This is an accepted version of a paper published in Elsevier Information Fusion.

If you wish to cite this paper, please use the following reference:

T. Murakami, T. Ohki, K. Takahashi, Optimal sequential fusion for multibiometric cryptosystems, Elsevier Information Fusion (Special Issue on Information Fusion in Biometrics), vol.32, pp.93-108, 2016.
http://dx.doi.org/10.1016/j.inffus.2016.02.002

Optimal Sequential Fusion for Multibiometric Cryptosystems

Takao Murakami[a,*], Tetsushi Ohki[a], Kenta Takahashi[b]

[a] *National Institute of Advanced Industrial Science and Technology, Tokyo 135-0064, Japan*

[b] *Hitachi, Ltd., Yokohama 244-0817, Japan*

Abstract

Biometric cryptosystems have been widely studied in the literature to protect biometric templates. To ensure sufficient security of the biometric cryptosystem against the offline brute-force attack (also called the FAR attack), it is critical to reduce FAR of the system. One of the most effective approaches to improve the accuracy is multibiometric fusion, which can be divided into three categories: feature level fusion, score level fusion, and decision level fusion. Among them, only feature level fusion can be applied to the biometric cryptosystem for security and accuracy reasons. Conventional feature level fusion schemes, however, require a user to input all of the enrolled biometric samples at each time of authentication, and make the system inconvenient.

In this paper, we first propose a general framework for feature level sequential fusion, which combines biometric features and makes a decision each time a user inputs a biometric sample. We then propose a feature level sequential fusion algorithm that can minimize the average number of input, and prove its optimality theoretically. We apply the proposed scheme to the fuzzy commitment scheme, and demonstrate its effectiveness through experiments using the finger-vein dataset that contains 6 fingers from 505 subjects. We also analyze the security of the proposed scheme against various attacks: attacks that exploit the relationship between multiple protected templates, the soft-decoding attack, the statistical attack, and the decodability attack.

*Corresponding author

Email addresses: takao-murakami@aist.go.jp (Takao Murakami), tetsushi.ohki@aist.go.jp (Tetsushi Ohki), kenta.takahashi.bw@hitachi.com (Kenta Takahashi)

Keywords: multimodal biometrics, sequential fusion, biometric cryptosystems, sequential probability ratio test

1. Introduction

Protecting biometric data is a critical issue in biometric authentication systems, since biometric information such as faces, fingerprints, irises and vein patterns are personal and privacy information. Although many conventional systems rely on standard encryption to protect biometric templates, the encrypted templates have to be decrypted at the time of verification to perform pattern matching, and thus a skilled attacker who aims at this timing can break them. Other systems use tamper-proof devices, such as hardware tokens, to protect biometric templates. However, these systems require a user to possess a token or to use limited devices in which a user's template is enrolled. These limitations can reduce the usability of the authentication systems.

To solve the problem fundamentally, various studies have been made on so-called *biometric template protection* techniques that keep biometric templates secret in the algorithm level even during verification. Among them, *biometric cryptosystems* have particularly attracted attention, in which a biometric feature is used as a source of a secret key of cryptosystems and verified without being revealed using cryptographic techniques [1, 2, 3, 4]. Biometric cryptosystems typically take a strategy of embedding a secret key into a biometric feature yielding so-called *auxiliary data* (AD), and releasing the secret key from the AD using a genuine biometric feature. For authentication, the secret key is verified using a *pseudo identifier* (PI), which is a public key or a hash value of the secret key. A set (AD, PI) is referred to as a *protected template*, and is enrolled into a database or smart card, along with a user ID.

There are two possible models for storing a protected template (AD, PI) in the biometric cryptosystem. The first model is to store AD and PI separately. For example, AD is stored into a smart card of a user, while PI is stored into the database in the authentication server. Then, even if either of AD or PI is compromised, we can restore the security by updating both of the two. In this model, the security requirement for the biometric cryptosystem is that it is sufficiently hard to guess a biometric feature or impersonate a user by use of either of AD or PI.

