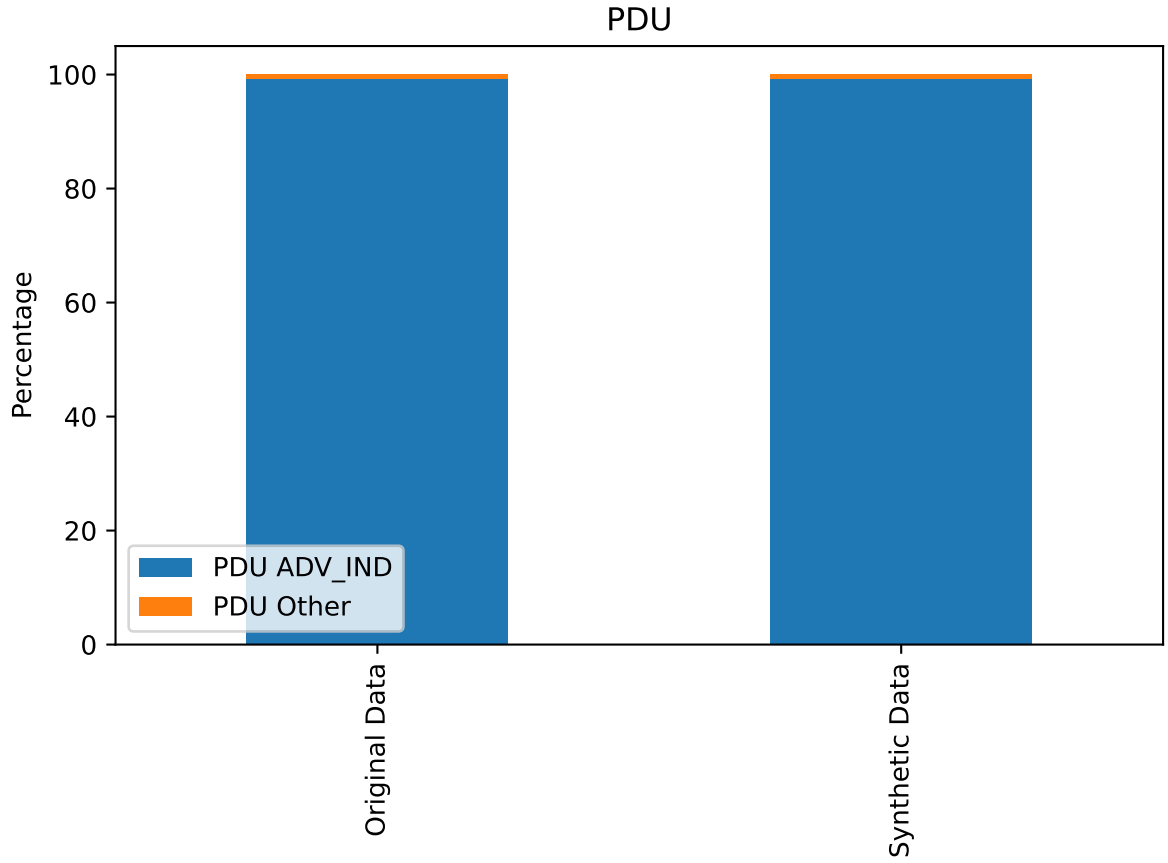


Figure 4.11: Comparison of Data Structures: PDU Type

Table 4.11 provides the numerical breakdown of these proportions, confirming the observations from Figure 4.11. The values show that PDU ADV_IND consistently accounts for more than 99% of the data in both datasets, with only slight variations in PDU Other.

Table 4.11: Comparison of Proportions by PDU Type

	PDU ADV_IND	PDU Other
Original Data	99.209585	0.790415
Synthetic Data	99.219500	0.780500

SmartTag Type

Figure 4.12 illustrates the distribution of SmartTag types in both the original and synthetic datasets. There is no significant structural change in SmartTag type between the original and synthetic data. In both cases, SmartTag type 5 accounts for more than 99% of the data, with some data having no SmartTag type.

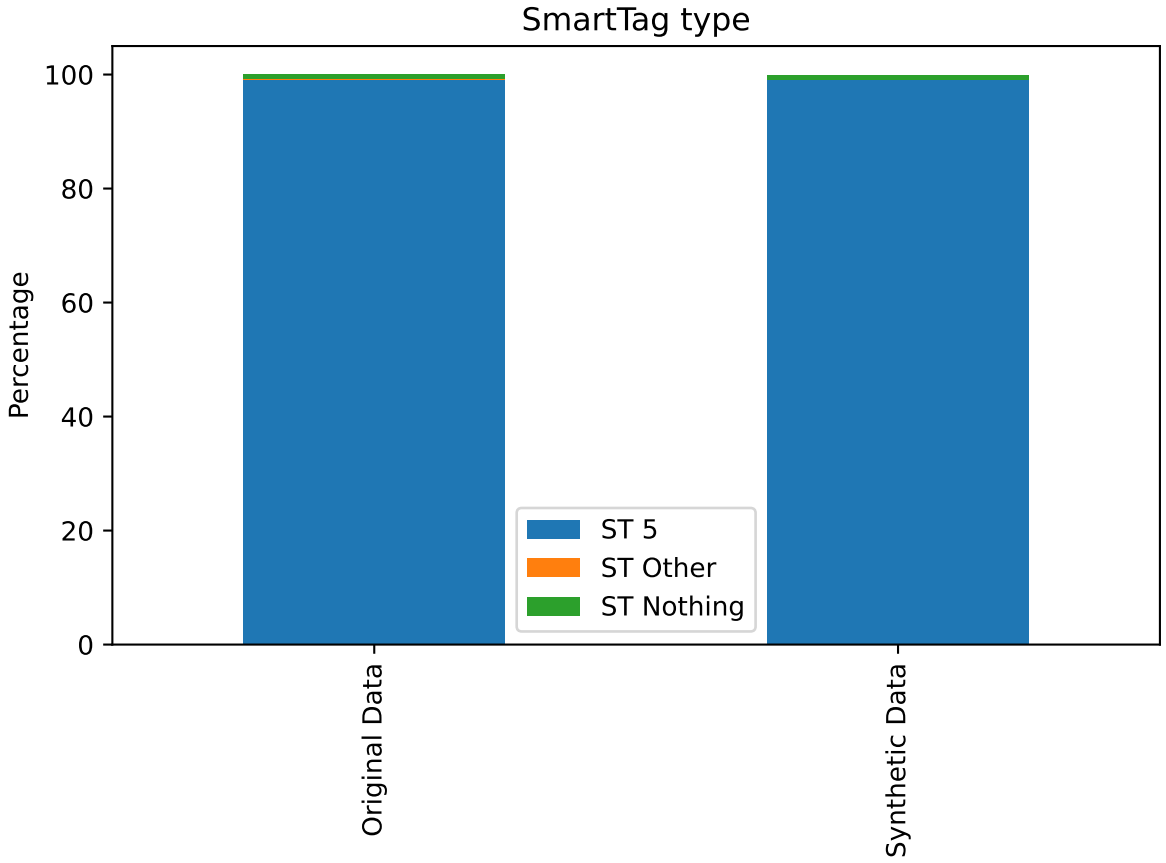


Figure 4.12: Comparison of Data Structures: SmartTag Type

Table 4.12 provides the numerical breakdown of these proportions, confirming the observations from Figure 4.12. The values show that ST 5 consistently accounts for more than 99% of the data in both datasets, with only minor differences in the less frequent categories.

Table 4.12: Comparison of Proportions by SmartTag Type

	ST 5	ST Other	ST Nothing
Original Data	99.138863	0.004160	0.856976
Synthetic Data	99.134167	0.003167	0.862667

Summary

No significant structural changes were found when comparing the synthetic data created in this study with the original data. This study generated the same amount of synthetic data using a Markov model, where all possible states for each item were used as initial values. Theoretically, there was a possibility that this method could cause changes in the structure, but no such changes were observed in this experiment.

4.2.2 Comparison of Confidence Levels

The neural network was used to examine the impact of synthetic data on the model. The model was trained in three patterns—the first pattern used only the original data as training data. The second pattern used the original data combined with synthetic data created by Stefan, which will be discussed in the following subsection. The third pattern used the original and synthetic data generated using the Markov model as training data. The Confusion Matrix of these models on the test data and the confidence levels of these models on the real data were examined.

Synthetic Data Generated by Stefan

Stefan created synthetic data for the real data, not those collected in the Faraday cage. The real data, *Banhof_V1* and *Bahnhof_V2* [38], were collected on the same day at the same location at Zürich central station. *Banhof_V1* was collected in 10 minutes, while *Banhof_V2* was collected over 35 minutes. Although there are differences in data size, the data have the same characteristics. Stefan created data for *Bahnhof_V1*. Since *Bahnhof_V1* is real data, it is not labeled. Therefore, he first applied a machine learning model created using data collected in the Faraday cage as training data to *Bahnhof_V1* to obtain the predictive labels for each device and the probability of that prediction. Then, based on the assumption that packet data with high prediction probabilities were correctly classified, the same labeling was applied to packet data with the same source address as the packet data with high prediction probabilities. This is because packets with essentially the same source address are considered to have been produced by the same device.

Data labeled as SmartTag (nearby) was extracted from the real data labeled this way. Then, only the data whose UUID was not Samsung was further extracted from the extracted data. In other words, the data was classified as "SmartTag (nearby)" but did not originate from Samsung devices, which were considered misclassified and extracted. Thus, data classified as SmartTag (nearby) but with non-Samsung UUIDs were grouped by UUID, and 50 samples were randomly extracted from each group. The extracted data were duplicated 4000 times to create the final synthetic data. At this time, the amount of data created was 600,000 and the amount of data after preprocessing was 600,000. This synthetic data was labeled "other Device" and used for model training.

In summary, Stefan created synthetic data by extracting data from the real dataset misclassified as "SmartTag (nearby)" but did not originate from Samsung devices and then duplicated these entries.

Model

The neural network created by Stefan [9] was used to evaluate the synthetic data. Stefan created a neural network, a Self-Training Classifier, and a decision tree. All of them were highly accurate. Although the decision tree was attractive because it was easy to

interpret, it could not quantify the confidence level of the prediction. Therefore, comparing the synthetic data with the original data was considered insufficient.

On the other hand, the Self-Training Classifier can quantitatively express the reliability of prediction using the Softmax function. It is also highly robust and more suitable than neural networks for actual operation. However, the Self-Training classifier assigns provisional labels to unlabeled data in self-learning. There is a risk that this provisional label may contain errors due to differences in the distribution of synthetic and real data, thus compromising the purity of the evaluation results. The neural network also allows prediction accuracy to be quantitatively expressed simply using the Softmax function. In addition, the neural network effectively learns nonlinear patterns and is superior in capturing minute differences between synthetic and real data. Hence, the neural network was employed to evaluate the synthetic data.

Stefan created the neural network used. This neural network is a basic implementation of the MLP classifier from the scikit-learn library. This neural network has 100 neurons in one hidden layer and uses ReLU as the activation function; the second, all-joined layer uses the Softmax function to output probabilities for each class [9].

The number of training data was 240,380 when training with only the original data without synthetic data, and when using synthetic data created by Stefan in addition to the original data. When synthetic data created by the Markov model was used in addition to the original data, the number of data was 328,290. An equal number of samples were used as training data for each label, based on the number of samples in the label with the fewest data points after aggregating the data by label. For example, in the original dataset, there was data for each label as shown in Table 4.13. The same number of data for other labels was randomly extracted to match the number of data for SmartTag (nearby), which had the least number of data, and used as training data (Table 4.14). Since the Markov model was used to generate 600,000 samples for the ”SmartTag (nearby)” label, the label with the fewest samples became ”SmartTag (lost).” As a result, the number of trained data increased for the model trained by original data and synthetic data produced by the Markov model (Table 4.15).

Table 4.13: Number of Samples in the Original Dataset

Label	**Number of Data**
iDevice	5,775,063
other Device	1,776,914
FindMy Tracker (unpaired)	304,646
FindMy Tracker (lost)	191,815
FindMy Tracker (nearby)	190,271
iDevice FindMy online	189,946
iDevice FindMy offline	182,397
Tile (lost)	66,129
SmartTag (lost)	32,829
SmartTag (nearby)	24,038

Table 4.14: Number of Samples in the Training Dataset

Label	**Number of Data**
SmartTag (lost)	24,038
iDevice	24,038
SmartTag (nearby)	24,038
other Device	24,038
Tile (lost)	24,038
FindMy Tracker (unpaired)	24,038
FindMy Tracker (lost)	24,038
iDevice FindMy offline	24,038
iDevice FindMy online	24,038
FindMy Tracker (nearby)	24,038

Table 4.15: Number of Samples in the Training Dataset Including Synthetic Data Generated by the Markov Model

Label	**Number of Data**
FindMy Tracker (unpaired)	32,829
iDevice FindMy online	32,829
FindMy Tracker (nearby)	32,829
iDevice FindMy offline	32,829
SmartTag (nearby)	32,829
Tile (lost)	32,829
iDevice	32,829
FindMy Tracker (lost)	32,829
other Device	32,829
SmartTag (lost)	32,829

For convenience, the model trained on only original data is called Model-O, the model trained with synthetic data created by Stefan is called Model-S, and the model trained with synthetic data generated by the Markov model is called Model-M in this paper.

Confusion Matrix

Figure 4.13 shows that a high classification level has already been achieved for the test data without using the synthetic data. FindMy Tracker (nearby) is sometimes incorrectly classified as (lost), but this is because FindMy Tracker sometimes switches from nearby to lost states automatically. Even with this state, FindMy Tracker has a 98.3% accuracy rate, which is a very high level.

In addition, Model-O and Model-S showed little difference in accuracy for the test data (Figure 4.14).

On the other hand, Model-O shows an increase in accuracy for SmartTag (nearby) (Figure 4.15). This may be because, unlike the Model-S, the synthetic data was created based on the original, not the real data.

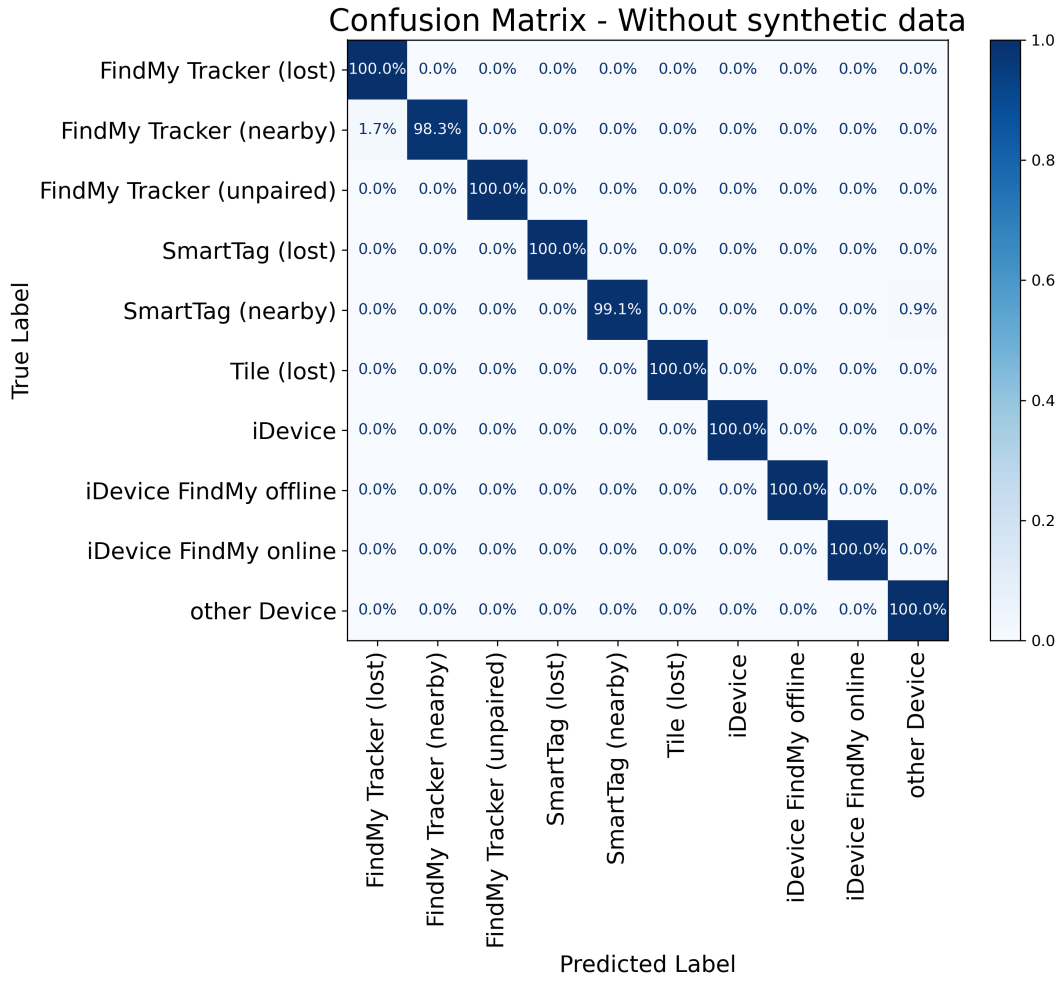


Figure 4.13: Confusion Matrix - Without Synthetic Data (Model-O)

