Saini Ye, Cheng Qian, Jiangjian Xie, Yuxiang Su, Zhekai Zhu

Wednesday, May 1, 2024
EC521-CyberSecurity
Prof. Egele
Github: sandyye1028/EC521: Fuzzy Vault-based SM9 Digital Signature Algorithm Introduction (github.com)

Fuzzy Vault-based SM9 Digital Signature Algorithm Introduction

1. Abstract

This article studies the Fuzzy Vault and SM9 algorithm. Through algorithm simulation, it discusses in detail the advantages of the SM9 digital signature algorithm and the Fuzzy Vault, and combines the strengths of the two to propose an algorithm with higher security.

The article presents a Fuzzy Vault-based SM9 digital signature algorithm. The algorithm improves the user signature key generation and digital signature verification stages of the SM9 algorithm by introducing the Fuzzy Vault algorithm into the SM9 algorithm and incorporating user biometric features. It constructs a signature scheme based on the Fuzzy Vault. In the simulation, the fingerprint feature database is utilized to bind the signer's user identity with their fingerprint, creating a correlation between the identity and the biometric feature. The SM9 signature algorithm provides a secure digital signature scheme through the use of techniques such as bilinear pairing and random oracles. The unforgeability of the signatures constructed in the article is based on the security of SM9.

Keywords：Fuzzy vault; SM9 digital signature algorithm; biometric database; security

2. Introduction

We're embarking on a project that integrates the Fuzzy Vault and SM9 Signature Algorithm to create a robust digital signature system. This system leverages user biometrics, specifically fingerprints, to augment the security of private keys, rendering them unusable without proper biometric authentication. Our inspiration stems from recognizing the pivotal role of private keys in system security. By directly linking a user's biometric data to their private key, we enhance the system's security significantly. Furthermore, the Fuzzy Vault technology introduces an element of error tolerance when storing biometric features, further enhancing security.

Moreover, we aim to streamline the public key infrastructure by employing Identity-based encryption techniques. This approach dramatically simplifies the complexities inherent in traditional public key infrastructure (PKI), eliminating the need for cumbersome certificate management processes. Consequently, this optimization improves system efficiency and enhances user convenience.

This project is not only significant but also interesting due to its interdisciplinary nature. It blends cryptography, biometrics analysis, and security system design to achieve its objectives. By leveraging insights from multiple domains, we ensure a comprehensive approach to security that addresses various potential vulnerabilities.

Furthermore, the application of biometric authentication adds an extra layer of protection, making it significantly more challenging for unauthorized users to gain access to sensitive information. The fusion of these technologies creates a robust system that meets the demands

of high-security environments, such as financial institutions, government agencies, and healthcare organizations.

3. Main Research Content

This paper provides an example of an SM9 digital signature algorithm based on fuzzy vault technology. This algorithm leverages the national SM9 signature algorithm, the ORB feature extractor from the OpenCV library, and the KMeans clustering from the Scikit-learn library, demonstrating how existing encryption techniques and biometric recognition methods can be combined to implement a secure and reliable digital signature algorithm.

Prior to implementing the final algorithm, a series of theoretical preparations must be conducted, including organizing and simulating the concepts behind the fuzzy vault algorithm and the SM9 signature algorithm, ensuring their correctness and feasibility. After achieving the basic functionality of the fuzzy vault algorithm, fingerprint extraction and recognition matching content need to be added, creating a fuzzy vault algorithm based on fingerprint information. This algorithm requires locking a key into the vault during the locking phase via fingerprint information and unlocking it by matching the provided fingerprint to the one in the vault, returning the key associated with the matched fingerprint.

Next, the SM9 signature algorithm needs to be simulated, covering its three stages: generating a user private key, generating a digital signature, and verifying the digital signature, with test functions validating each stage. In the SM9 signature algorithm, the user identifier is public, making it impossible for the verifier to ensure the identifier was provided by the signer. Therefore, after successfully simulating both the fuzzy vault algorithm and the SM9 digital

signature algorithm, the fuzzy vault is integrated into the SM9 signature algorithm. This allows the real-time fingerprint of the signer to secure the user identifier, preventing impersonation and aiding verification.