The second model is to store both AD and PI into a single place (e.g. the authentication server). In this case, it can happen that both of AD and PI are compromised from the place at the same time. Thus, the security requirement for the biometric cryptosystem in this model is that it is sufficiently hard to guess a biometric feature or to impersonate a user even if both of AD and PI are available to the adversary. If this security requirement is satisfied, we can realize an authentication system where a user is not required to have a smart card that stores AD secretly nor required to present the smart card at the authentication phase. We can even disclose the protected template (AD, PI) to the public, or share it across multiple organizations. Thus, we refer to this model as a *public template model*. The aim of this paper is to realize this model in a secure manner.

In the public template model, it is required that the biometric cryptosystem defends against the *offline brute-force attack (also called the FAR attack);* the attack where the adversary prepares a large number of biometric features, and matches each of them with a protected template (AD, PI) offline to find a biometric feature that successes in authentication (i.e. a correct secret key is reproduced). Let α be FAR (False Acceptance Rate) of the biometric cryptosystem. Then, this attack results in success after matching on average $1/(2\alpha)$ biometric features. Thus, it is required that FAR α is sufficiently small to realize the public template model.

It is difficult to achieve sufficient security (comparable to typical cryptographic keys) against the offline brute-force attack using only one source of biometric information (e.g. one finger-vein). For example, to achieve at least 64-bit security, FAR should be smaller than $2^{-64} \simeq 5.4 \times 10^{-20}$, whereas FAR of many commercial biometric authentication systems is $10^{-5} \sim 10^{-7}$.

Multibiometric fusion [5], which combines multiple sources of biometric information (e.g. fingerprint, face, and iris; multiple finger-veins) for authentication, is expected to fill this gap. Some studies also applied multibiometric fusion to the biometric cryptosystem [6, 7, 8, 9]. However, most of the conventional fusion schemes (including [6, 7, 8, 9]) require users to input *all* the enrolled biometric information (e.g., 10 finger-veins) at the authentication phase. Such schemes are referred to as *parallel fusion schemes* [10], and cause inconvenience for users.

To make the authentication system convenient for users, some studies proposed a *sequential (or serial) fusion scheme* [11, 10, 12, 13], which makes a decision each time a user inputs his/her biometric sample. Some sequential fusion schemes [12, 13] can optimize the trade-off between the accuracy and

the number of biometric inputs required before acceptance, and thus realize a secure and convenient biometric system. These schemes combine multiple sources of biometric information at the matching score level (i.e. score level fusion) to make such an optimal decision.

However, it must be noted that biometric cryptosystems must not output a matching score for a security reason; an attacker can reconstruct the original biometric feature by using the score as a clue (i.e. *hill-climbing attack* [14]). Thus, the conventional sequential fusion schemes using scores cannot be applied to biometric cryptosystems. To reduce FAR as much as possible without disclosing scores during verification, we should construct a multibiometric cryptosystem based on *feature level fusion*, which combines biometric features into a single (but large-sized) feature. To the best of our knowledge, no studies have attempted to develop a sequential fusion scheme at the feature level.

In this paper, we propose an optimal sequential fusion scheme at the feature level to realize secure and convenient biometric cryptosystems in the public template model. The main contributions are as follows:

- We firstly propose a general framework for feature level sequential fusion. To the best of our knowledge, this is the first framework that enables secure and convenient biometric cryptosystems in the public template model (Section 3.1).

- Based on this framework, we secondly propose an optimal algorithm for feature level sequential fusion. This algorithm is based on the SPRT (Sequential Probability Ratio Test) [15, 16], and minimizes the average number of input while keeping FAR less than or equal to the required value. We also provide a formal proof of this optimality (Section 3.2 - 3.4).

- We apply the proposed scheme to the fuzzy commitment scheme, and demonstrate its effectiveness through experiments using the finger-vein dataset in [17], which contains 6 fingers from 505 subjects (33298 finger-vein images in total) (Section 4).