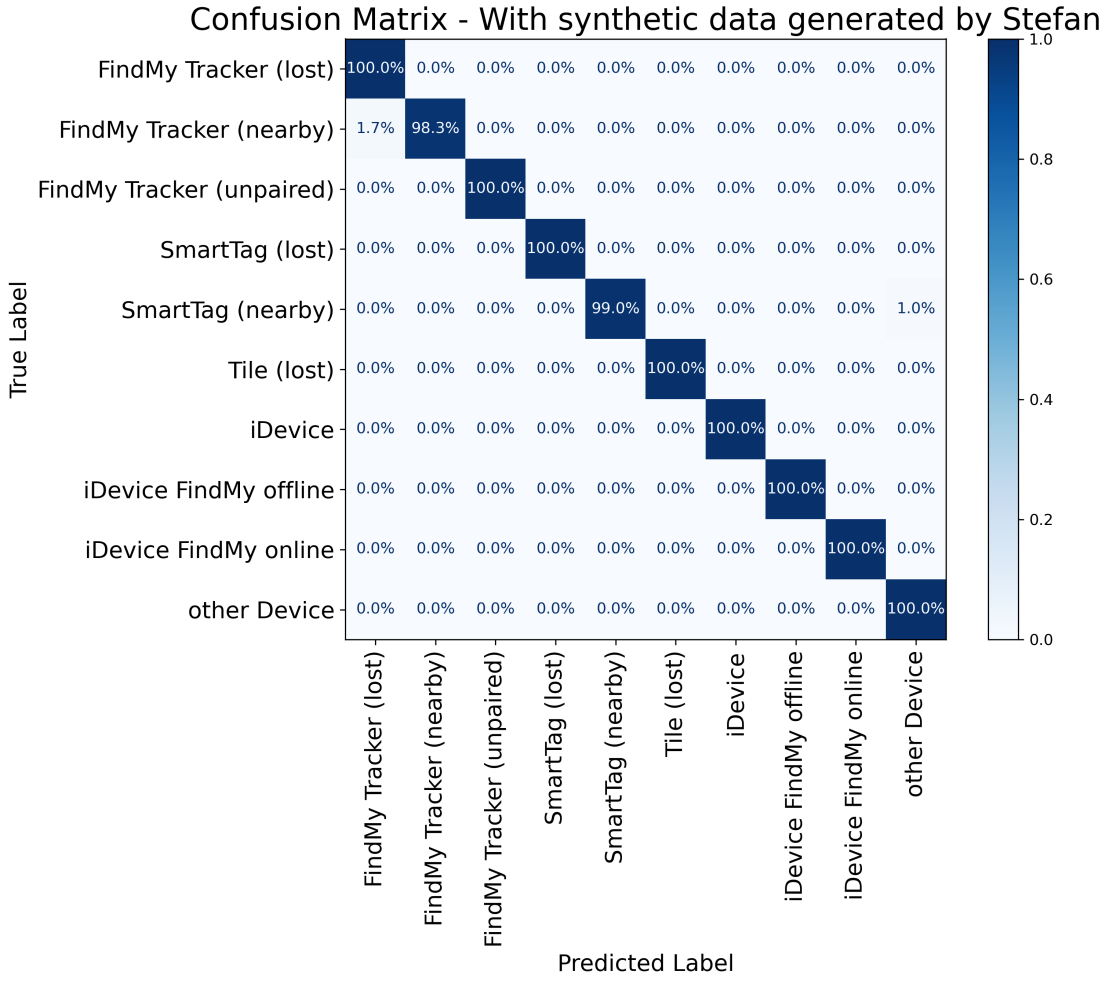


Figure 4.14: Confusion Matrix - With Synthetic Data Generated by Stefan (Model-S)

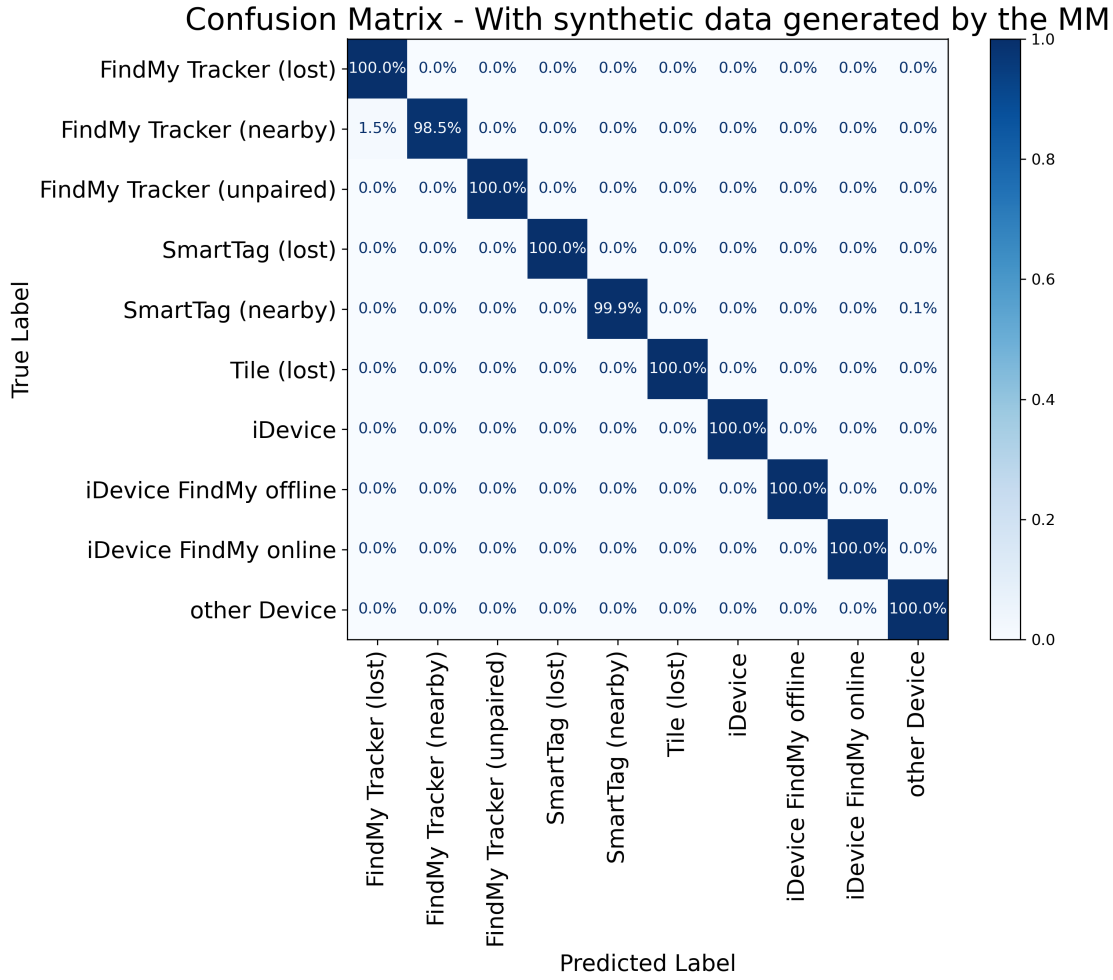


Figure 4.15: Confusion Matrix - With Synthetic Data Generated by the Markov Model (Model-O)

Considering the possibility that differences in classification accuracy may arise due to differences in data size, the model was trained on a range of training datasets from 1/64 to the full dataset and checked the accuracy against test data. As shown in Figure 4.16, the accuracy was high enough even with 1/64 of the data size, suggesting that the difference was not due to the data size but to the synthetic data.

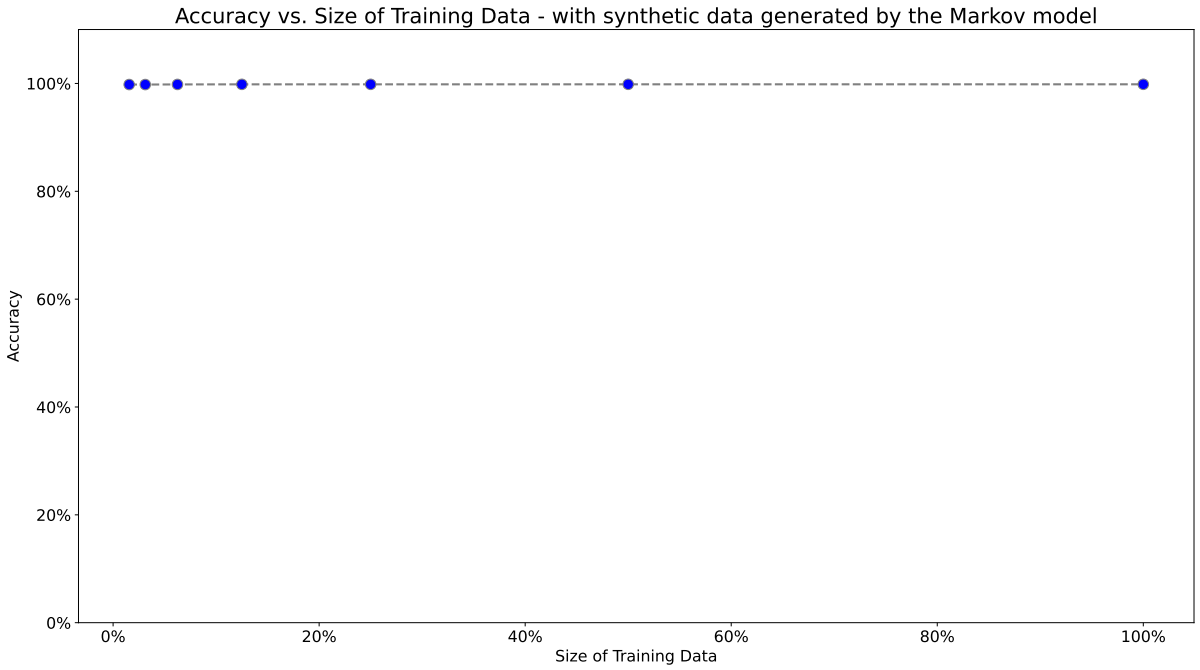


Figure 4.16: Accuracy vs. Size of Training Data - With Synthetic Data Generated by the Markov model (Model-O)

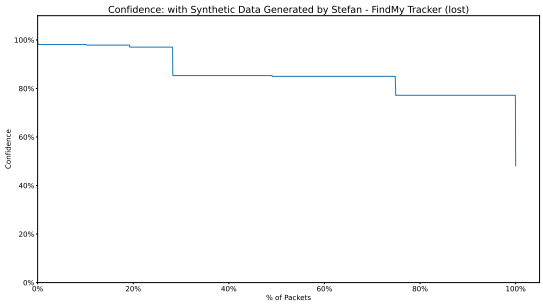
Confidence Level

Model-O, Model-S, and Model-O were each applied to real-world data (*Bahnhof_V2*). The results for Model-O are shown in Figure 3.3, Model-S in Figure 4.17, and Model-M in Figure 4.18.

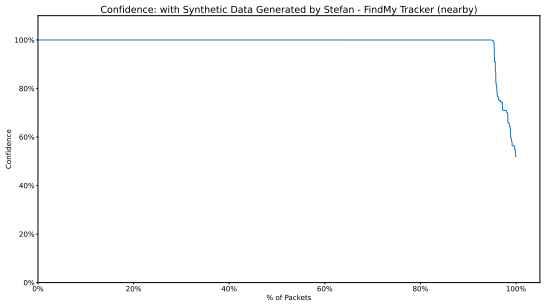
Compared to Model-O, the confidence in the SmartTag (nearby) classification is higher for Model-S and Model-M, which use synthetic data. In addition, Model-M shows higher confidence in SmartTag (nearby) than Model-S, although this is a slight difference.

Moreover, Model-M shows an overall improvement in confidence, not only for Smart-Tag (nearby). For example, Model-M has higher confidence in FindMy Tracker (lost), FindMy Tracker (nearby), and rooter Device than Model-O. However, the confidence in the classification decreased for FindMy Tracker (unpaired).

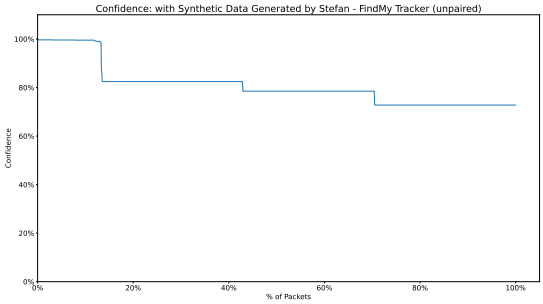
Alternatively, Model-S improved slightly in confidence in its classifications for FindMy Tracker (lost) and SmartTag (nearby). Conversely, there was little change or a slight decrease in confidence for other devices. Specifically, the confidence level of the Tile (lost) classification decreased a little. Also, the confidence level for SmartTag (lost) dropped significantly.



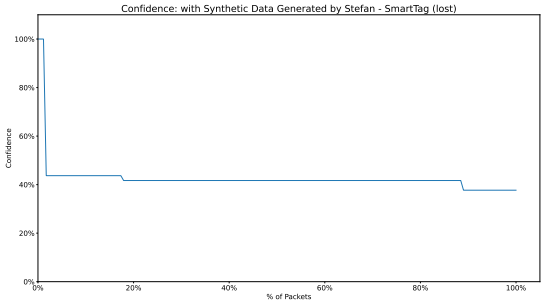
(a) Confidence - FindMy Tracker (lost)



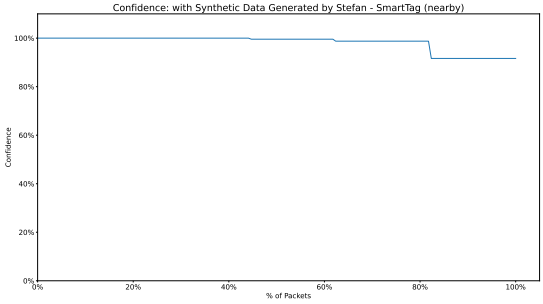
(b) Confidence - FindMy Tracker (nearby)



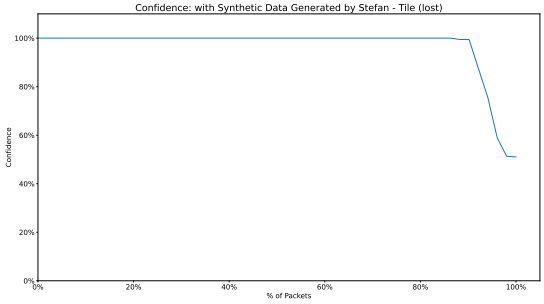
(c) Confidence - FindMy Tracker (unpaired)



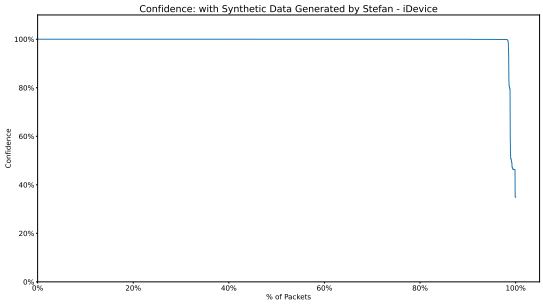
(d) Confidence - SmartTag (lost)



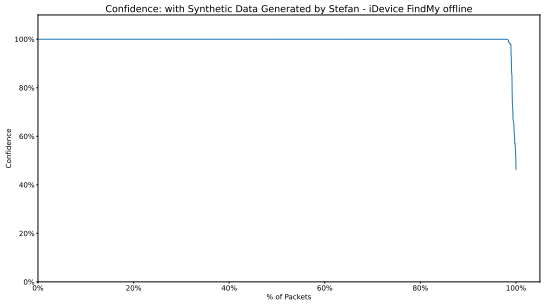
(e) Confidence - SmartTag (nearby)



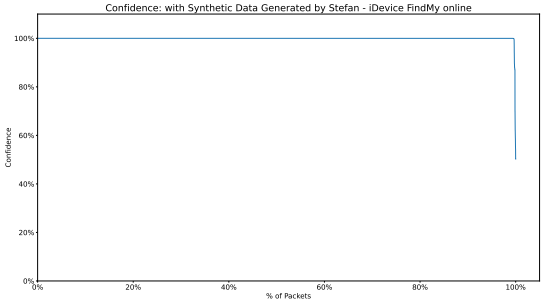
(f) Confidence - Tile (lost)



