PRIVACY-PRSERVING PRESELECTION FOR PROTECTED BIOMETRIC IDENTIFICATION USING PUBLIC-KEY ENCRYPTION WITH KEYWORD SEARCH

P.Sri Chandana, Department of CSE, Narayana Engineering College, Gudur, India
Mrs. S.Sunanda, Assistant Professor, Department of CSE, Narayana Engineering College, Gudur, India

Abstract: The efficiency of biometric systems, in particular efficient and accurate biometric identification, is one of the most challenging open problems in biometrics today. Adding to that, biometric data are sensitive data deserving adequate protection. As a solution, this work proposes an efficient privacy-preserving reduction of the computational workload of biometric identification systems using public-key encryption with keyword search. For long-term protection of the biometric data, fully homomorphic encryption is applied for template protection. As all applied cryptographic schemes are lattice-based, they also offer post-quantum security. Throughout the system, the recognition accuracy of the unprotected system is preserved. The system is identified the person is authorized or un-authorized. If the person is authorized, the user can view the original information or else the user can't able to view the information. The system is developed the machine learning algorithm for classifying the person is authorized or un-authorized by using random forest classifier. Finally, the experimental results shows that some performance metrics such as accuracy and error rate.

I. INTRODUCTION

Automated biometric recognition has become an established part of everyday life, from personal device access to smart border control gates. While biometric authentication offers high usability and security, it comes with potential privacy risks, as biometric data are recognised as sensitive personal data by the General Data Protection Regulation. This is particularly true when biometric data are used for identification searches, where the biometric references need to be stored centrally to facilitate a search for an unknown probe. If an attacker gains access to the signal representation of the biometric characteristic of a subject, the reference attributed to this characteristic can no longer be used securely for authentication due to the risk of impersonation.

Deep learning-based methods represent the current state of-the-art for solving pattern recognition tasks including biometric recognition. Applied feature extraction methods are commonly trained using differentiable loss functions, e.g. Euclidean distance. Therefore, extracted feature vectors are usually represented as real-valued vectors of fixed dimension, which define biometric templates.

II. IMPLEMENTATION

1.INPUT IMAGE:

- The dataset, FERET dataset is implemented as input. The dataset is taken from dataset repository.
- The input dataset is in the format '.png, '.jpg.
- In this step, we have to read or load the input image by using the imread () function.
- In our process, we are used the tkinter file dialogue box for selecting the input image.

2: PREPROCESSING:

- In our process, we have to resize the image and convert the image into gray scale.
- To **resize an image**, you call the resize () method on it, passing in a two-integer tuple argument representing the width and height of the resized image.
- The function doesn't modify the used image; it instead returns another Image with the new dimensions.
- Convert an Image to **Grayscale** in Python Using the Conversion Formula and the matplotlib Library.
- We can also convert an image to grayscale using the standard RGB to grayscale conversion formula that is imgGray = 0.2989 * R + 0.5870 * G + 0.1140 * B.

3: FEATURE EXTRACTION:

- In this step, we can implement or can extract the features from pre-processed image by using LBP.
- **Local Binary Pattern (LBP)** is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighbourhood of each pixel and considers the result as a binary number.
- LBPH (Local Binary Pattern Histogram) is a Face-Recognition algorithm it is used to recognize the face of a person.
- It is known for its performance and how it is able to recognize the face of a person from both front face and side face.

4: BIOMETRIC INFORMATION:

- In this step, we can extract the biometric information for corresponding input person.
- The biometric information such as person name, age, sex and so on.

5: ENCRYPTION:

- In this step, we can encrypt the original biometric information by using RSA and the encrypted data will be stored in cloud.
- The **RSA** algorithm is an asymmetric cryptography algorithm; this means that it uses a public key and a private key (i.e two different, mathematically linked keys).
- As their names suggest, a public key is shared publicly, while a private key is secret and must not be shared with anyone.

6 .CLASSIFICATION:

- In our process, we have to implement the machine learning algorithm such as random forest.
- **Random forest** is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems.
- It builds decision trees on different samples and takes their majority vote for classification and average in case of regression.
- Below are some points that explain why we should use the Random Forest algorithm: It takes less training time as compared to other algorithms.
- It predicts output with high accuracy, even for the large dataset it runs efficiently.
- It can also maintain accuracy when a large proportion of data is missing.