- We finally analyze the security of the proposed scheme against various attacks: attacks that exploit the relationship between multiple protected templates, the soft-decoding attack [18, 19], the statistical attack [18, 20], and the decodability attack [21, 22] (Sections 5 and 6).

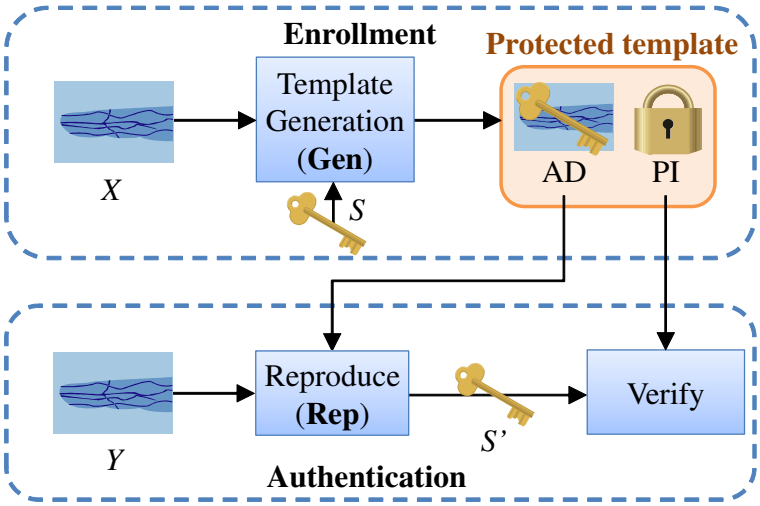

Figure 1: Architecture of a biometric cryptosystem (X, Y: biometric feature, S, S': secret key, AD: auxiliary data, PI: pseudo identifier). If $\mathrm{dis}(X, Y) \leq \delta$, then $S = S'$ (δ: distance threshold).

2. Related work

2.1. Biometric cryptosystems

Biometric cryptosystems have been widely studied in the literature [23]. Examples include fuzzy commitment [3], fuzzy vault [2], and fuzzy extractor [1].

Figure 1 shows an architecture of a typical biometric cryptosystem (we also describe the fuzzy commitment scheme [3] as an example of the biometric cryptosystem in Section 2.2). At the enrollment phase, a template generation algorithm **Gen** receives a biometric feature X from a user. It then encodes a secret key S (typically a random string) using an ECC (error-correcting code), and embeds it into X to make AD (auxiliary data; it is also called helper data). AD is designed so that if the user inputs a biometric feature Y that is sufficiently close to X according to some distance metric dis (i.e. $\mathrm{dis}(X, Y) \leq \delta$ for a predetermined distance threshold δ), a secret key S is reproduced from AD. **Gen** also makes PI (pseudo identifier), which is used to verify the reproduced secret key. PI is a public key, or a hash value of S. A protected template (AD, PI) is enrolled into a database (or smart card), along with a user ID.