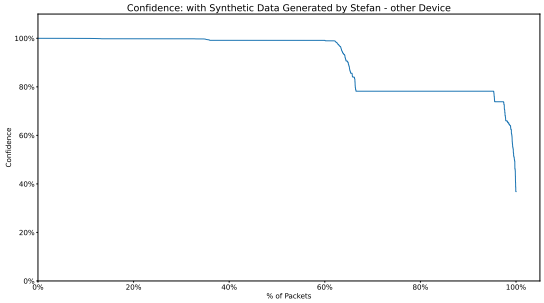
(g) Confidence - iDevice



(h) Confidence - iDevice FindMy offline

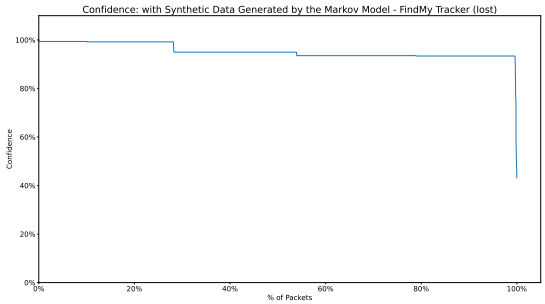


(i) Confidence - iDevice FindMy online

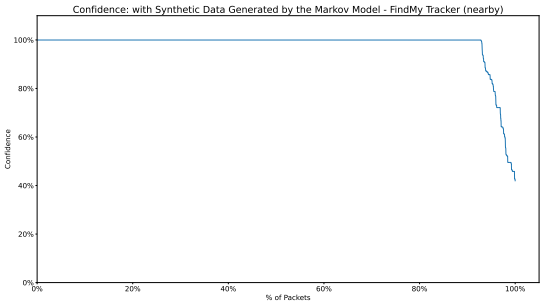


(j) Confidence - other Device

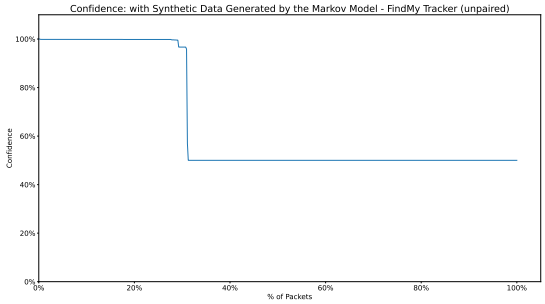
Figure 4.17: Confidence - With Synthetic Data Generated by Stefan [50]



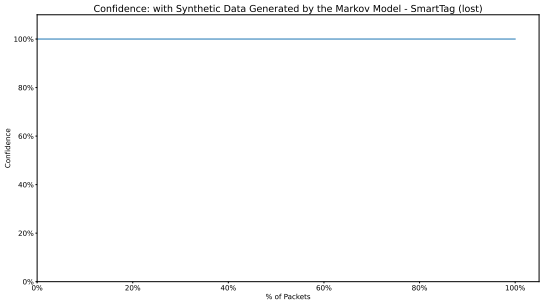
(a) Confidence - FindMy Tracker (lost)



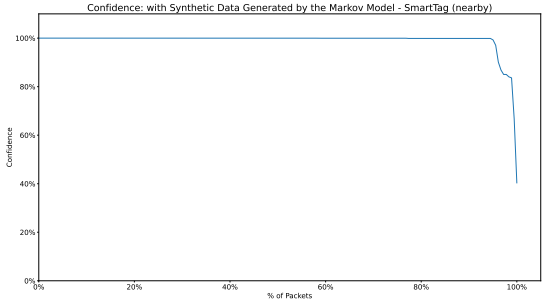
(b) Confidence - FindMy Tracker (nearby)



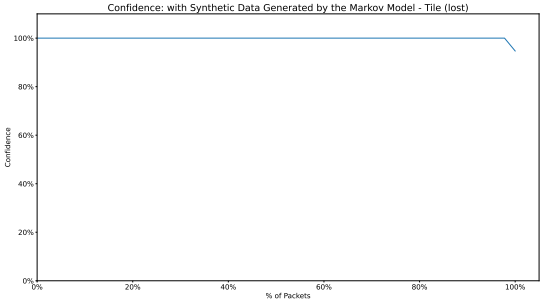
(c) Confidence - FindMy Tracker (unpaired)



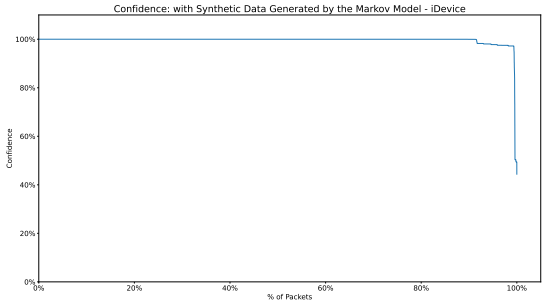
(d) Confidence - SmartTag (lost)



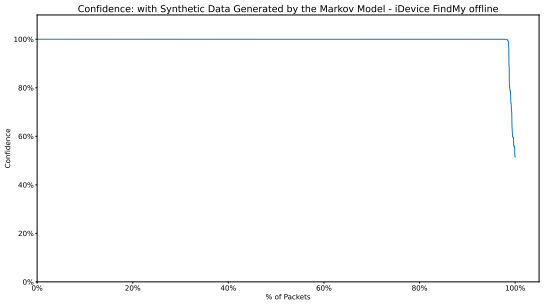
(e) Confidence - SmartTag (nearby)



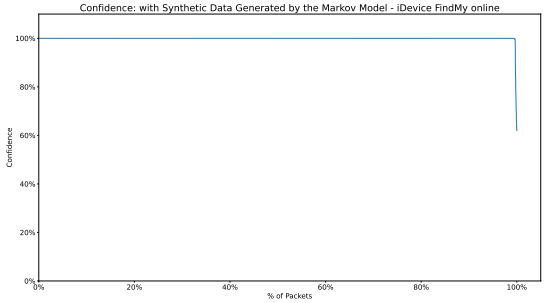
(f) Confidence - Tile (lost)



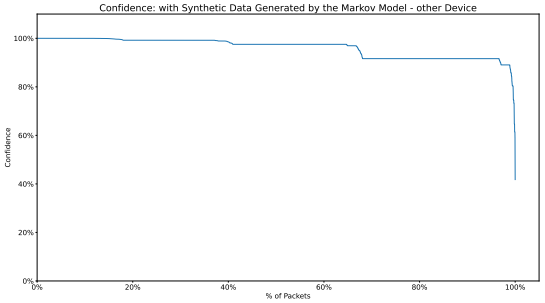
(g) Confidence - iDevice



(h) Confidence - iDevice FindMy offline



(i) Confidence - iDevice FindMy online



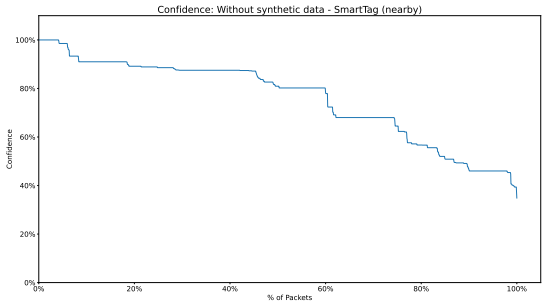
(j) Confidence - other Device

Figure 4.18: Confidence - With Synthetic Data Generated by the Markov Model

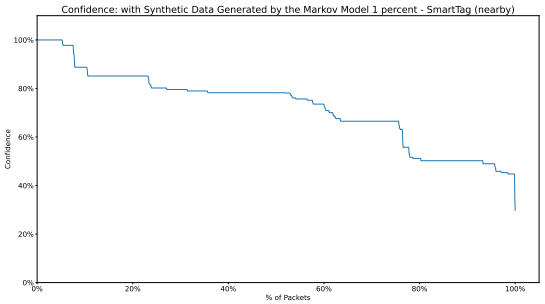
The results were examined by varying the ratio of synthetic data in the training dataset to investigate the impact of synthetic data on the model. The results are shown in Figure 4.19. The number of data used in this case was the same as in Table 4.14, 24,038 for each label. The percentage of synthetic data in the SmartTag (nearby) training data was varied from 0% to 1%, 5%, 10%, 30%, 50%, 75%, and 100%. The model's confusion matrix and the confidence level for all labels at each percentage level are included in Appendix B and C.

Up to 50%, there is no significant improvement or deterioration. On the other hand, there is an overall increase in confidence levels at the 75% and 100% percentages of synthetic data. Particularly, the confidence level is exceptionally high when the training data for SmartTag (nearby) are all synthetic.

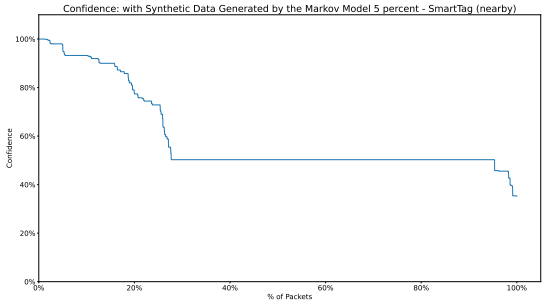
Incidentally, for devices other than SmartTag (nearby), there was no significant change in the confidence level based on the percentage of synthetic data.



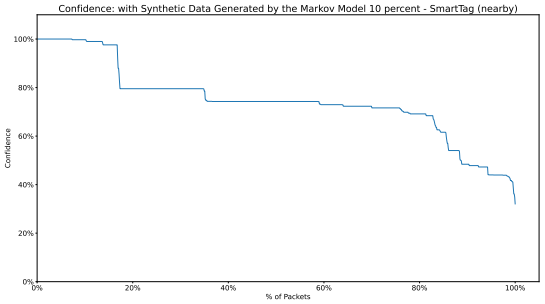
(a) 0 percent - SmartTag (nearby)



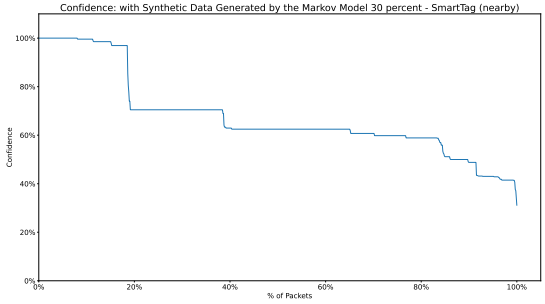
(b) 1 percent - SmartTag (nearby)



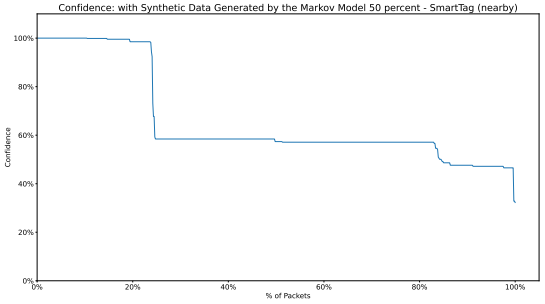
(c) 5 percent - SmartTag (nearby)



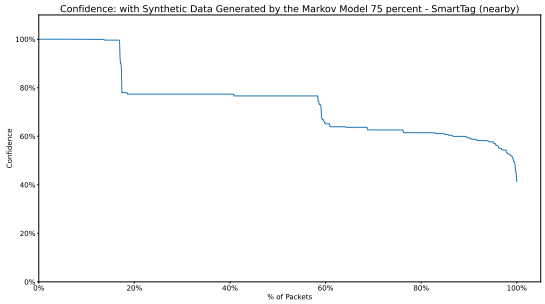
(d) 10 percent - SmartTag (nearby)



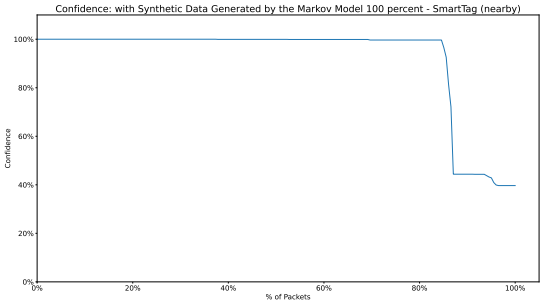
(e) 30 percent - SmartTag (nearby)



(f) 50 percent - SmartTag (nearby)



(g) 75 percent - SmartTag (nearby)



(h) 100 percent - SmartTag (nearby)

Figure 4.19: Confidence levels for SmartTag (nearby) with varying proportions of synthetic data in the training set.

4.3 Discussion on Synthetic Data Generation Using the Markov Model

As Figures 3.3, 4.17, and 4.18 show, the synthetic data generated by the Markov model increased the accuracy of the model's classification. As seen from Figure 4.16, this is not due to an increase in the amount of data but rather to the diversity of the data generated by the synthetic data. The diversity of the given data is not because each column was created individually but because the number of combinations of values in each column has increased. This is because, as discussed in the subsection 4.2.1, there was little difference in the distribution of data per column between the original and synthetic data. Therefore, the diversity in the combination of each column has expanded the range of data that can be handled and has made it possible to maintain a high level of accuracy for real data.

The fact that the synthetic data generated by the Markov model contributed to the improvement of the model can be seen in Figure 4.19. Since the confidence level increased as the percentage of synthetic data in the training data increased, it can be inferred that the synthetic data had higher diversity than the original data, contributing to the improved accuracy of the model.

On the other hand, even when synthetic data was used for all of the SmartTag (nearby) training data, the model could only produce a confidence level of about 50% for about 15% of the packets. There are two possible reasons for this. First, the diversity given by the Markov model is limited. Due to the strict format of BLE packets, the diversity provided by the Markov model, which generates synthetic data based on existing datasets, may be insufficient. Second, SmartTag (nearby) packets are similar to other labeled packets and may be challenging to classify perfectly in the first place.

Moreover, it is noted that Model-M has reduced confidence in FindMy Tracker (unpaired). One possible cause for this could be that the synthetic data from SmartTag (nearby) changed the correlations of new and existing features, potentially causing interference with the features in FindMy Tracker (unpaired). However, a clear cause cannot be determined from this research.

Compared to Stefan's synthetic data, the synthetic data generated by the Markov model contributed more to the accuracy of the model. Stefan's method labeled all packet data misclassified as SmartTag (nearby) as "other." While this increased the diversity of the data, it may have also increased noise and produced bias in the data. Specifically, the "other" class became too ambiguous, and packet data from devices that should have been labeled correctly were classified as "other." This is believed to have reduced the reliability of classification for some devices.

Moreover, Stefan's method requires collecting real data to create synthetic data. On the other hand, the Markov model method requires only the original data, not the real data, saving time and money in data collection. In addition, the Markov model itself is very simple and does not require large computational resources.

4.4 Limitations

While this study focused on SmartTag (nearby), other devices, such as Tile and AirTag, would not necessarily yield similar results. Each device has unique communication patterns and characteristics that may require different models and approaches to reproduce them accurately.

Furthermore, the Markov model is based on the simple assumption that only the current state determines the next state. Therefore, if the BLE packet data behaves non-Markovian, synthetic data using a Markov model may not be suitable. For example, the Markov model may be inadequate in cases where synthetic data must be created based on data where the battery level of the BLE device is low, the frequency of advertisements is reduced, or where packets are missing due to external noise or signal interference.

It should also be noted that the synthetic data generated in this study are not necessarily the data that could exist in reality. As shown in Figure 3.1, this study created synthetic data for each grouped column and combined them to form the final data. Therefore, there is a possibility that combinations of values may be included that are not possible. To prevent this problem, highly related columns were treated as the same group as much as possible, but it cannot be assured that this was avoided entirely.

Chapter 5

Final Considerations

This chapter provides a summary of this study and answers to the research questions. Additionally, it discusses several future directions for this research.

5.1 Summary

In this study, synthetic data was created using a Markov model with the goal of improving the accuracy of the BLE classification model. First, the suitability of the Markov model was examined for creating synthetic data for BLE packet data by comparing it with other synthetic data creation methods.

Next, the Markov model generated synthetic data for BLE packet data. In this research, synthetic data was not created for all BLE devices but rather for those in the nearby state of Samsung's SmartTag. This approach was chosen due to time constraints and to facilitate the evaluation of the usefulness of the synthetic data.

Finally, the synthetic data created was evaluated by comparing its structure and the resulting model performance. The evaluation confirmed that the synthetic data generated by the Markov model improved model accuracy. This was attributed not to an increase in the volume of data but rather to an increase in its diversity. In particular, the effect is believed to stem from an improvement in the combination of values across columns rather than an enhancement in the diversity of individual columns.

5.2 Conclusions

To conclude, this section evaluates how the three goals set forth in the introduction were achieved.

1. **Clarify the appropriateness of the Markov model for creating BLE data.**

   Markov models are well suited for time series and structured data. This is because the feature of considering the next state based on the current state is compatible with time series data. It is also suitable for structured data because it protects the structure of existing data rather than adding random noise. The Markov model is appropriate for BLE packet data because it is both time series and structured data. Therefore, the Markov model is considered suitable for generating synthetic data for BLE packets.

2. **Create synthetic data of BLE using a Markov model.**

   Synthetic data for SmartTag (nearby) in a preprocessed state was produced using the Markov model. Compared to other devices, SmartTag (nearby) data is limited, and its classification accuracy is not as high. Stefan, who conducted previous research, aimed to improve the classification accuracy of SmartTag (nearby) using another method. The synthetic data created was targeted at preprocessed data that could be directly used to train existing models and facilitate evaluation.