7: PERFORMANCE:

The Final Result will get generated based on the overall classification and prediction. The performance of this proposed approach is evaluated using some measures like

- **Accuracy**

Accuracy of classifier refers to the ability of classifier. It predicts the class label correctly and the accuracy of the predictor refers to how well a given predictor can guess the value of predicted attribute for a new data.

$$AC= (TP+TN)/ (TP+TN+FP+FN)$$

- **Error rate** refers to a measure of the degree of prediction error of a model made with respect to the true model.

III. SYSTEM REQUIREMENTS

1 HARDWARE REQUIREMENTS:

- System : Pentium IV 2.4 GHz
- Hard Disk : 200 GB
- Mouse : Logitech.
- Keyboard : 110 keys enhanced
- Ram : 4GB

2 SOFTWARE REQUIREMENTS:

- O/S : Windows 7.
- Language : Python
- Front End : Anaconda Navigator – Spyder

3 SOFTWARE DESCRIPTION:

3.1 Python

Python is one of those rare languages which can claim to be both *simple* and powerful. You will find yourself pleasantly surprised to see how easy it is to concentrate on the solution to the problem rather than the syntax and structure of the language you are programming in. The official introduction to Python is Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms. I will discuss most of these features in more detail in the next section.

3.2 Features of Python

- **Simple**

Python is a simple and minimalistic language. Reading a good Python program feels almost like reading English, although very strict English! This pseudo-code nature of Python is one of its greatest strengths. It allows you to concentrate on the solution to the problem rather than the language itself.

- **Easy to Learn**

As you will see, Python is extremely easy to get started with. Python has an extraordinarily simple syntax, as already mentioned.

- **Free and Open Source**

Python is an example of a *FLOSS* (Free/Libré and Open Source Software). In simple terms, you can freely distribute copies of this software, read its source code, make changes to it, and use pieces of it in new free programs. FLOSS is based on the concept of a community which shares knowledge. This is one of the reasons why Python is so good - it has been created and is constantly improved by a community who just want to see a better Python.

- **High-level Language**

When you write programs in Python, you never need to bother about the low-level details such as managing the memory used by your program, etc.

- **Portable**

Due to its open-source nature, Python has been ported to (i.e. changed to make it work on) many platforms. All your Python programs can work on any of these platforms without requiring any changes at all if you are careful enough to avoid any system-dependent features.

You can use Python on GNU/Linux, Windows, FreeBSD, Macintosh, Solaris, OS/2, Amiga, AROS, AS/400, BeOS, OS/390, z/OS, Palm OS, QNX, VMS, Psion, Acorn RISC OS, VxWorks, PlayStation, Sharp Zaurus, Windows CE and PocketPC!

You can even use a platform like Kivy to create games for your computer *and* for iPhone, iPad, and Android.

- **Interpreted**

This requires a bit of explanation.

A program written in a compiled language like C or C++ is converted from the source language i.e. C or C++ into a language that is spoken by your computer (binary code i.e. 0s and 1s) using a compiler with various flags and options. When you run the program, the linker/loader software copies the program from hard disk to memory and starts running it.

Python, on the other hand, does not need compilation to binary. You just *run* the program directly from the source code. Internally, Python converts the source code into an intermediate form called byte codes and then translates this into the native language of your computer and then runs it. All this, actually, makes using Python much easier since you don't have to worry about compiling the program, making sure that the proper libraries are linked and loaded, etc. This also makes your Python programs much more portable, since you can just copy your Python program onto another computer and it just works!

- **Object Oriented**

Python supports procedure-oriented programming as well as object-oriented programming. In *procedure-oriented* languages, the program is built around procedures or functions which are nothing but reusable pieces of programs. In *object-oriented* languages, the program is built around objects which combine data and functionality. Python has a very powerful but simplistic way of doing OOP, especially when compared to big languages like C++ or Java.

- **Extensible**

If you need a critical piece of code to run very fast or want to have some piece of algorithm not to be open, you can code that part of your program in C or C++ and then use it from your Python program.

- **Embeddable**

You can embed Python within your C/C++ programs to give *scripting* capabilities for your program's users.