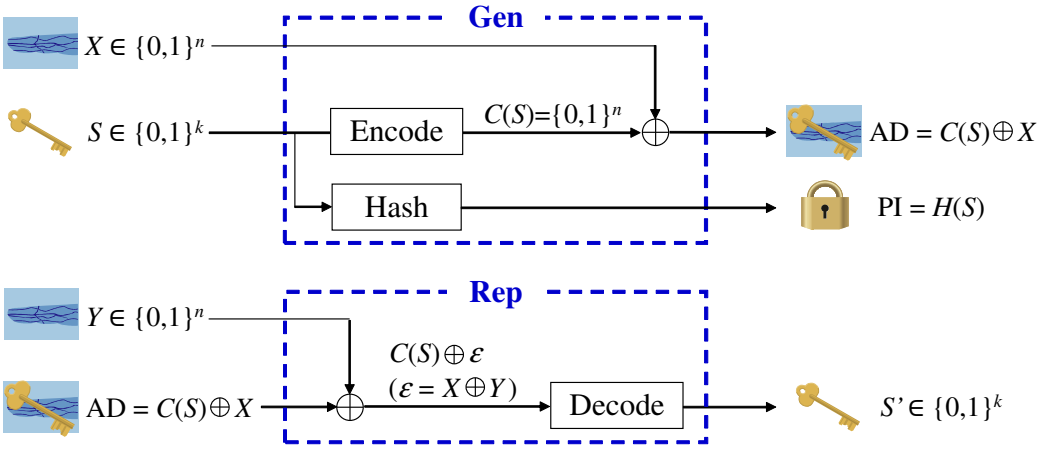

Figure 2: Realization of the template generation algorithm **Gen** and the reproduce algorithm **Rep** using the fuzzy commitment scheme (PI is a hash value of S).

At the authentication phase, a reproduce algorithm **Rep** receives a new biometric feature Y and AD, and reproduces a secret key S' from AD using Y. As described above, if X and Y are sufficiently close (i.e. $\mathrm{dis}(X, Y) \leq \delta$), a correct secret key is reproduced (i.e. $S = S'$). The system authenticates a user using S' and PI.

As described in Section 1, we do not have to store the protected template (AD, PI) secretly in the public template model. We can even disclose it to the public, or share it across multiple organizations. To achieve this ultimate goal, FAR needs to be sufficiently small.

2.2. Fuzzy Commitment

We now briefly explain the fuzzy commitment scheme [3], which was proposed by Juels and Wattenberg in 1999, as an example of the biometric cryptosystem. Figure 2 shows the realization of the template generation algorithm **Gen** and the reproduce algorithm **Rep** using the fuzzy commitment scheme. This scheme assumes that a biometric feature is represented as a binary string, and uses the Hamming distance as a distance metric. Examples of such biometric features include the IrisCode [24, 25]. We also extract a finger-vein feature represented as a binary string in our experiments in Section 4.

At the enrollment phase, the template generation algorithm **Gen** receives a biometric feature $X \in \{0, 1\}^n$ that is represented as an n-bit binary string.

Gen generates a codeword $C(S) \in \mathcal{C} \subseteq \{0,1\}^n$ ($\mathcal{C}$ is a set of codewords) by passing a secret key $S \in \{0,1\}^k$ (k-bit random string) through an error-correcting encoder. Note that $C(S)$ is a codeword of an (n, k, d_{min}) linear ECC (error-correcting code), where n, k, and d_{min} ($= 2\delta + 1$) represent the codeword length, the number of information symbols, and the minimum distance, respectively (it can correct up to δ errors). It then generates auxiliary data (also called a "commitment") AD as $\text{AD} = X \oplus C(S)$ ($\oplus$ denotes a bitwise XOR operator). **Gen** also makes PI by computing the hash value $H(S)$ of the secret key S (or a public key corresponding to S), and stores (AD, PI) into a database along with a user ID.

At the authentication phase, the reproduce algorithm **Rep** receives a new biometric feature $Y \in \{0,1\}^n$ and AD. **Rep** computes S' by passing $Y \oplus \text{AD} = C(S) \oplus \epsilon$ ($\epsilon = X \oplus Y$) through an error-correcting decoder. If the Hamming distance between X and Y is less than or equal to δ (i.e. if $||\epsilon|| \leq \delta$, where $|| \cdot ||$ denotes the Hamming weight), a correct secret key is reproduced (i.e. $S = S'$). The system can determine whether the user is a genuine user or an impostor by checking whether $\text{PI} = H(S')$ or not.

2.3. Multibiometric fusion

A number of studies have been made on multibiometric fusion, which combines multiple sources of biometric information, to improve accuracy of biometric authentication systems [5]. Some studies also applied multibiometric fusion to biometric cryptosystems [6, 7, 8, 9].

According to the type of information sources to be combined, multibiometric fusion can be divided into the following categories: feature level fusion, score level fusion, and decision level fusion. They combine multiple biometric features, scores, and decision results (i.e. match/non-match), respectively. Examples of feature level fusion include feature concatenation [6, 7, 8, 9, 26] (e.g. concatenating feature vectors or binary strings), and those of score level fusion include Bayesian fusion [27], SVM-based fusion [28], and combination rules such as the sum rule, product rule, and max rule [29]. Examples of decision level fusion include the AND rule [11] and majority voting [30].