3. **Evaluate the synthetic data**

   The synthetic data generated by the Markov model was found to improve the model's classification accuracy. This result confirmed that this improvement in classification accuracy was not due to an increase in the amount of data but rather to an increase in the diversity of the data. On the other hand, the distribution of values for each column is not significantly different between the original data and the synthetic data, suggesting that the increase in the number of combinations of values for each column contributes to the diversity of the data and improves the generalization performance of the model. This improved the classification accuracy for real data.

The goals of this study were achieved. The synthetic data produced by the Markov model was found to be useful in improving the accuracy of the classification model for BLE packets.

5.3 Future Work

After comparing the characteristics of the BLE packet data with those of the Markov model, this study concluded that the Markov model is suitable for creating synthetic data for BLE packet data. However, the results of creating synthetic data using other methods were not confirmed. Therefore, it is considered beneficial to generate synthetic data using methods not employed in this study, such as GAN, and to compare the results with those obtained in this research. At this time, it is also worthwhile to consider a broader range of methods for generating synthetic data, such as VAE, in addition to GAN and hidden Markov models, which were compared with Markov models in this study.

It is also possible to change the data to be handled. In this study, synthetic data was created based on preprocessed data, but it is also possible to create synthetic data from

raw data. By using raw data, it may be possible to create synthetic data that includes trivial information and delicate patterns that may be lost in preprocessing.

Another possibility is to generate synthetic data for devices other than SmartTag (nearby). For example, creating synthetic data for FindMy Tracker (unpaired) allows confirmation of whether Model-M can complement devices with decreased classification reliability. Additionally, creating synthetic data for other devices verifies whether the conclusions reached in this study are universal or specific to SmartTag (nearby).

Moreover, synthetic data could be created for real data. Real data is likely to contain diversity and noise that is not available in the data collected in the Faraday cage, and this information may enhance the generalization performance of the model.

Then, in evaluating the structure of the synthetic data in this study, the data distribution per column was compared to the original data. In addition to this approach, one could also consider using indicators of data diversity and realism, such as Fréchet Inception Distance or Maximum Mean Discrepancy, for a more quantitative evaluation.

Bibliography

[1] U. Raj, "Bluetooth low energy: A comprehensive wireless technology," *International Journal of Advance Research and Innovation*, 2021.

[2] Mordor Intelligence. (2024) Smart tracker market size & share analysis - growth trends & forecasts (2024 - 2029). Mordor Intelligence. [Accessed: Jul. 23, 2024]. [Online]: https://www.mordorintelligence.com/industry-reports/smart-tracker-market

[3] ela INNOVATION. (2024) Bluetooth low energy (ble). ela INNOVATION. [Accessed: Jul. 23, 2024]. [Online]: https://elainnovation.com/en/bluetooth-low-energy/

[4] YipitData. (2024) Amazon brands analysis: Apple airtag sales accelerate; irobot outpaces peers. YipitData. [Accessed: Jul. 23, 2024]. [Online]: https://www.prnewswire.com/news-releases/amazon-brands-analysis-apple-airtag-sales-accelerate-irobot-outpaces-peers-301616318.html

[5] D. Hoffmeyer. (2024) Apples airtag macht stalking kinderleicht – hat der us-konzern die gefahr seines ■peilsenders■ unterschätzt? NZZ. [Accessed: Dec. 29, 2024]. [Online]: https://www.nzz.ch/panorama/airtag-stalking-hat-apple-die-gefahr-unterschaetzt-ld.1688765?reduced=true

[6] A. Barua, M. A. Al Alamin, M. S. Hossain, and E. Hossain, "Security and privacy threats for bluetooth low energy in iot and wearable devices: A comprehensive survey," *IEEE Open Journal of the Communications Society*, Vol. 3, pp. 251–281, 2022.

[7] N. Thornton. (2023) How does location tracking affect your privacy?: What big tech doesn't want you to know. PrivacyEnd. [Accessed: Jul. 23, 2024]. [Online]: https://www.privacyend.com/how-location-tracking-affects-privacy/#:˜:text=By%20allowing%20these%20apps%20to,such%20as%20stalking%20or%20theft.

[8] T. Yu, J. Henderson, A. Tiu, and T. Haines, "Security and privacy analysis of samsung's crowd-sourced bluetooth location tracking system," *USENIX Security Symposium*, 2024, [Accessed: Jul. 23, 2024]. [Online]: https://www.usenix.org/conference/usenixsecurity24/presentation/yu-tingfeng

[9] S. R. Saxer, "Dataset generation and feature extraction for high-traffic environment personal tracker identification," Bachelor's thesis, University of Zurich, Zurich, Switzerland, 2024.

[10] J. Briggs and C. Geeng, "Ble-doubt: Smartphone-based detection of malicious bluetooth trackers," *2022 IEEE Security and Privacy Workshops (SPW)*, 2022, pp. 208–214.

[11] P. Locatelli, M. Perri, D. M. Jimenez Gutierrez, A. Lacava, and F. Cuomo, "Device discovery and tracing in the bluetooth low energy domain," *Computer Communications*, Vol. 202, pp. 42–56, 2023. [Online]: https://www.sciencedirect.com/science/article/pii/S0140366423000452

[12] L. Alzubaidi, J. Bai, A. Al-Sabaawi, J. Santamaría, A. S. Albahri, B. S. N. Al-dabbagh, M. A. Fadhel, M. Manoufali, J. Zhang, A. H. Al-Timemy, Y. Duan, A. Abdullah, L. Farhan, Y. Lu, A. Gupta, F. Albu, A. Abbosh, and Y. Gu, "A survey on deep learning tools dealing with data scarcity: definitions, challenges, solutions, tips, and applications," *Journal of Big Data*, Vol. 10, No. 1, p. 46, Apr 2023. [Online]: https://doi.org/10.1186/s40537-023-00727-2

[13] L. Held and D. Sabanés Bové, *Markov Models for Time Series Analysis*. Berlin, Heidelberg, Springer Berlin Heidelberg, 2020, pp. 315–342. [Online]: https://doi.org/10.1007/978-3-662-60792-3_10

[14] C. S. W. Group, "Bluetooth core specification v5.4," Bluetooth Special Interest Group, Tech. Rep., January 2023, accessed: 11-09-2024. [Online]: https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=556599

[15] N. Semiconductor, "Bluetooth le advertising - advertisement packet," https://academy.nordicsemi.com/courses/bluetooth-low-energy-fundamentals/lessons/lesson-2-bluetooth-le-advertising/topic/advertisement-packet/, 2024, accessed: 1-09-2024.

[16] S. R. Saxer, "Dataset generation and feature extraction for high-traffic environment personal tracker identification," Bachelor Thesis Presentation, Department of Informatics - CSG, University of Zurich, July 2024, final Presentation.

[17] E. D. P. Supervisor, "Synthetic data," accessed: 08-10-2024. [Online]: https://www.edps.europa.eu/press-publications/publications/techsonar/synthetic-data_en#:~:text=Synthetic%20data%20is%20artificial%20data,undergoing%20the%20same%20statistical%20analysis.

[18] C. M. Bowen, V. L. Bryant, L. Burman, S. Khitatrakun, R. McClelland, L. Mucciolo, M. Pickens, and A. R. Williams, "Synthetic individual income tax data: Promises and challenges," *National Tax Journal*, Vol. 75, No. 4, pp. 767–790, 2022. [Online]: https://doi.org/10.1086/722094

[19] Z. Qian, T. Callender, B. Cebere, S. Janes, N. Navani, and M. V. D. Schaar, "Synthetic data for privacy-preserving clinical risk prediction," 2023.

[20] M. Pereira, S. Pentyala, A. Nascimento, R. T. D. Sousa, and M. D. Cock, "Secure multiparty computation for synthetic data generation from distributed data," *ArXiv*, Vol. abs/2210.07332, 2022.

[21] E. Barbierato, A. Pozzi, and D. Tessera, "Controlling bias between categorical attributes in datasets: A two-step optimization algorithm leveraging structural equation modeling," *IEEE Access*, Vol. 11, pp. 115 493–115 510, 2023.

[22] B. Draghi, Z. Wang, P. Myles, and A. Tucker, "Bayesboost: Identifying and handling bias using synthetic data generators," pp. 49–62, 2021.

[23] A. W. Services, "What is synthetic data?" accessed: 08-10-2024. [Online]: https://aws.amazon.com/what-is/synthetic-data/?nc1=h_ls

[24] K. M. Marcoulides, "A bayesian synthesis approach to data fusion using data-dependent priors," *Multivariate Behavioral Research*, Vol. 52, pp. 111 – 112, 2017.

[25] J. Drechsler and J. P. Reiter, "An empirical evaluation of easily implemented, nonparametric methods for generating synthetic datasets," *Computational Statistics Data Analysis*, Vol. 55, No. 12, pp. 3232–3243, 2011. [Online]: https://www.sciencedirect.com/science/article/pii/S0167947311002076

[26] J.-G. Choi, Y. Nah, I. Ko, and S. Han, "Deep learning approach to generate a synthetic cognitive psychology behavioral dataset," *IEEE Access*, Vol. 9, pp. 142 489–142 505, 2021.

[27] M. Goyal and Q. H. Mahmoud, "A systematic review of synthetic data generation techniques using generative ai," *Electronics*, Vol. 13, No. 17, 2024. [Online]: https://www.mdpi.com/2079-9292/13/17/3509

[28] Q. Mao, H.-Y. Lee, H.-Y. Tseng, S. Ma, and M.-H. Yang, "Mode seeking generative adversarial networks for diverse image synthesis," 2019, presented at CVPR 2019. [Online]: https://openaccess.thecvf.com/content_CVPR_2019/papers/Mao_Mode_Seeking_Generative_Adversarial_Networks_for_Diverse_Image_Synthesis_CVPR_2019_paper.pdf

[29] S. Liu, T. Wang, D. Bau, J.-Y. Zhu, and A. Torralba, "Diverse image generation via self-conditioned gans," 2022, presented at CVPR 2022. [Online]: https://openaccess.thecvf.com/content_CVPR_2020/papers/Liu_Diverse_Image_Generation_via_Self-Conditioned_GANs_CVPR_2020_paper.pdf

[30] Figueira and B. Vaz, "Survey on synthetic data generation, evaluation methods and gans," *Mathematics*, 2022.

[31] N. Torres-Reyes and S. Latifi, "Audio enhancement and synthesis using generative adversarial networks: A survey," *International Journal of Computer Applications*, 2019.

[32] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.

[33] F. A. Sonnenberg and J. R. Beck, "Markov models in medical decision making: A practical guide," *Medical Decision Making*, Vol. 13, No. 4, pp. 322–338, 1993, pMID: 8246705. [Online]: https://doi.org/10.1177/0272989X9301300409

[34] H. Wu and F. No'e, "Variational approach for learning markov processes from time series data," *Journal of Nonlinear Science*, Vol. 30, pp. 23 – 66, 2017.

[35] Y. Li, "Hidden markov models with states depending on observations," *Pattern Recognition Letters*, Vol. 26, No. 7, pp. 977–984, 2005. [Online]: https://www.sciencedirect.com/science/article/pii/S0167865504002922

[36] T. Lin, M. Wang, M. Yang, and X. Yang, "A hidden markov ensemble algorithm design for time series analysis," *Sensors (Basel, Switzerland)*, Vol. 22, 2022.

[37] R. M. Altman, "Mixed hidden markov models," *Journal of the American Statistical Association*, Vol. 102, No. 477, pp. 201–210, 2007. [Online]: https://doi.org/10.1198/016214506000001086

[38] S. Saxer, "Ble packets from tracking devices," https://www.kaggle.com/datasets/stefansaxer/ble-packets-from-tracking-devices, 2024, accessed: February 28, 2024.

[39] F. S. Daníş, A. T. Cemgíl, and C. Ersoy, "Adaptive sequential monte carlo filter for indoor positioning and tracking with bluetooth low energy beacons," *IEEE Access*, Vol. 9, pp. 37 022–37 038, 2021.

[40] F. S. Daniş and A. T. Cemgil, "Model-based localization and tracking using bluetooth low-energy beacons," *Sensors*, Vol. 17, No. 11, 2017. [Online]: https://www.mdpi.com/1424-8220/17/11/2484

[41] M. Ghamari, E. Villeneuve, C. Soltanpur, J. Khangosstar, B. Janko, R. S. Sherratt, and W. Harwin, "Detailed examination of a packet collision model for bluetooth low energy advertising mode," *IEEE Access*, Vol. 6, pp. 46 066–46 073, 2018.

[42] R. Shavelis and K. Ozols, "Bluetooth low energy wireless sensor network library in matlab simulink," *J. Sens. Actuator Networks*, Vol. 9, p. 38, 2020.

[43] G. Noblet, C. Lefebvre, P. Owezarski, and W. Ritchie, "Netglyph: Representation learning to generate network traffic with transformers," *2024 20th International Conference on Network and Service Management (CNSM)*, 2024.

[44] B. Ngoko, H. Sugihara, and T. Funaki, "Synthetic generation of high temporal resolution solar radiation data using markov models," *Solar Energy*, Vol. 103, pp. 160–170, 2014. [Online]: https://www.sciencedirect.com/science/article/pii/S0038092X14001042

[45] Z. Wang and J. Olivier, "Synthetic high-resolution wind data generation based on markov model," *2021 13th IEEE PES Asia Pacific Power Energy Engineering Conference (APPEEC)*, 2021, pp. 1–6.

[46] K. Brokish and J. Kirtley, "Pitfalls of modeling wind power using markov chains," *2009 IEEE/PES Power Systems Conference and Exposition*, 2009, pp. 1–6.

[47] P. Nystrup, H. Madsen, and E. Lindström, "Stylised facts of financial time series and hidden markov models in continuous time," *Quantitative Finance*, Vol. 15, pp. 1531 – 1541, 2015.

[48] P. Green and S. Richardson, "Hidden markov models and disease mapping," *Journal of the American Statistical Association*, Vol. 97, pp. 1055 – 1070, 2002.

[49] I. Saadi, A. Mustafa, J. Teller, B. Farooq, and M. Cools, "Hidden markov model-based population synthesis," *Transportation Research Part B: Methodological*, Vol. 90, pp. 1–21, 2016. [Online]: https://www.sciencedirect.com/science/article/pii/S0191261515300904

[50] S. R. Saxer, "Bachelor thesis: Stefan richard saxer," https://github.com/stsaxe/Bachelor-Thesis-Stefan-Richard-Saxer, 2023, [Accessed: Nov. 23, 2024].

[51] T. Yu, J. Henderson, A. Tiu, and T. Haines, "Privacy analysis of samsung's crowd-sourced bluetooth location tracking system," 2022. [Online]: https://arxiv.org/abs/2210.14702

Abbreviations

AdvA	Advertising Address
AdvData	Advertising Data
AD Length	Advertising Data Length
AD Type	Advertising Data Type
BLE	Bluetooth Low Energy
CRC	Cyclic Redundancy Check
GAN	Generative Adversarial Network
IDS	Intrusion Detection System
IP	Internet Protocol
MLP	Multi Layer Perceptron
MS Data	Manufacturer Specific Data
PDU	Protocol Data Unit
RSSI	Received Signal Strength Indicator
ST	SmartTag Type
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
USB	Universal Serial Bus
UUID	Universally Unique Identifier

List of Figures

List of Tables

Listings

Appendix A

Contents of the Repository

The GitHub repository can be found under the following URL: `https://github.com/keyyke/master_thesis`. This research expanded the Stefan's work [9]. Therefore, the program he created was partially used. The GitHub repository for Stefan's program is available at: `https://github.com/stsaxe/Bachelor-Thesis-Stefan-Richard-Saxer`.

NOTE: The program was run in Python version 3.12.

The code repository contains the following content:

Data for Markov Model

The data used for synthetic data generation with the Markov model and the synthetic data created:

- **processed_SmartTag_(nearby).csv**: The data used as the basis for the synthetic data. This data has been preprocessed. The raw data is located at `./stefan/-data/csv/SmartTag/SmartTag_(nearby).csv`.
- **synthetic_data_mm_SmartTag_(nearby).csv**: The synthetic data generated by the Markov model.

Evaluation

Figures and tables of the evaluation results of synthetic data are stored:

- **Confidence Level**: Diagrams showing the confidence level of the model.
- **Confusion Matrix**: Diagrams showing the confusion matrix.
- **Confusion Report**: CSV files showing the scores.

- **Structure**: Comparison of original and synthetic data structures.
- **Training Data Size**: Figure showing the relationship between training data size and accuracy.