- **Extensive Libraries**

The Python Standard Library is huge indeed. It can help you do various things involving regular expressions, documentation generation, unit testing, threading, databases, web browsers, CGI, FTP, email, XML, XML-RPC, HTML,

WAV files, cryptography, GUI (graphical user interfaces), and other system-dependent stuff. Remember, all this is always available wherever Python is installed. This is called the *Batteries Included* philosophy of Python.

Besides the standard library, there are various other high-quality libraries which you can find at the Python Package Index.

4. TESTING PRODUCTS:

System testing is the stage of implementation, which aimed at ensuring that system works accurately and efficiently before the live operation commence. Testing is the process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an error. A successful test is one that answers a yet undiscovered error.

Testing is vital to the success of the system. System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved. . A series of tests are performed before the system is ready for the user acceptance testing. Any engineered product can be tested in one of the following ways. Knowing the specified function that a product has been designed to from, test can be conducted to demonstrate each function is fully operational. Knowing the internal working of a product, tests can be conducted to ensure that "al gears mesh", that is the internal operation of the product performs according to the specification and all internal components have been adequately exercised.

4.1 UNIT TESTING:

Unit testing is the testing of each module and the integration of the overall system is done. Unit testing becomes verification efforts on the smallest unit of software design in the module. This is also known as 'module testing'.

The modules of the system are tested separately. This testing is carried out during the programming itself. In this testing step, each model is found to be working satisfactorily as regard to the expected output from the module. There are some validation checks for the fields. For example, the validation check is done for verifying the data given by the user where both format and validity of the data entered is included. It is very easy to find error and debug the system.

4.2 INTEGRATION TESTING:

Data can be lost across an interface, one module can have an adverse effect on the other sub function, when combined, may not produce the desired major function. Integrated testing is systematic testing that can be done with sample data. The need for the integrated test is to find the overall system performance. There are two types of integration testing. They are:

i) Top-down integration testing. ii) Bottom-up integration testing.

4.3 TESTING TECHNIQUES/STRATEGIES:

- **WHITE BOX TESTING:**

White Box testing is a test case design method that uses the control structure of the procedural design to drive cases. Using the white box testing methods, we Derived test cases that guarantee that all independent paths within a module have been exercised at least once.

- **BLACK BOX TESTING:**

1. Black box testing is done to find incorrect or missing function
2. Interface error
3. Errors in external database access
4. Performance errors.
5. Initialization and termination errors

In 'functional testing', is performed to validate an application conforms to its specifications of correctly performs all its required functions. So this testing is also called 'black box testing'. It tests the external behaviour of the system. Here the engineered product can be tested knowing the specified function that a product has been designed to perform, tests can be conducted to demonstrate that each function is fully operational.

4.4 SOFTWARE TESTING STRATEGIES

VALIDATION TESTING:

After the culmination of black box testing, software is completed assembly as a package, interfacing errors have been uncovered and corrected and final series of software validation tests begin validation testing can be defined as many But a single definition is that validation succeeds when the software functions in a manner that can be reasonably expected by the customer.

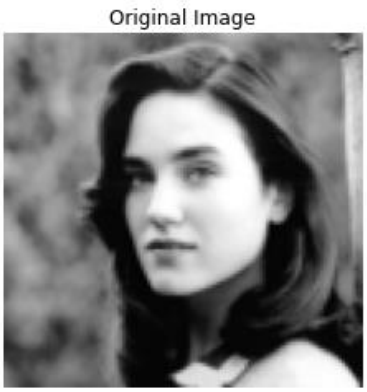
USER ACCEPTANCE TESTING:

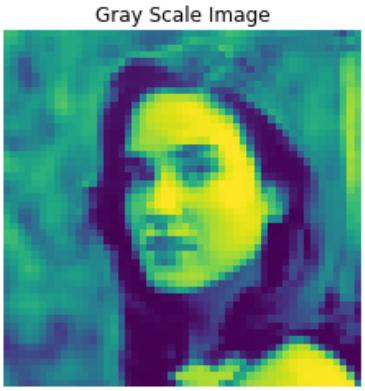
User acceptance of the system is the key factor for the success of the system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system at the time of developing changes whenever required.