Most of the conventional multibiometric fusion schemes are *parallel fusion schemes* that make a decision after a user inputs all of the enrolled biometric samples. The major problem of these schemes is that it can make the system inconvenient. For example, a fingerprint sensor generally allows a user to input only one finger at a time. If the system requires the user to input all

of 10 fingers to such a sensor at the authentication phase, the user has to repeat the input operation as many as 10 times.

To solve this problem, some studies proposed a *sequential (or serial) fusion scheme* that makes a decision each time a user inputs his/her biometric sample [11, 10, 12, 13]. Since this scheme uses less biometric inputs, it can offer a more convenient way to improve accuracy. The simplest sequential fusion scheme is the OR rule [11], which makes a match/non-match decision by comparing a matching score (similarity or distance) to the threshold each time a user inputs his/her biometric sample. If some biometric sample is decided as "match", this rule accepts a user and terminates the authentication process. Thus, the OR rule is a kind of decision level sequential fusion. Since binary decision results (i.e. match/non-match) have much less information than scores or features, decision level fusion has only a limited effect in improving accuracy.

Some studies proposed a sequential fusion scheme that uses scores as information sources (i.e. score level sequential fusion). Examples include the likelihood-ratio rule [13] and the posterior probability rule [12]. These rules are based on SPRT (Sequential Probability Ratio Test) [15, 16], which minimizes the average number of observations among all binary hypothesis tests with the same error probabilities, or MSPRT (Multi-hypothesis SPRT) [31], which is a multi-hypothesis version of SPRT. Thus, they can minimize the average number of biometric inputs among all sequential fusion schemes with the same error probabilities in verification (or identification).

However, the biometric cryptosystem must not output a matching score (similarity or distance) but output only a decision result (i.e. match/non-match) at the matching phase. This is because the adversary can carry out a *hill-climbing attack* [14] based on the matching score; he/she can construct a biometric feature that is sufficiently close to the feature of the target user by changing a biometric feature little by little based on a matching score as a clue. For the same reason, the threshold at the matching phase must not be changed in the biometric cryptosystem. Unfortunately, since both the likelihood-ratio rule and posterior probability-based rule mentioned above use matching scores as information sources (i.e. score level fusion), they cannot be applied to the biometric cryptosystem.

Table 1 shows the classification of conventional multibiometric fusion schemes. Unfortunately, parallel fusion makes the system inconvenient, decision level fusion has only a limited effect in improving accuracy, and score level fusion cannot be applied to biometric cryptosystems, as mentioned

Table 1: Classification of conventional multibiometric fusion schemes. Parallel fusion makes the system inconvenient. Decision level fusion has only a limited effect in improving accuracy. Score level fusion cannot be applied to the biometric cryptosystem.

	Feature level	Score level	Decision level
Parallel	feature concatenation [6, 7] [8, 9, 26], etc.	Bayes [27], SVM [28], combination rules (e.g. sum, product, max) [29], etc.	AND rule [11], majority voting [30], etc.
Sequential		likelihood-ratio rule [13], posterior probability rule [12], etc.	OR rule [11], etc.

above. Although some studies applied feature level fusion (feature concatenation) to biometric cryptosystems [6, 7, 8, 9], all of their approaches fall under parallel fusion that makes the system inconvenient. To the best of our knowledge, no studies have attempted to develop a sequential fusion scheme at the feature level.

3. Optimal sequential fusion for multibiometric cryptosystems

We propose an optimal sequential fusion scheme at the feature level for secure and convenient biometric cryptosystems. We first propose a framework for feature level sequential fusion (Section 3.1). We then propose a sequential fusion algorithm that minimizes the average number of inputs while keeping FAR less than or equal to the required value, and prove its optimality under some assumptions (Section 3.2 - 3.4). We finally describe how to set parameters k (the number of information symbols) and d_{min} (the minimum distance) of the error-correcting code in the proposed sequential fusion scheme (Section 3.5).