Notebooks

Programs used to create and evaluate synthetic data:

- **analysing_data_structure.ipynb**: Compare the structure of synthetic data with that of original data.
- **generating_synthetic_data_markov.ipynb**: Generate the synthetic data using the Markov model.
- **inferring_with_synthetic_data_markov_percentage.ipynb**: Apply models with varying proportions of synthetic data in the training data to the real data. Parts of this program use code created by Stefan, which is noted in the corresponding cells through comments.
- **inferring_with_synthetic_data_markov.ipynb**: Apply the model trained using synthetic data generated by the Markov model to real data. Parts of this program use code created by Stefan, noted in the corresponding cells through comments.
- **modeling_with_synthetic_data_markov_percentage.ipynb**: Train the model, varying the percentage of synthetic data in the training data. Parts of this program use code created by Stefan, which is noted in the corresponding cells through comments.
- **modeling_with_synthetic_data_markov.ipynb**: Train the model using the synthetic data generated by the Markov model. Parts of this program use code created by Stefan, noted in the corresponding cells through comments.
- **preparing_data_markov.ipynb**: Prepare the data by applying preprocessing to the raw data. This data was used as the basis for synthetic data generation. Parts of this program use code created by Stefan, noted in the corresponding cells through comments.

Order of Execution:

1. preparing_data_markov
2. generating_synthetic_data_markov
3. modeling_with_synthetic_data_markov
4. analysing_data_structure
5. inferring_with_synthetic_data_markov

Pickle for Markov Model

Pickle objects of models and scalers were saved using synthetic data created by the Markov model as training data.

Presentation

Presentation slides used in the mid-term and final presentations.

State Transition Diagram

State transition diagram created during the synthetic data generation process using the Markov model.

Stefan's Program

A program created by Stefan. The repository link is: `https://github.com/stsaxe/Bachelor-Thesis-Stefan-Richard-Saxer`.

In this research, the model created without synthetic data and the synthetic data generated by Stefan were used. In some parts of the program, file names and paths were modified. These modifications are noted in the corresponding cells through comments.

Refer to the README in Stefan's repository for explanations of each program. This document focuses exclusively on the programs included in the `notebooks` directory:

- **Inference.ipynb**: Applying a model without synthetic data to real data.
- **InferenceWithSyntheticData.ipynb**: Applying a model with synthetic data to real data.
- **Modeling.ipynb**: Building a model without synthetic data.
- **ModelingWithSyntheticData.ipynb**: Building a model with synthetic data.
- **SyntheticDataGeneration.ipynb**: Synthetic data generation. Stefan created synthetic data by duplicating a subset of real data that was misclassified as SmartTag (nearby).

Order of Execution:

1. Modeling
2. Inference

3. SyntheticDataGeneration
4. ModelingWithSyntheticData
5. InferenceWithSyntheticData

Note: The data is too large to upload to GitHub. It can be accessed on Kaggle: `https://www.kaggle.com/datasets/stefansaxer/ble-packets-from-tracking-devices`

Appendix B

Confusion Matrices

Confusion matrices for SmartTag (nearby) with synthetic data proportions of 1%, 5%, 10%, 30%, 50%, 75%, and 100% in the training dataset.