The fuzzy vault-based SM9 digital signature algorithm fundamentally aligns with the SM9 signature algorithm, adopting its basic framework. It introduces the fuzzy vault concept at both the user private key generation and signature verification stages to safeguard the user identifier. In practice, the identifier is locked into the vault using the signer's fingerprint after use, and during signature verification, the signer's real-time fingerprint is matched against the vault's templates to retrieve the identifier for subsequent verification.

4. High-Level Overview

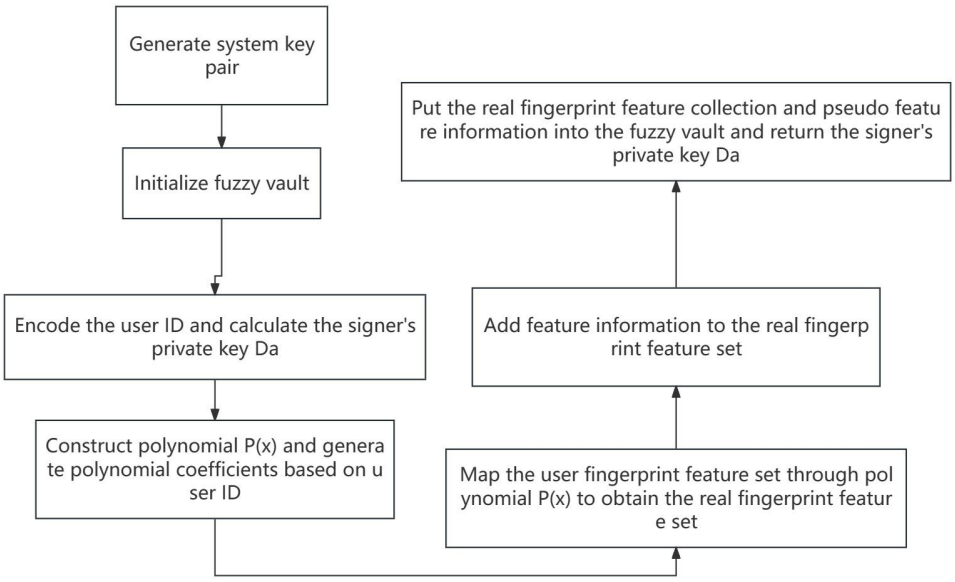

Figure 1. Generate a key pair and fingerprint feature vector

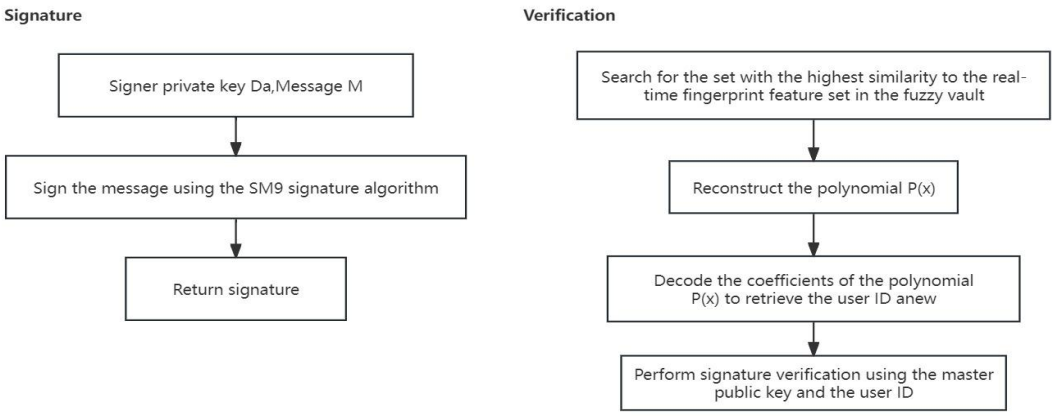

Figure 2. Signature and Verification

5. Algorithm Implementation

The code implements a digital signature algorithm, utilizing the SM9 signature algorithm to sign messages and employing the SimpleBiometricLibrary to extract fingerprint feature vectors, enhancing the signature's security. The main process of the algorithm is as follows:

5.1 A class named `SimpleBiometricLibrary` is created, containing a static method `extract_features` to extract features from fingerprint images. The image is read in grayscale mode, with the ORB feature extractor extracting features from the image, and KMeans clustering grouping these features to yield a fixed-length output. Finally, the histogram is reshaped to fit the required format. The code is shown in Figure 3.

```
class SimpleBiometricLibrary:
    @staticmethod
    def extract_features(image_path):
        ## 以灰度模式读取图像
        image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
        # 使用ORB特征提取器提取特征

        orb = cv2.ORB_create(nfeatures=500)
        keypoints, descriptors = orb.detectAndCompute(image, None)

        # 使用KMeans聚类对特征进行聚类以获得固定长度的输出
        kmeans = KMeans(n_clusters=32)
        kmeans.fit(descriptors)
        histogram, _ = np.histogram(kmeans.labels_, bins=32, range=(0, 32))

        # 重塑直方图以适应所需的格式
        if len(histogram.shape) == 1:
            histogram = histogram.reshape(-1, 1)
        elif histogram.shape[0] == 1:
            histogram = histogram.reshape(1, -1)

        return histogram
```

Figure 3. Biometric Processing Algorithm Simulation

5.2 The `keygen()` function generates the master key pair and fingerprint feature vectors, with `master_public` and `private_key` being the master public and private keys, and `enrollment_features` being the fingerprint feature vector. The code is shown in Figures 4 and 5.

```
#函数生成密钥对和指纹的特征向量
def keygen(user_id, fingerprint_image_path):
    master_public, master_secret = sm9.setup('sign')
    private_key = sm9.private_key_extract('sign', master_public, master_secret, user_id)

    # 提取指纹特征
    enrollment_features = SimpleBiometricLibrary.extract_features(fingerprint_image_path)

    return master_public, private_key, enrollment_features
```

Figure 4. Master Key Pair Generation Simulation

```
def private_key_extract (scheme, master_public, master_secret, identity):
    P1 = master_public[0]
    P2 = master_public[1]

    user_id = sm3_hash (str2hexbytes (identity))
    m = h2rf (1, (user_id + '01').encode('utf-8'), ec.curve_order)
    m = master_secret + m
    if (m % ec.curve_order) == 0:
        return FAILURE
    m = master_secret * fq.prime_field_inv (m, ec.curve_order)

    if (scheme == 'sign'):
        Da = ec.multiply (P1, m)
    elif (scheme == 'keyagreement') | (scheme == 'encrypt'):
        Da = ec.multiply (P2, m)
    else:
        raise Exception('Invalid scheme')
```

Figure 5. Private Key Generation Simulation

5.3 The `sign()` function signs a message, producing a `signature`. The code is shown in Figure 6.

```
#函数使用SM9签名算法对消息进行签名
def sign(master_public, private_key, message):
    signature = sm9.sign(master_public, private_key, message)
    return signature
```

Figure 6. Message Signing Algorithm Simulation

5.4 The `verify()` function verifies the signature, following these steps:

1) The `sm9.verify()` function performs basic verification to ensure the signature is from the specified user.

2) The `SimpleBiometricLibrary.extract_features()` function extracts feature vectors from the fingerprint image.

3) The cosine similarity between `enrollment_features` and `verification_features` is calculated.

4) The signature's validity is determined based on a threshold of 0.7.