3.1. Feature level sequential fusion

Let T be the number of modalities (or fingers) used in the multibiometric cryptosystem. The conventional feature level fusion schemes for the biometric cryptosystem [6, 7, 8, 9] create a large biometric feature by combining all of T biometric features at the enrollment phase (e.g. Tn-bit binary string by concatenating T biometric features each of which is represented as an n-bit string), and makes a protected template based on the large biometric feature. However, it must be noted that to reproduce a secret key from AD that is

composed of T biometric features, a user has to input all of the T biometric samples in the same order as the enrollment. In other words, we cannot construct a sequential fusion scheme using only the large biometric feature that is composed of T biometric features.

We therefore propose to create a large biometric feature (and the corresponding protected template) for every possible combination of biometric features. If the input order at the authentication phase is determined in advance, there are T possible combinations. If the input order is not determined, there are $\sum_{t=1}^{T} {}_T C_t = 2^T - 1$ possible combinations[1]. After a user inputs the t-th biometric sample at the authentication phase, the system combines the t biometric features into one large biometric feature and matches it with the corresponding protected template. If a correct secret key is reproduced, the system accepts the user and terminates the authentication process.

Figure 3 shows the proposed framework for feature level sequential fusion. Here, X^t and Y^t $(1 \leq t \leq T)$ denote the t-th biometric feature at the enrollment phase and authentication phase, respectively. In this example, the input order is determined to be $Y^1 \rightarrow Y^2 \rightarrow \cdots \rightarrow Y^T$. Thus, the system creates a large biometric feature and a protected template for each of T possible combinations. f_t $(1 \leq t \leq T)$ denotes a mapping function that combines $X^1, \cdots, X^t$ into a large biometric feature $X_t = f_t(X^1, \cdots, X^t)$ (e.g. bit concatenation). After creating a large biometric feature X_t $(1 \leq t \leq T)$ using f_t, the system generates a secret key S_t and creates a protected template $(\mathrm{AD}_t, \mathrm{PI}_t)$ from X_t and S_t using a template generation algorithm $\mathbf{Gen}_t$. Then

[1]This is because the system can assign a number to each modality, and combine biometric features in ascending order of the numbers. For example, when the system uses fingerprint, face, and voice ($T = 3$), it can assign numbers 1, 2, and 3 to fingerprint, face, and voice, respectively, and combine them in this order. Then, there are $\sum_{t=1}^{3} {}_3 C_t = 2^3 - 1 = 7$ possible combinations in this case: [fingerprint], [face], [voice], [fingerprint $\rightarrow$ face], [fingerprint $\rightarrow$ voice], [face $\rightarrow$ voice], and [fingerprint $\rightarrow$ face $\rightarrow$ voice]. The system creates a protected template for each of the 7 combinations. Similarly, the system combines biometric features in ascending order of the numbers at the authentication phase (in the case of multiple fingerprints (or finger-veins), the user indicates which type of finger is input at the authentication phase (by pressing a button, for example), since a single sensor is used in common). For example, if the user inputs voice, fingerprint, and face at the 1st, 2nd, and 3rd input, respectively, the system creates large biometric features as follows: [voice], [fingerprint $\rightarrow$ voice], and [fingerprint $\rightarrow$ face $\rightarrow$ voice] at the 1st, 2nd, and 3rd input, respectively. Then the system compares each of them with the corresponding template.