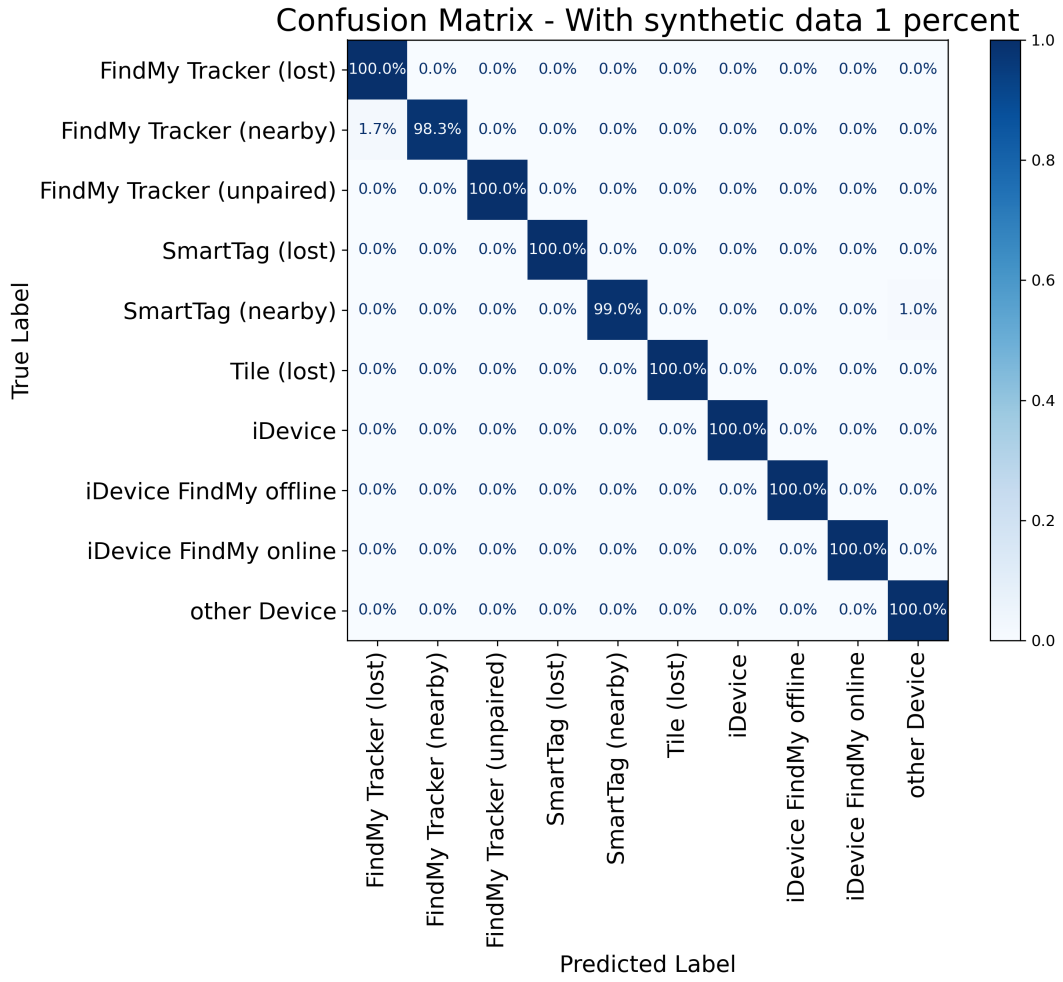


Figure B.1: Confusion Matrix - With synthetic data 1 percent

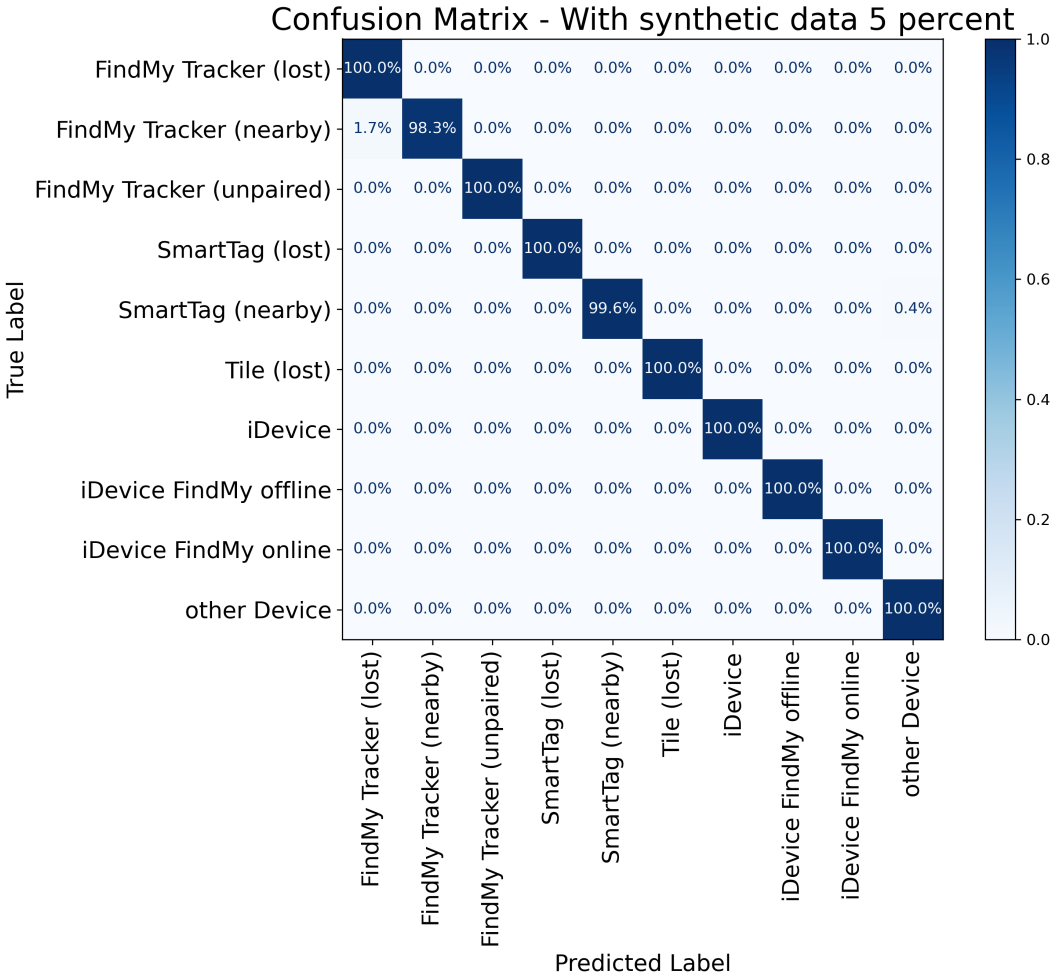


Figure B.2: Confusion Matrix - With synthetic data 5 percent

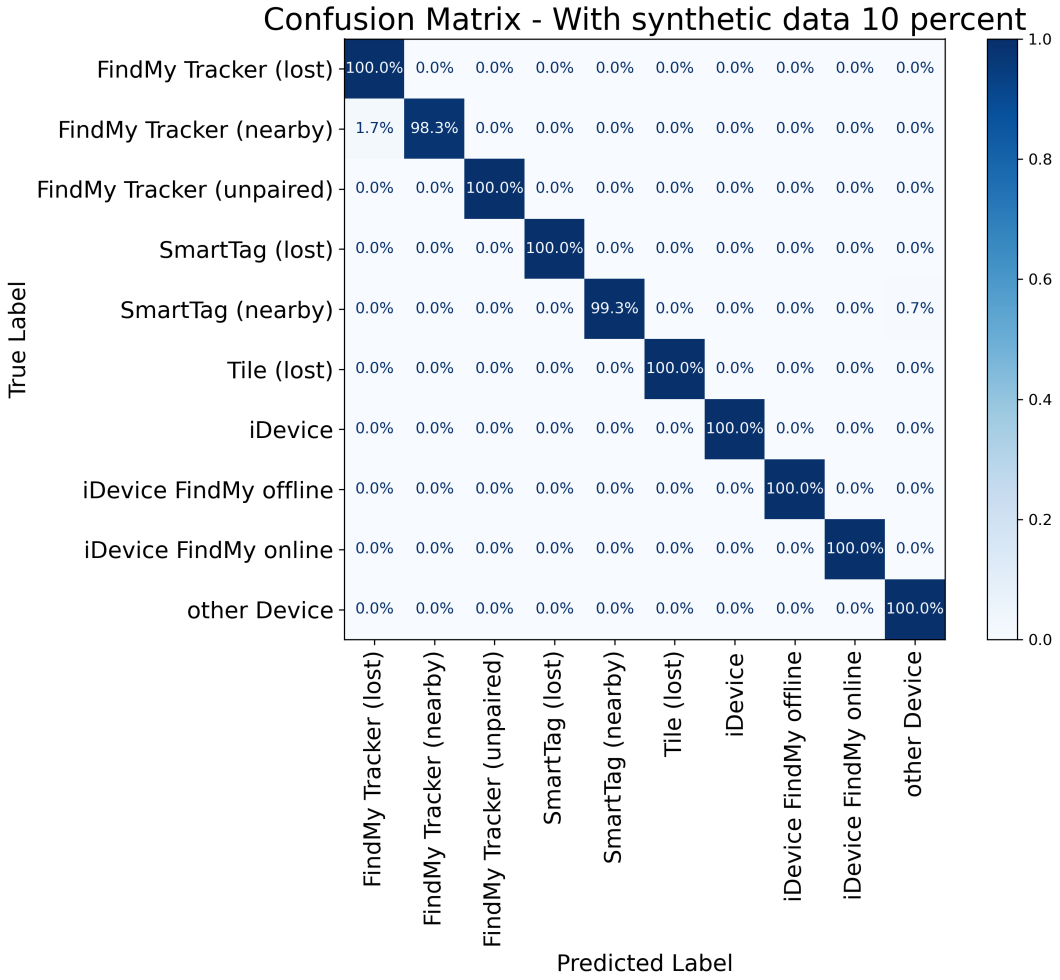


Figure B.3: Confusion Matrix - With synthetic data 10 percent

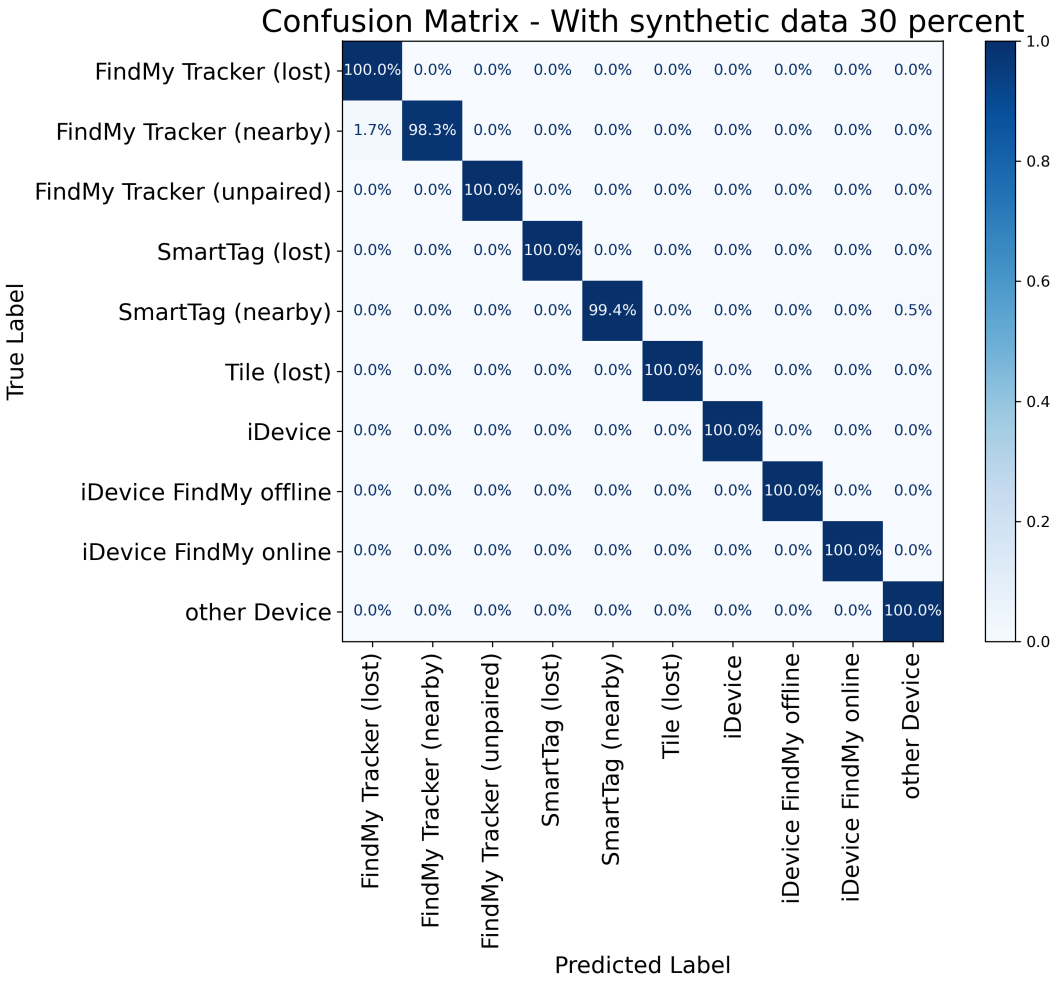


Figure B.4: Confusion Matrix - With synthetic data 30 percent

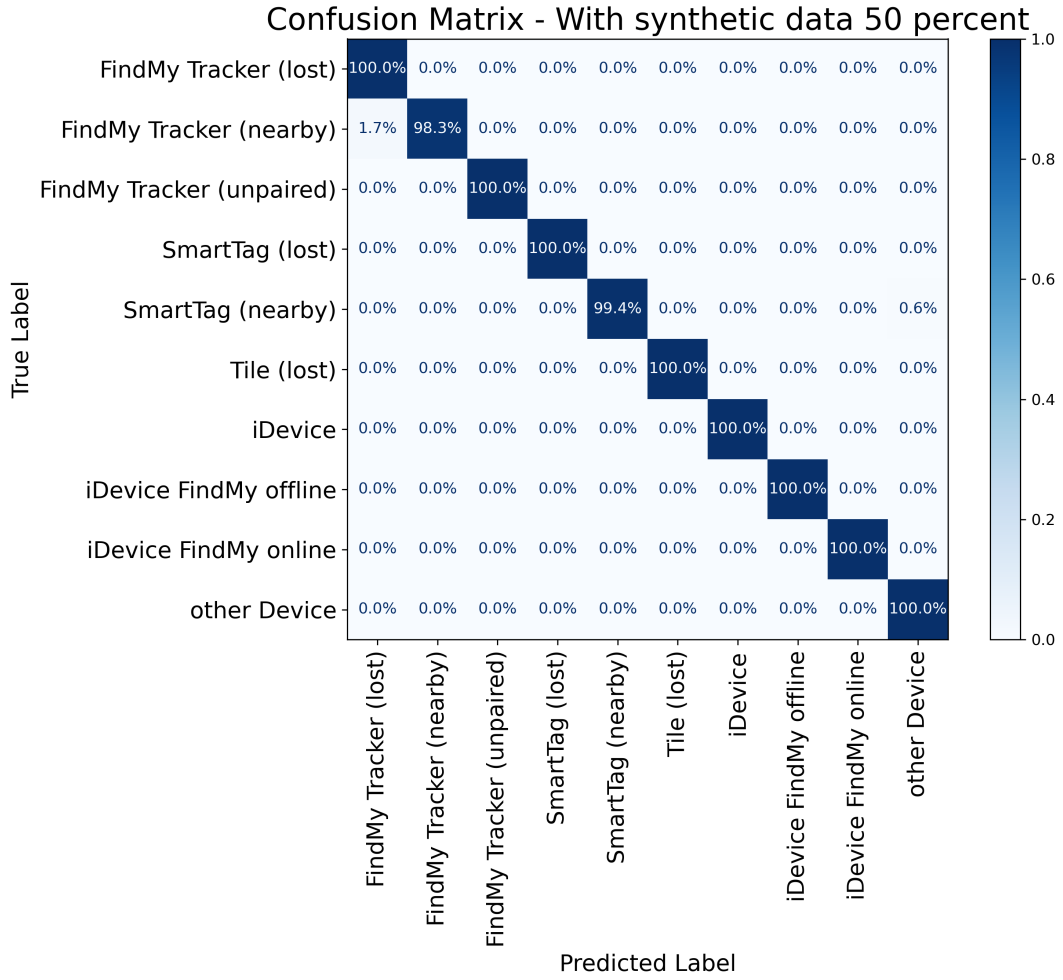


Figure B.5: Confusion Matrix - With synthetic data 50 percent

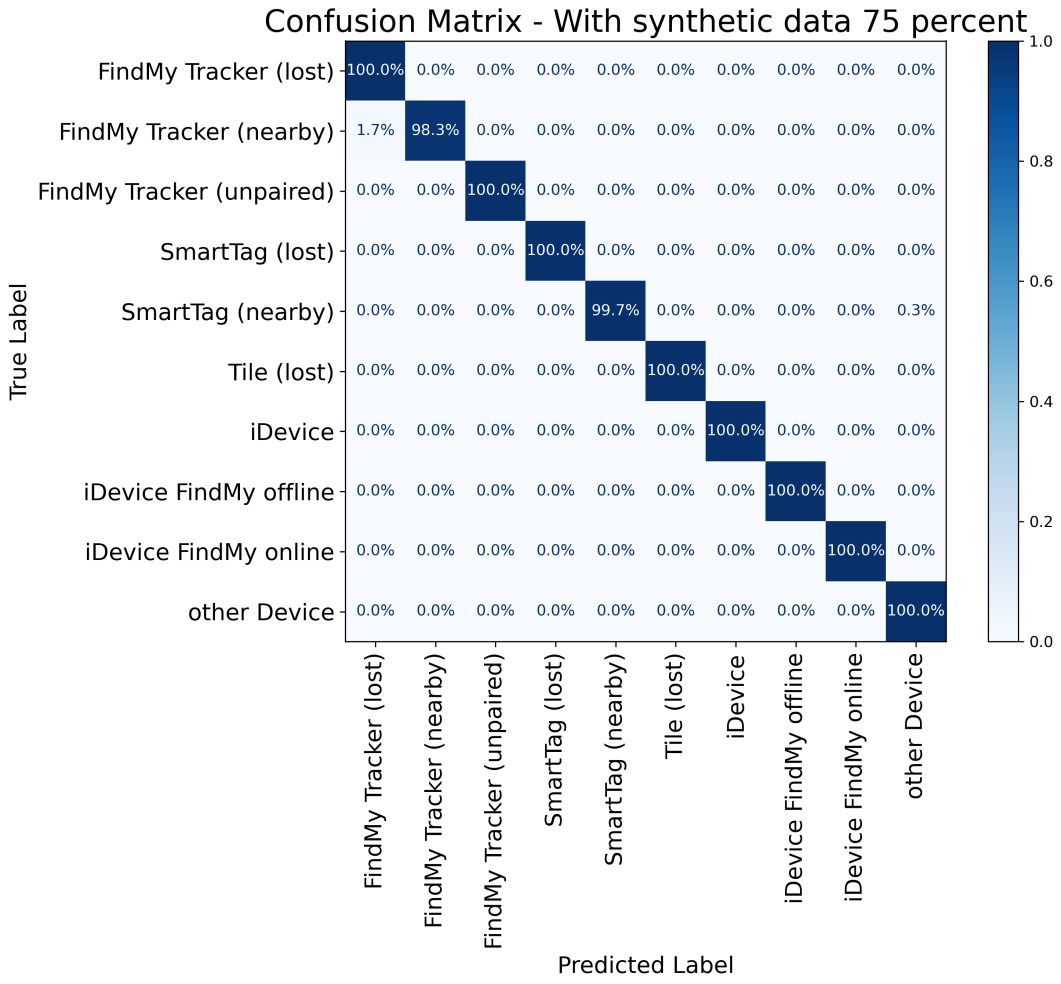


Figure B.6: Confusion Matrix - With synthetic data 75 percent

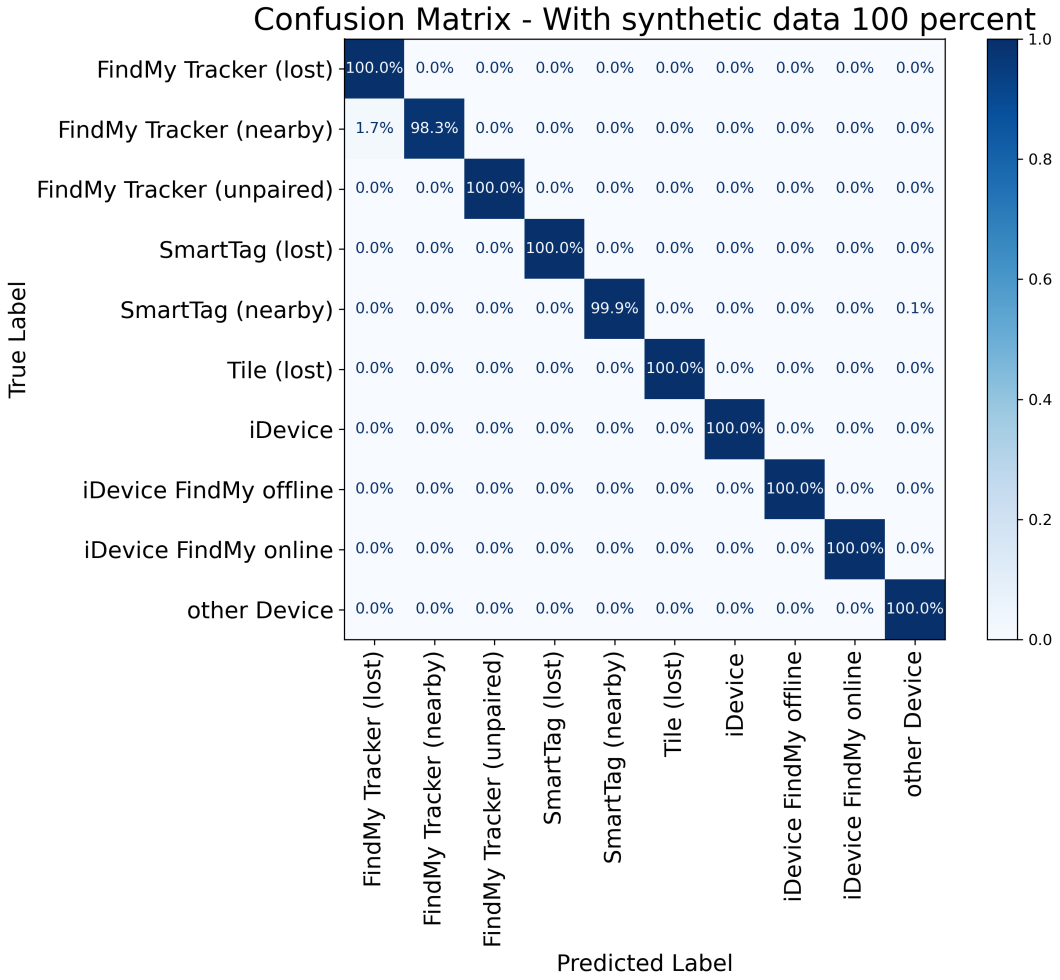
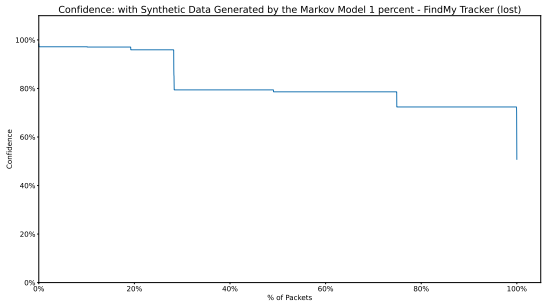


Figure B.7: Confusion Matrix - With synthetic data 100 percent

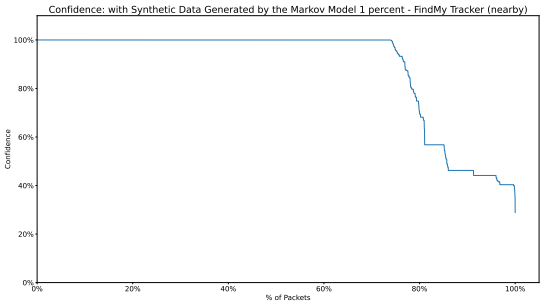
Appendix C

Confidence Levels

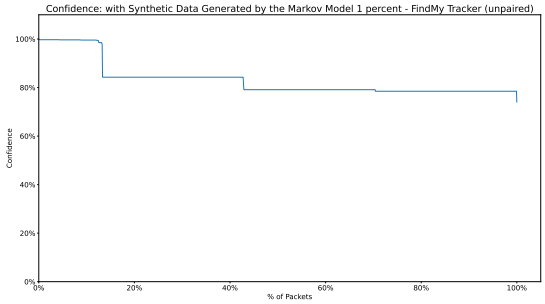
Confusion levels for SmartTag (nearby) with synthetic data proportions of 1%, 5%, 10%, 30%, 50%, 75%, and 100% in the training dataset.



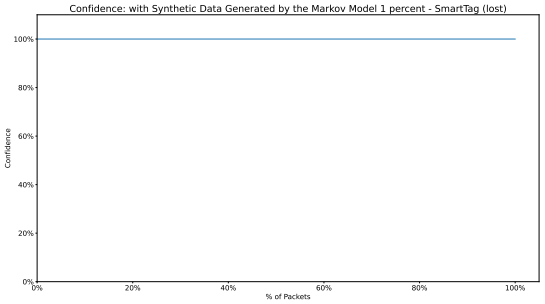
(a) Confidence - FindMy Tracker (lost)



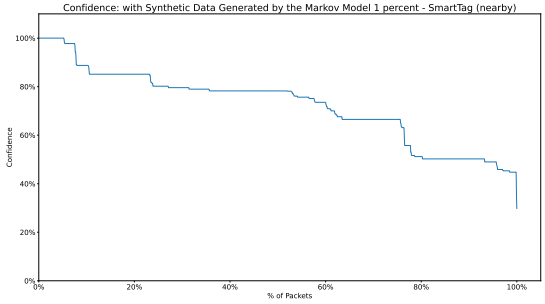
(b) Confidence - FindMy Tracker (nearby)



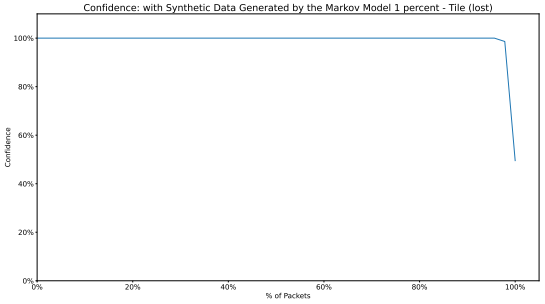
(c) Confidence - FindMy Tracker (unpaired)



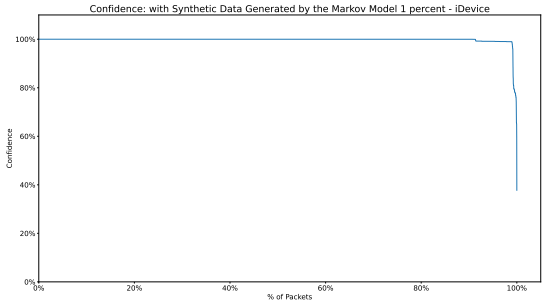
(d) Confidence - SmartTag (lost)



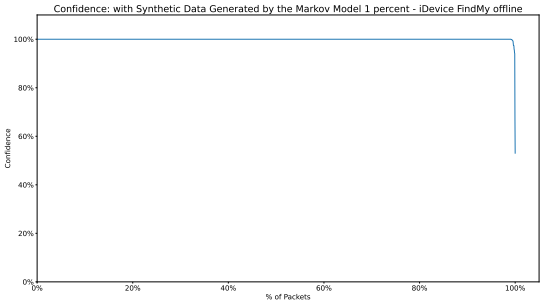
(e) Confidence - SmartTag (nearby)



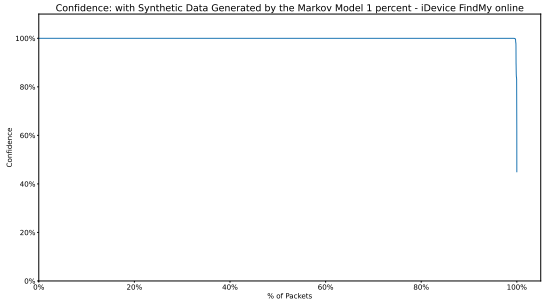
(f) Confidence - Tile (lost)



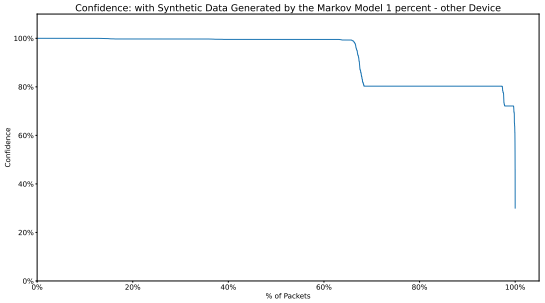
(g) Confidence - iDevice



(h) Confidence - iDevice FindMy offline

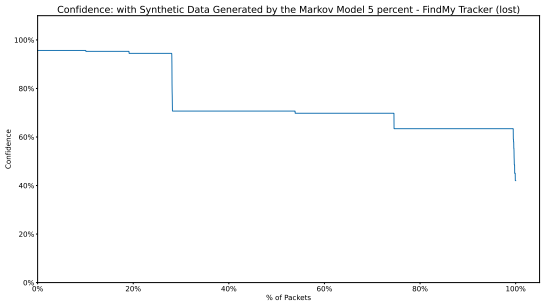


(i) Confidence - iDevice FindMy online

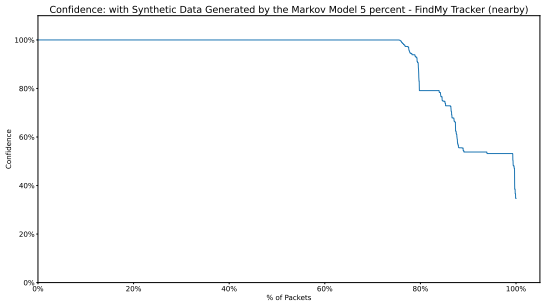


(j) Confidence - other Device

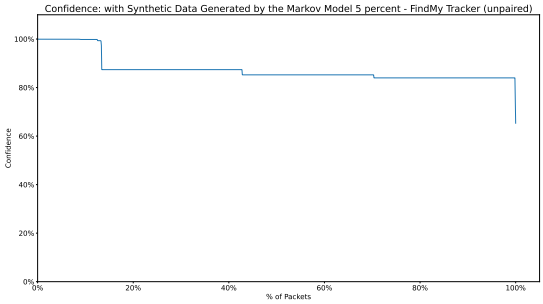
Figure C.1: Confidence - With Synthetic Data Generated by the Markov Model 1 percent