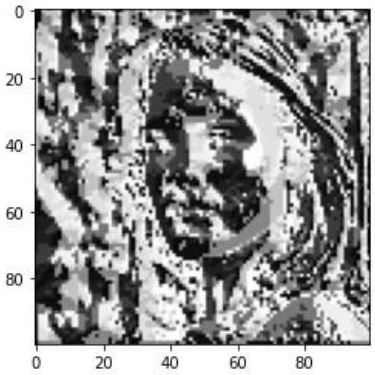
OUTPUT TESTING:

After performing the validation testing, the next step is output asking the user about the format required testing of the proposed system, since no system could be useful if it does not produce the required output in the specific format. The output displayed or generated by the system under consideration. Here the output format is considered in two ways. One is screen and the other is printed format. The output format on the screen is found to be correct as the format was designed in the system phase according to the user needs.

IV. RESULTS AND DISCUSSION



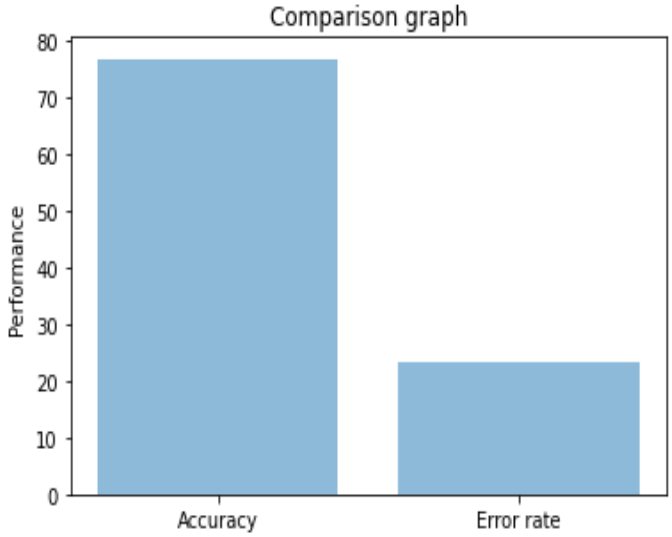





```
------------------------------------------------
Step 3 ------> Extract Biometric Information
------------------------------------------------

 Details

['PersonA', 25, 'Female']
```

```
-----------------------------------------
Step 4 ------> Encryption
-----------------------------------------

The Encrypted String is

b'\x896\xa6\x03\x8cR\xd9t\xbbl)\xd0\x05\x88(2\xc5\x97+\x04P|\xd6"6\xb9\xff,W\xfe\x9a^\xa4?
\x7f!\xf9\xc6\x94\x02\xaa\xb1\xdc\xdf@\xaa:
\x84\xe8E\xa81\xe8\x97\xbb\xdd\xa9\xb8c\x87\x9b\xb8\xdf\x16'
```

```
-------------------------------------------
Step 5 ------> Performance Analysis
-------------------------------------------

1. Accuracy =  76.66666666666667 %

2. Error rate =  23.33333333333333
```




```
-----------------------------------------
Step 6 ------> Identification
-----------------------------------------


The person is AUTHORIZED
```

```
-----------------------------------------
Step 7 ------> Decryption
-----------------------------------------

The decrypted string :

 PersonA 25 Female
```

V. CONCLUSION

In this study, the utilization of machine learning algorithms, particularly Support Vector Machine (SVM), for predicting stock market movements has been explored. By leveraging data collected from global financial markets, the SVM algorithm has demonstrated its effectiveness in analyzing large datasets and making accurate predictions.

One of the key advantages of SVM is its ability to avoid overfitting issues, ensuring robust performance in predicting stock index movements. The models developed in this study have shown high efficiency in forecasting the daily trends of market stocks, outperforming selected benchmarks.

The practical trading models built upon the well-trained predictor have the potential to generate higher profits compared to traditional investment strategies. By incorporating machine learning techniques into stock market prediction, this study contributes to reducing uncertainty in investment decision-making and enhancing the overall efficiency of financial trading strategies.

VI. REFERENCES

[1] European Parliament, EU Regulation 2016/679 of the European Parliament and of the Council (General Data Protection Regulation), 2016.

[2] ISO/IEC JTC1 SC27 Security Techniques, ISO/IEC 24745:2022. Information Technology - Security Techniques - Biometric Information Protection, 2022.