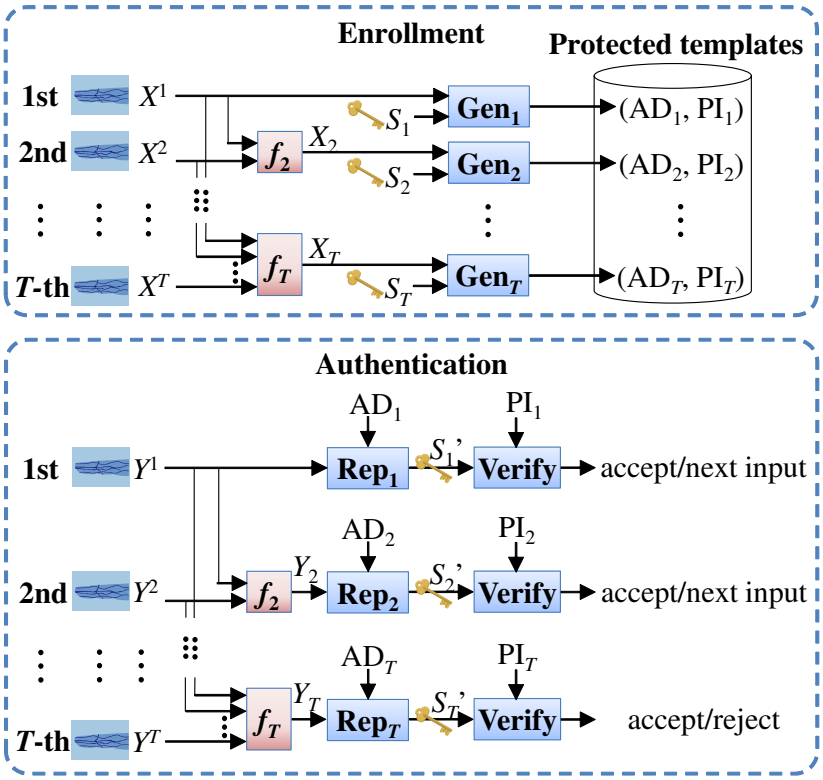

Figure 3: Proposed framework for feature level sequential fusion.

it stores $(\mathrm{AD}_t, \mathrm{PI}_t)$ $(1 \leq t \leq T)$ into the database.

After a user inputs Y^t $(1 \leq t \leq T)$ at the authentication phase, the system combines $Y^1, \cdots, Y^t$ into a large biometric feature $Y_t = f_t(Y^1, \cdots, Y^t)$, and reproduces a secret key S'_t from Y_t and AD_t using a reproduce algorithm $\mathbf{Rep}_t$. Then it verifies whether $S'_t = S_t$ or not using PI_t, and makes a decision as follows: if $S'_t = S_t$, accept the user and terminate the authentication process; otherwise, require the next biometric input (if $t = T$, reject).

This framework improves accuracy of the biometric cryptosystem while keeping down the number of biometric inputs. The following question now arises: how can we minimize the number of biometric inputs? We propose such an optimal sequential fusion algorithm by appropriately setting a distance threshold in creating each protected template. More specifically, we set a distance threshold so that the proposed sequential fusion algorithm is equivalent to SPRT [15, 16]. In the following, we describe the details.

3.2. Assumptions

Assume that the input order at the authentication phase is determined to be $Y^1 \to Y^2 \to \cdots \to Y^T$ for simplicity. The discussion below can also be applied to the case where the input order is not determined.

Suppose that a user (claimant) claims an identity corresponding to the enrolled biometric features $X^1, \cdots, X^T$ at the authentication phase. Let H_1 be a hypothesis that the claimant is a genuine user, and H_0 be a hypothesis that he/she is an impostor. Then, after the t-th input (i.e. after $Y^1, \cdots, Y^t$ are obtained), the posterior probability that the claimant is a genuine user (referred to as a *genuine probability*) can be expressed as $p_t = P(H_1|Y^1, \cdots, Y^t)$.

Let $\mathrm{dis}_t(X_t, Y_t)$ be a distance between $X_t = f_t(X^1, \cdots, X^t)$ and $Y_t = f_t(Y^1, \cdots, Y^t)$ (e.g. hamming distance). In this paper, we make the following assumptions on the distance function dis_t and genuine probability p_t:

(i) For each number of inputs $t = 1, 2, \cdots, T$, there exist a template generation algorithm $\mathbf{Gen}_t$ and a reproduce algorithm $\mathbf{Rep}_t$ such that a correct secret key is reproduced iff $\mathrm{dis}_t(X_t, Y_t) \leq \delta_t$, where δ_t is a distance threshold.