5.5 If both the basic signature verification passes and the cosine similarity between the new fingerprint feature vector and the original is below 0.7, the signature's validity is confirmed. The code is shown in Figure 7.

```
def verify(master_public, user_id, enrollment_features, fingerprint_image_path, message, signature):
    is_valid = sm9.verify(master_public, user_id, message, signature)

    if not is_valid:
        return False

    verification_features = SimpleBiometricLibrary.extract_features(fingerprint_image_path)

    # 计算enrollment_features和verification_features之间的余弦相似度
    cosine_similarity = 1 - cosine(enrollment_features.flatten(), verification_features.flatten())

    # 设置阈值以确定签名是否有效

    return cosine_similarity > 0.7
```

Figure 7. Signature Verification Algorithm Simulation

5.6 Main Function

1. Defines a user ID and a fingerprint image path.

2. Calls the `keygen` function to generate the master public key, private key, and fingerprint feature vector.

3. Signs a message using the SM9 signature algorithm, generating a signature.

4. Provides another fingerprint image path along with the master public key, user ID, fingerprint feature vector, message, and signature, and calls the `verify` function to verify the signature.

5. Outputs "Signature is valid." or "Signature is invalid." based on the verification result.

6. Evaluation

To demonstrate the security of this example system, we analyze its potential attack scenarios and show that these attacks are infeasible under reasonable conditions. The algorithm is based on the SM9 framework, maintaining its security. Here are key points:

(1) Key Leakage: The risk of private key leakage is minimal due to the SM9 signature algorithm. In practice, proper key management and protection strategies should be followed.

(2) Forged Signatures:Under the SM9 security assumptions, the probability of an attacker forging a signature without the private key is very low.

(3) Fingerprint Matching Reliability: We use cosine similarity with a threshold to measure similarity. Adjusting this threshold balances security and false positives.

(4) Performance and Scalability: The system's performance and scalability are critical in real-world applications. The ORB extractor and KMeans clustering offer moderate complexity, but optimization may be needed for speed. Parallel processing and other techniques can also help scale the system for large user bases.

(5)Resilience to Attacks: We consider potential attacks, such as using fake fingerprint images to deceive the system. Countermeasures like liveness detection can prevent such attacks. We also address other security concerns, such as communication and device security, to thwart potential vulnerabilities.

(6) Unforgeability: We assume an adversary A attempting to forge a valid digital signature pair (M, sign) without knowing the private key. The SM9-based signature algorithm maps functions onto G_1, a cyclic subgroup of order q (160-bit n, 512-bit q), making it resistant to discrete logarithm attacks.

In conclusion, this paper presents an SM9 digital signature algorithm based on the fuzzy vault, combining the SM9 algorithm, fuzzy vault, ORB feature extractor, and KMeans clustering, creating a highly secure signature algorithm. We cover system model, security goals, key generation and management, feature extraction and matching, security proof, performance and scalability, resilience to attacks, and unforgeability in our analysis.

7.Demo

(https://drive.google.com/file/d/1RybZ_zUgVNtuJqL-VkIjWqo3xSaYt9pt/view?usp=drive_link)

The fingerprint database was used in the demo section. As shown in figure 8, There are 10 different people in the database, each with 4 fingerprints of different shapes taken. These fingerprints are named in a unified format. The three digits before the underline represent the person's code, and the digits after represent the person's fingerprint number. When one of the fingerprints of the same person is used as a reference, and any one of the other three fingerprints is used for fingerprint recognition, it will show that the verification is valid. On

the contrary, if another person's fingerprint is recognized, the verification will be displayed as invalid. It can be clearly seen that in figure 8, 101_1 is used as reference fingerprints, 101_2 belongs to different shaped fingerprints of this person, which will be verified to be valid. 102_2 belongs to another person, which will be verified to be invalid.