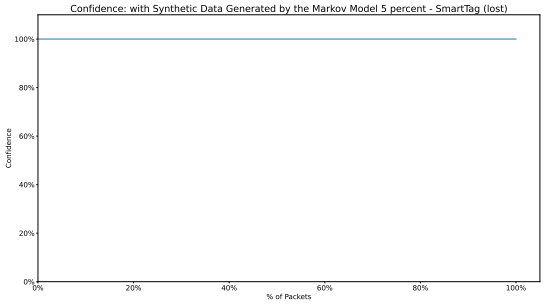
(a) Confidence - FindMy Tracker (lost)



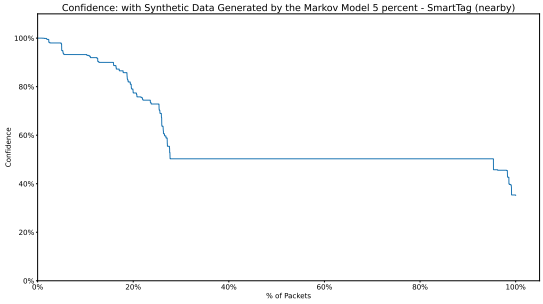
(b) Confidence - FindMy Tracker (nearby)



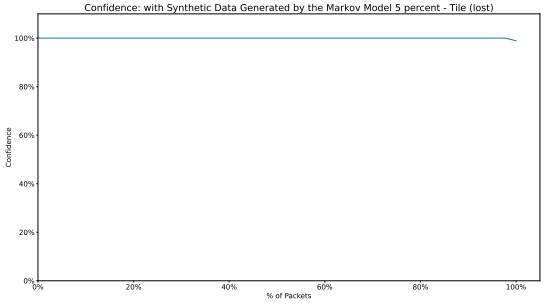
(c) Confidence - FindMy Tracker (unpaired)



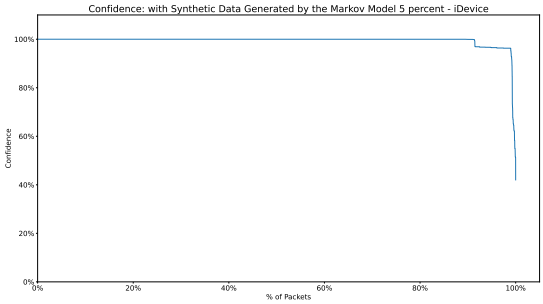
(d) Confidence - SmartTag (lost)



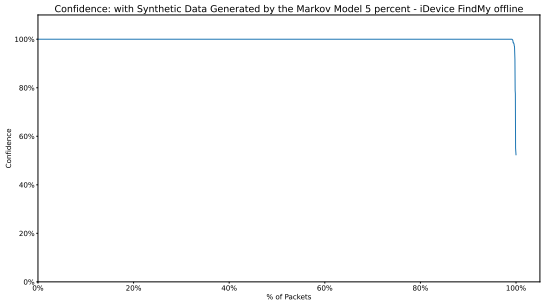
(e) Confidence - SmartTag (nearby)



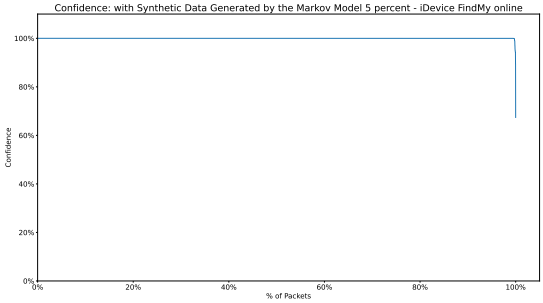
(f) Confidence - Tile (lost)



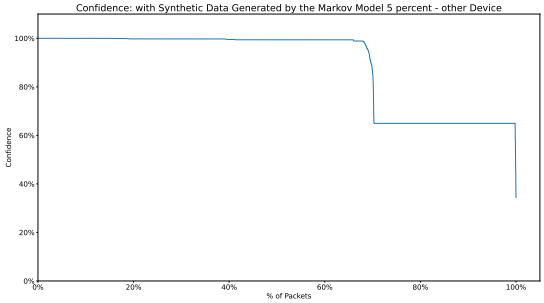
(g) Confidence - iDevice



(h) Confidence - iDevice FindMy offline

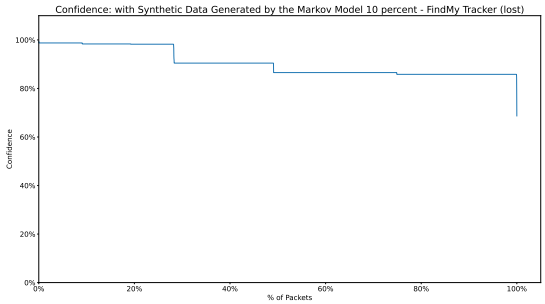


(i) Confidence - iDevice FindMy online

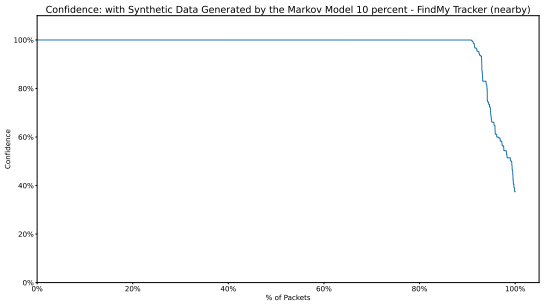


(j) Confidence - other Device

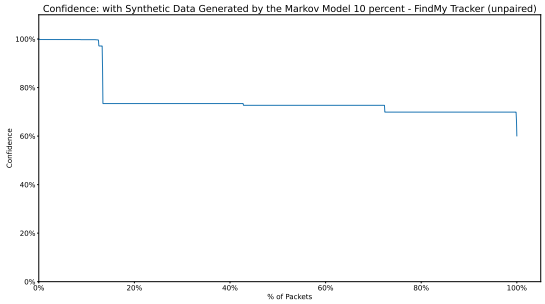
Figure C.2: Confidence - With Synthetic Data Generated by the Markov Model 5 percent



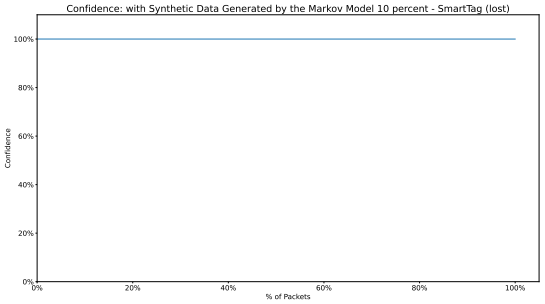
(a) Confidence - FindMy Tracker (lost)



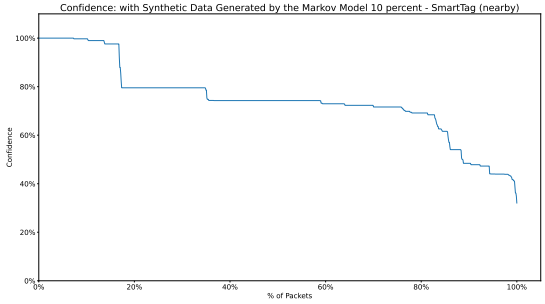
(b) Confidence - FindMy Tracker (nearby)



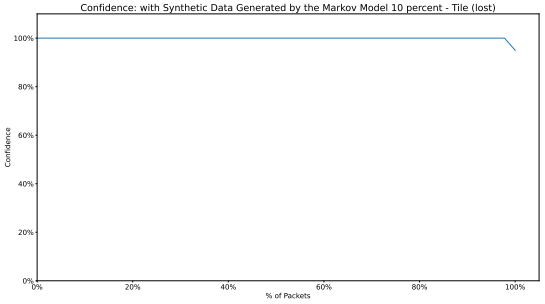
(c) Confidence - FindMy Tracker (unpaired)



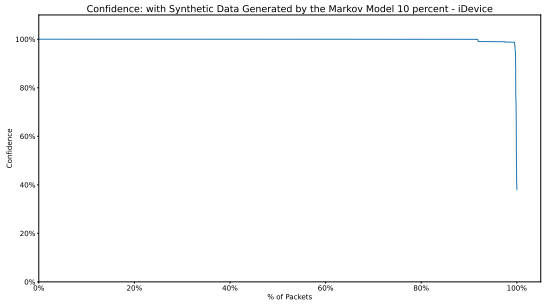
(d) Confidence - SmartTag (lost)



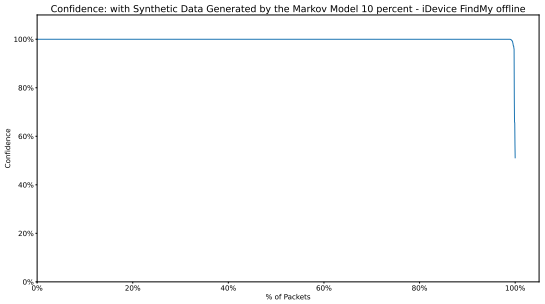
(e) Confidence - SmartTag (nearby)



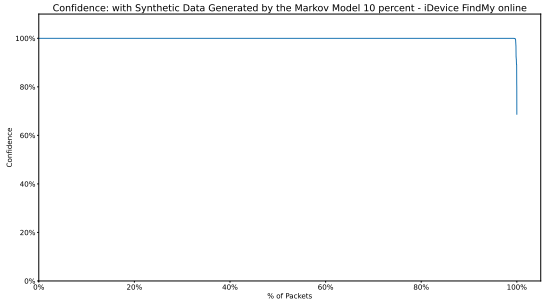
(f) Confidence - Tile (lost)



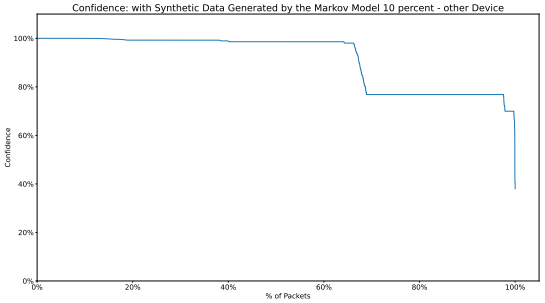
(g) Confidence - iDevice



(h) Confidence - iDevice FindMy offline

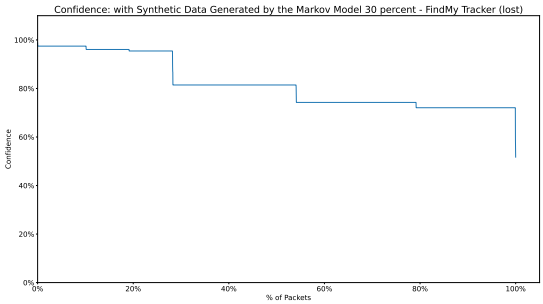


(i) Confidence - iDevice FindMy online

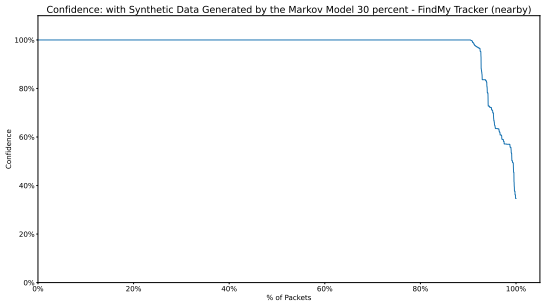


(j) Confidence - other Device

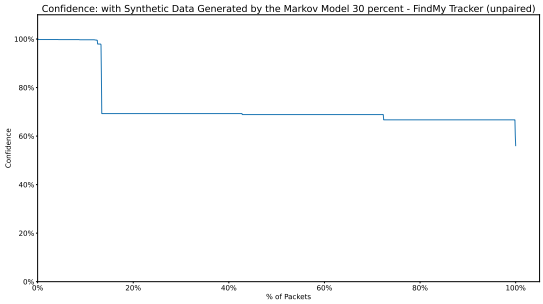
Figure C.3: Confidence - With Synthetic Data Generated by the Markov Model 10 percent



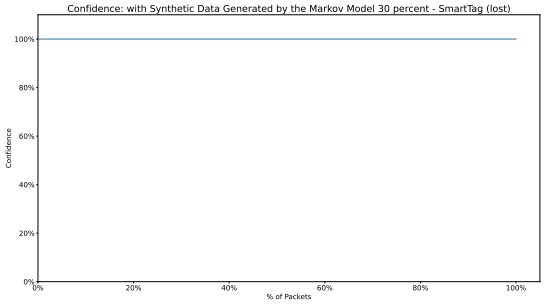
(a) Confidence - FindMy Tracker (lost)



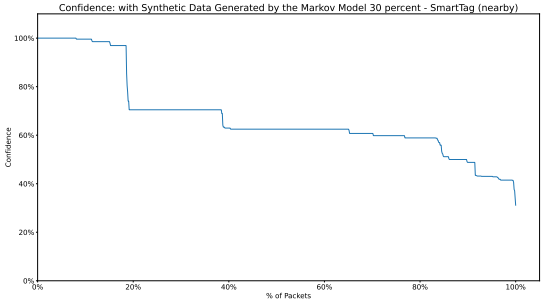
(b) Confidence - FindMy Tracker (nearby)



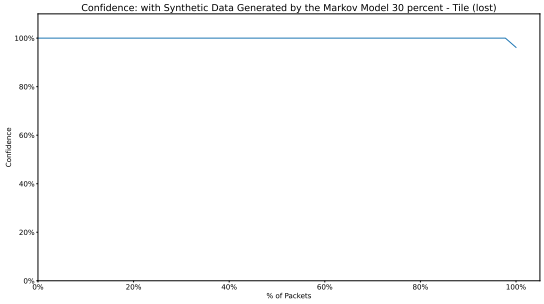
(c) Confidence - FindMy Tracker (unpaired)



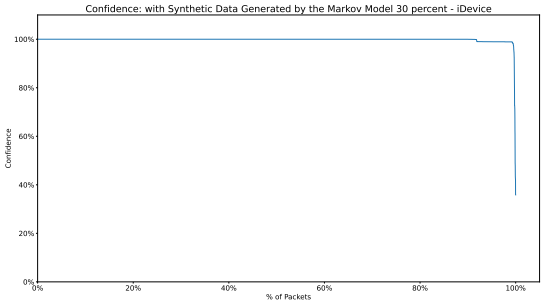
(d) Confidence - SmartTag (lost)



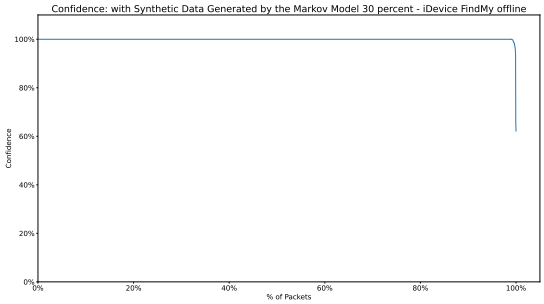
(e) Confidence - SmartTag (nearby)



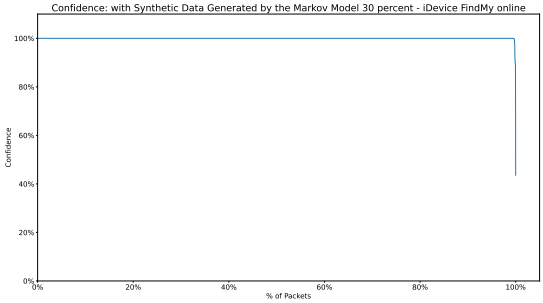
(f) Confidence - Tile (lost)



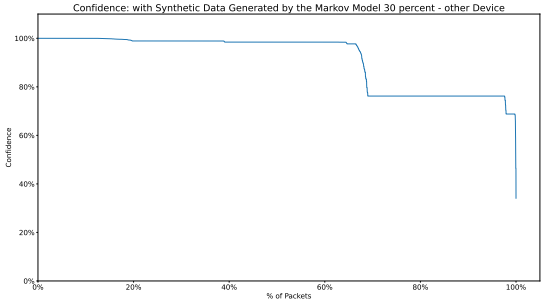
(g) Confidence - iDevice



(h) Confidence - iDevice FindMy offline

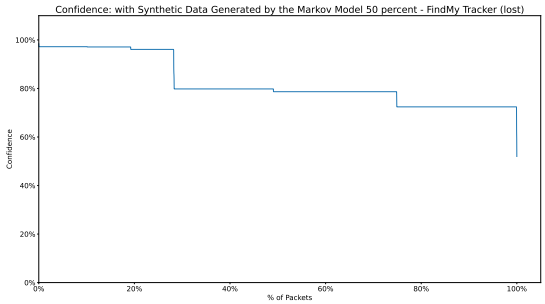


(i) Confidence - iDevice FindMy online

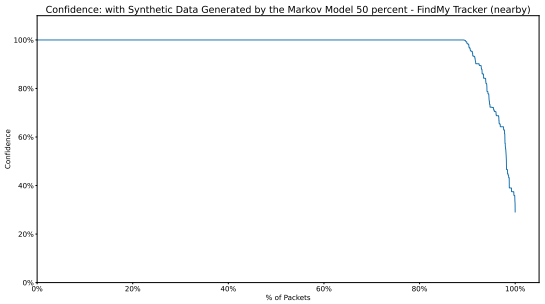


(j) Confidence - other Device

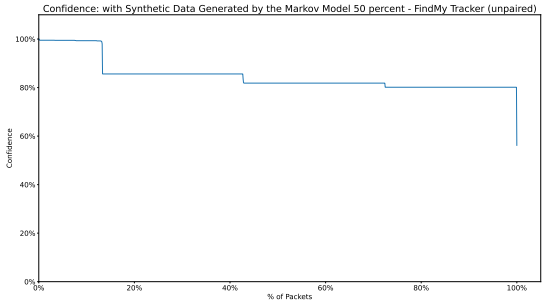
Figure C.4: Confidence - With Synthetic Data Generated by the Markov Model 30 percent