[3] J. Kolberg, P. Bauspieß, M. Gomez-Barrero, C. Rathgeb, M. Durmuth, ¨ and C. Busch, “Template protection based on

homomorphic encryption: Computationally efficient application to iris-biometric verification and identification," in Intl. Workshop on Information Forensics and Security (WIFS), pp. 1–6, IEEE, 2019.

[4] P. Drozdowski, C. Rathgeb, and C. Busch, "Computational workload in biometric identification systems: An overview," IET Biometrics, vol. 8, no. 6, pp. 351–368, 2019.

[5] J. Kolberg, P. Drozdowski, M. Gomez-Barrero, C. Rathgeb, and C. Busch, "Efficiency analysis of post-quantum-secure face template protection schemes based on homomorphic encryption," in Intl. Conf. of the Biometrics Special Interest Group (BIOSIG), pp. 1–4, IEEE, 2020.

[6] V. N. Boddeti, "Secure face matching using fully homomorphic encryption," in Intl. Conf. on Biometrics Theory, Applications and Systems (BTAS), pp. 1–10, IEEE, 2018.

[7] J. J. Engelsma, A. K. Jain, and V. N. Boddeti, "HERS: Homomorphically encrypted representation search," IEEE Trans. on Biometrics, Behavior, and Identity Science (T-BIOM), 2022.

[8] R. Behnia, A. A. Yavuz, and M. O. Ozmen, "High-speed high-security public key encryption with keyword search," in IFIP Conf. on Data and Applications Security and Privacy, pp. 365–385, Springer, 2017.

[9] D. Osorio-Roig, C. Rathgeb, P. Drozdowski, and C. Busch, "Stable hash generation for efficient privacy-preserving face identification," Trans. on Biometrics, Behavior, and Identity Science (TBIOM), 2021.

[10] J. J. Engelsma, A. K. Jain, and V. N. Boddeti, "HERS: Homomorphically encrypted representation search," IEEE Trans. on Biometrics, Behavior, and Identity Science (T-BIOM), 2022.

[11] P. Bauspieß, J. Olafsson, J. Kolberg, P. Drozdowski, C. Rathgeb, and C. Busch, "Improved homomorphically encrypted biometric identification using coefficient packing," in Proc. Intl. Workshop on Biometrics and Forensics (IWBF), 2022.

[12] Y. Zhang, J. Qin, and L. Du, "A secure biometric authentication based on peks," Concurrency and Computation: Practice and Experience, vol. 28, no. 4, pp. 1111–1123, 2016.

[13] X. Zhang, C. Huang, D. Gu, J. Zhang, and H. Wang, "Bib-mks: Postquantum secure biometric identity-based multi-keyword search over encrypted data in cloud storage systems," IEEE Transactions on Services Computing, 2021.

[14] P. Drozdowski, F. Stockhardt, C. Rathgeb, D. Osorio-Roig, and C. Busch, "Feature fusion methods for indexing and retrieval of biometric data: Application to face recognition with privacy protection," IEEE Access, vol. 9, pp. 139361–139378, October 2021.

[15] P. Bauspieß, J. Olafsson, J. Kolberg, P. Drozdowski, C. Rathgeb, and C. Busch, "Improved homomorphically encrypted biometric identification using coefficient packing," in IEEE Intl. Workshop on Biometrics and Forensics (IWBF), 2022.

[16] A. Dantcheva, P. Elia, and A. Ross, "What else does your biometric data reveal? a survey on soft biometrics," Trans. on Information Forensics and Security, vol. 11, no. 3, pp. 441–467, 2015.

[17] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in Intl. Conf. on the Theory and Applications of Cryptographic Techniques, pp. 506–522, Springer, 2004.

[18] C. Bosch, P. Hartel, W. Jonker, and A. Peter, "A survey of provably se- ¨ cure searchable encryption," ACM Computing Surveys (CSUR), vol. 47, no. 2, pp. 1–51, 2014.

[19] J. Daemen and V. Rijmen, The design of Rijndael, vol. 2. Springer, 2002.

[20] J. Phillips, H. Moon, S. Rizvi, and P. Rauss, "The FERET evaluation methodology for face-recognition algorithms," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 22, no. 10, pp. 1090–1104, 2000.