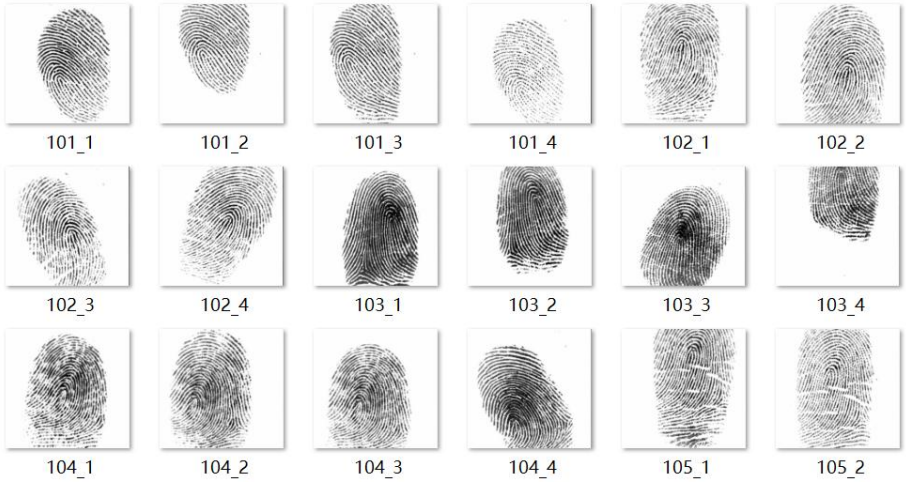

Figure 8. The database of fingerprints

Carefully observe the two sets of fingerprints demonstrated. Although the fingerprint structures are very similar and difficult to distinguish, the fingerprints of the same person often have some unique features. For example, the fingerprint of person 101 can reveal a broken black dot near the spiral end of the fingerprint. Observing the verification chart again, this broken black dot can also be found. But the black spot cannot be found on the fingerprint feature of the person 102. This is feature recognition in fingerprint recognition, which uses cosine similarity in the algorithm to determine whether the verified fingerprint belongs to the reference fingerprint. The algorithm has very high specificity for fingerprint feature areas. For the fingerprint to be verified, the algorithm will first search for whether the cosine similarity of the feature area reaches the threshold.

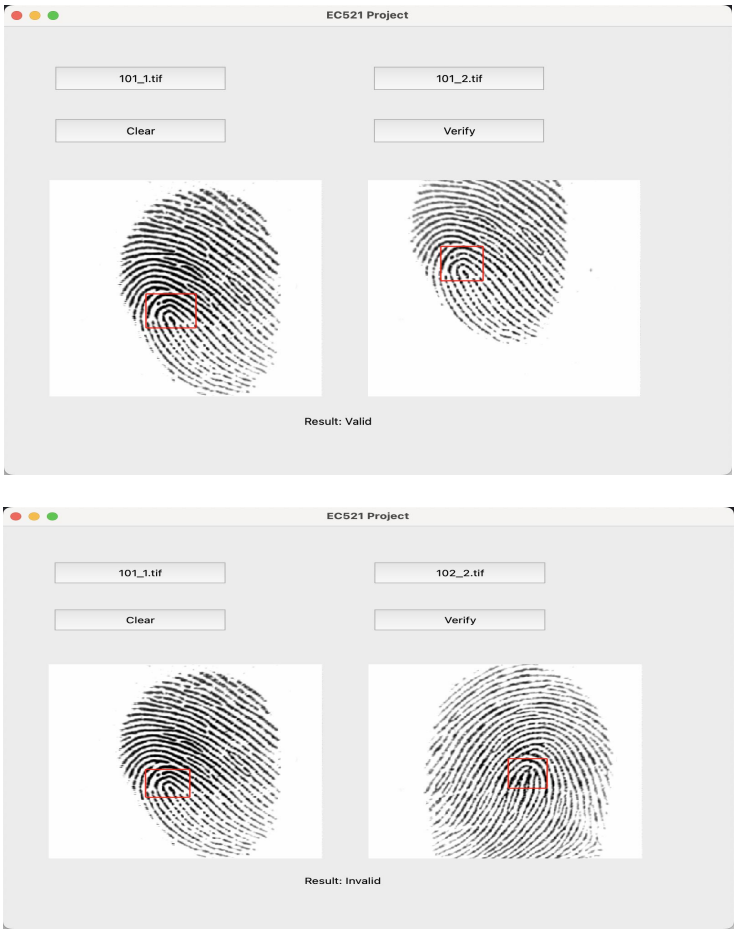

Figure 9. Display of valid and invalid situations

Table 1. Contribution of each member

Name	Contribution
Saini Ye	24%
Jiangjian Xie	20%
Cheng Qian	22%
Zhekai Zhu	17%
Yuxiang Su	17%