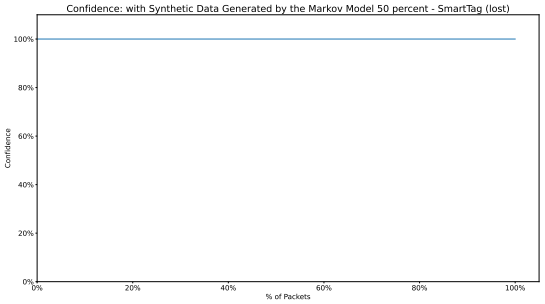
(a) Confidence - FindMy Tracker (lost)



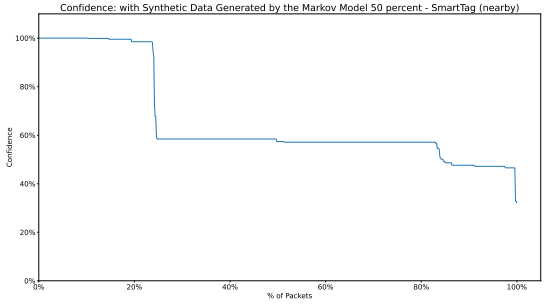
(b) Confidence - FindMy Tracker (nearby)



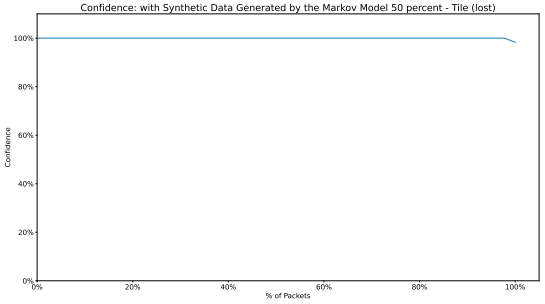
(c) Confidence - FindMy Tracker (unpaired)



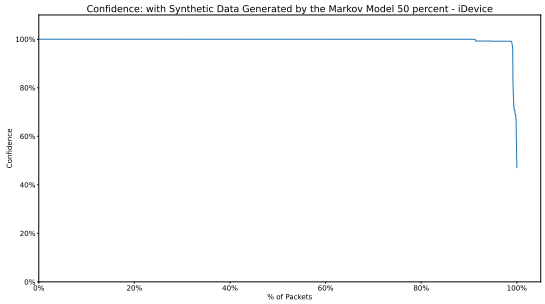
(d) Confidence - SmartTag (lost)



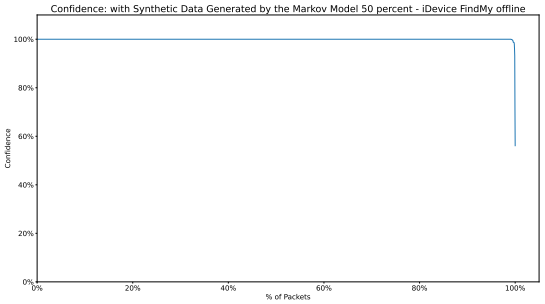
(e) Confidence - SmartTag (nearby)



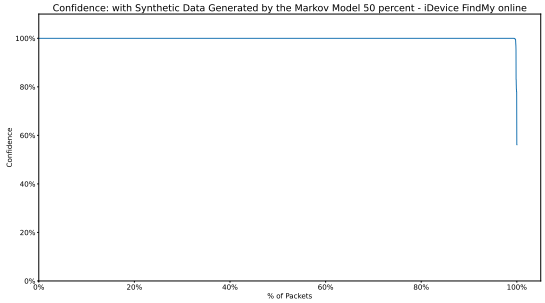
(f) Confidence - Tile (lost)



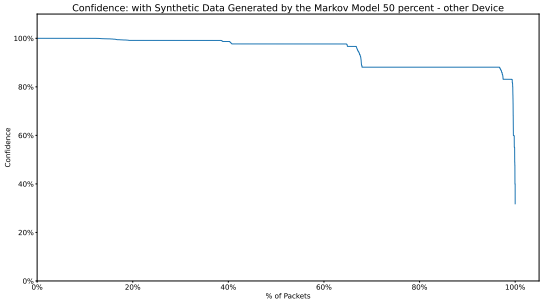
(g) Confidence - iDevice



(h) Confidence - iDevice FindMy offline

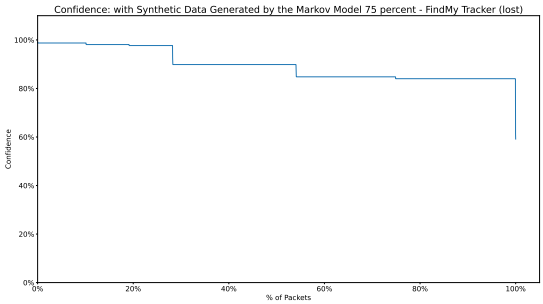


(i) Confidence - iDevice FindMy online

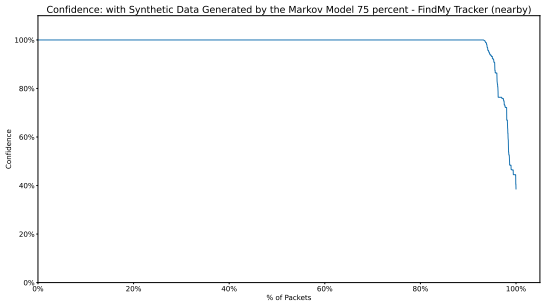


(j) Confidence - other Device

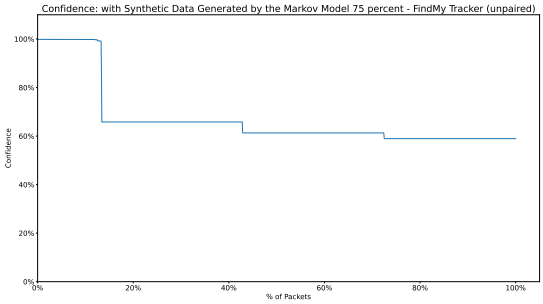
Figure C.5: Confidence - With Synthetic Data Generated by the Markov Model 50 percent



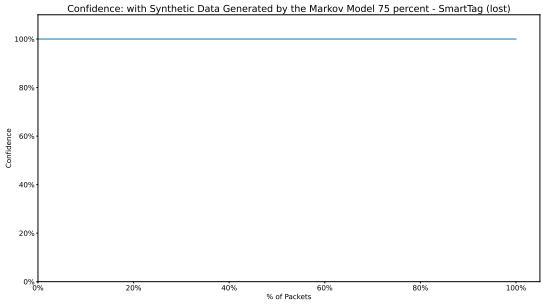
(a) Confidence - FindMy Tracker (lost)



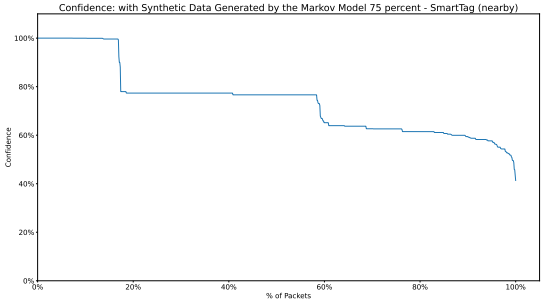
(b) Confidence - FindMy Tracker (nearby)



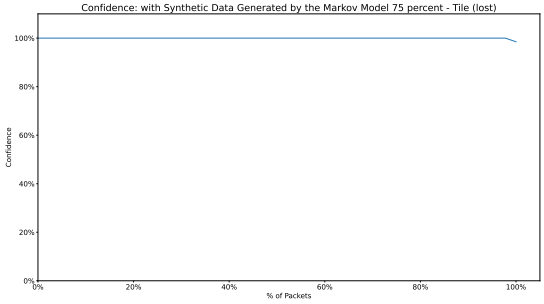
(c) Confidence - FindMy Tracker (unpaired)



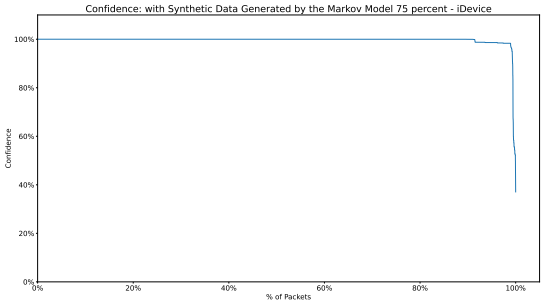
(d) Confidence - SmartTag (lost)



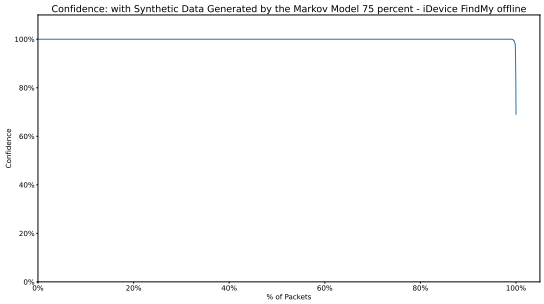
(e) Confidence - SmartTag (nearby)



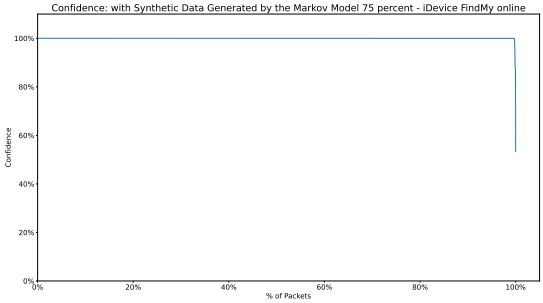
(f) Confidence - Tile (lost)



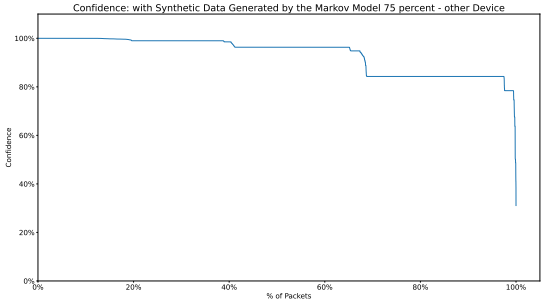
(g) Confidence - iDevice



(h) Confidence - iDevice FindMy offline

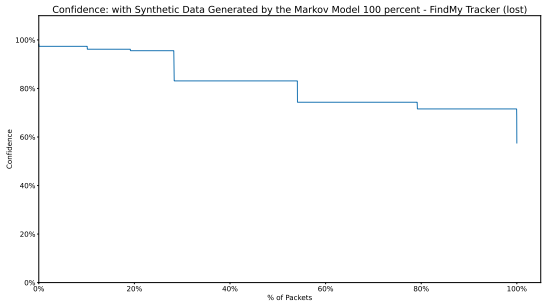


(i) Confidence - iDevice FindMy online

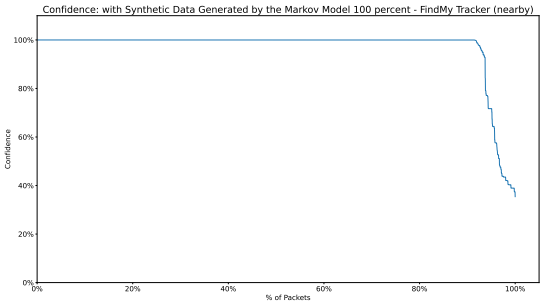


(j) Confidence - other Device

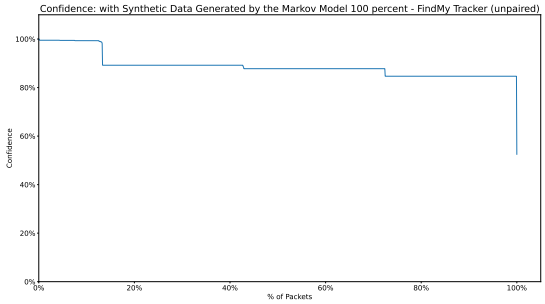
Figure C.6: Confidence - With Synthetic Data Generated by the Markov Model 75 percent



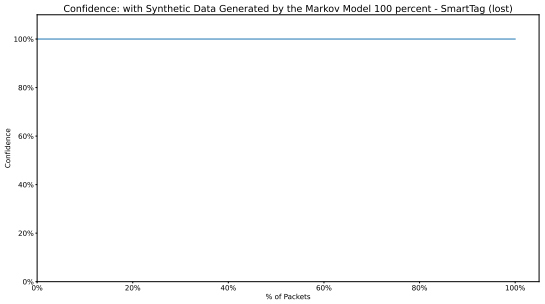
(a) Confidence - FindMy Tracker (lost)



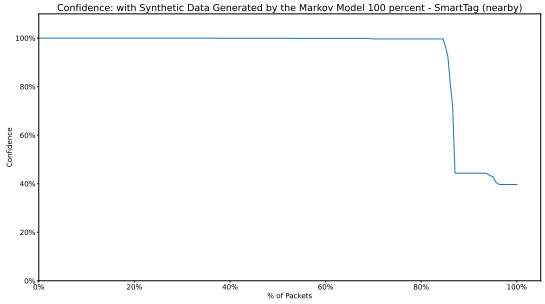
(b) Confidence - FindMy Tracker (nearby)



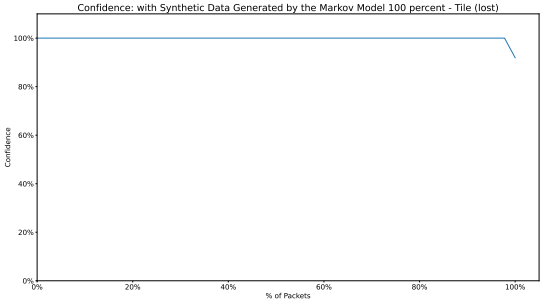
(c) Confidence - FindMy Tracker (unpaired)



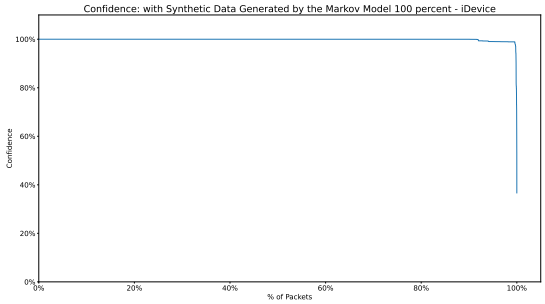
(d) Confidence - SmartTag (lost)



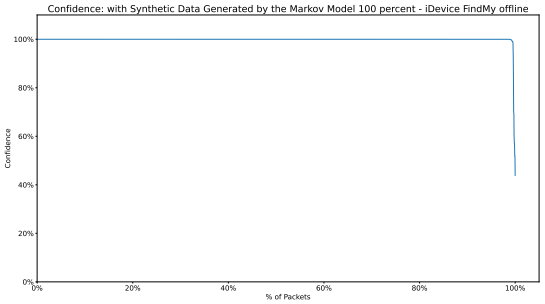
(e) Confidence - SmartTag (nearby)



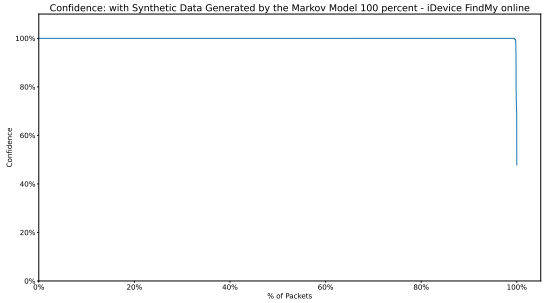
(f) Confidence - Tile (lost)



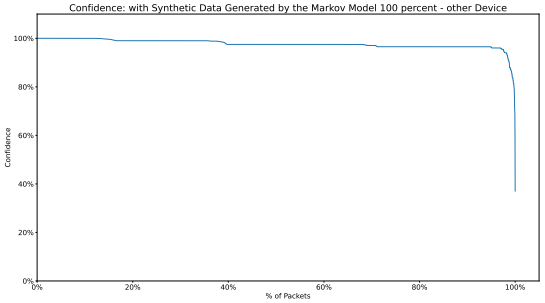
(g) Confidence - iDevice



(h) Confidence - iDevice FindMy offline



(i) Confidence - iDevice FindMy online



(j) Confidence - other Device

Figure C.7: Confidence - With Synthetic Data Generated by the Markov Model 100 percent