(ii) The genuine probability p_t depends only on the distance $d_t = \mathrm{dis}_t(X_t, Y_t)$. That is, there exists a function ρ_t such that $p_t = \rho_t(d_t)$.

(iii) $\rho_t(\cdot)$ is a monotonically decreasing function. That is, the smaller the distance d_t is, the higher the genuine probability p_t is.

For example, if the fuzzy commitment scheme, which is described in Section 2.2, is used as a biometric cryptosystem (i.e. if biometric features are expressed as n-bit binary strings and the Hamming distance is used as a distance metric) and a bit concatenation is used as f_t, large biometric features are expressed as tn-bit binary strings (i.e. $X_t, Y_t \in \{0,1\}^{tn}$). Thus, the assumption (i) can be satisfied by using an (tn, k, d_{min}) linear ECC and appropriately setting parameters k (the number of information symbols) and d_{min} $(= 2\delta_t + 1)$ (the minimum distance). We describe how to set these parameters in details in Section 3.5.

The assumption (ii) is made in score level fusion schemes that compute posterior probabilities from scores [12, 27]. More specifically, they assume that a matching score (distance) d from a genuine user (genuine score) is generated according to $g_1(d)$, and a matching score from an impostor (impostor score) is generated according to $g_0(d)$, and express the genuine probability p

after observing a biometric feature Y, using Bayes' theorem, as follows:

$$\begin{aligned} p &= P(H_1 \mid Y) = P(H_1 \mid d) && (1)\\ &= \frac{P(d \mid H_1)P_1}{P(d \mid H_1)P_1 + P(d \mid H_0)P_0} && (2)\\ &= \frac{g_1(d)P_1}{g_1(d)P_1 + g_0(d)P_0}, && (3) \end{aligned}$$

where $P_1 = P(H_1)$ and $P_0 = P(H_0)$ (i.e. P_1 and P_0 are prior probabilities). Since it is shown that these schemes work very well through the experiments, we can assume that (ii) is an appropriate assumption.

A distance metric is generally designed so that the smaller the distance d_t is, the higher the genuine probability p_t is. Thus, the assumption (iii) is also an appropriate assumption.

3.3. Algorithm

We now describe the proposed sequential fusion algorithm, which is divided into the enrollment algorithm and the authentication algorithm.

3.3.1. Enrollment algorithm

The enrollment algorithm uses biometric features $X^1, \cdots, X^T$ as input data, and repeats the following procedure for each of $t = 1, 2, \cdots, T$:

1. Combine biometric features $X^1, \cdots, X^t$ into a large biometric feature $X_t = f_t(X^1, \cdots, X^t)$.
2. Set a distance threshold δ_t as follows:

$$\delta_t = \rho_t^{-1}\left(\frac{P_1}{P_1 + P_0\alpha_{max}}\right), \qquad (4)$$

where $P_1 = P(H_1)$, $P_0 = P(H_0)$ (i.e. P_1 and P_0 are prior probabilities), and α_{max} is a required FAR. A function ρ_t can be computed by assuming that a genuine score (distance) d_t is generated from $g_{t,1}(d_t)$ and an impostor score is generated from $g_{t,0}(d_t)$, and express the genuine probability p_t, in the same way as (3), as follows:

$$\begin{aligned} p_t &= \frac{g_{t,1}(d_t)P_1}{g_{t,1}(d_t)P_1 + g_{t,0}(d_t)P_0} && (5)\\ &= \rho_t(d_t). && (6) \end{aligned}$$

In Section 3.4, we prove that by setting a distance threshold in this way, the proposed algorithm minimizes the average number of inputs while keeping FAR less than or equal to α_{max}.

3. Generate a secret key S_t, and create a protected template $(\mathrm{AD}_t, \mathrm{PI}_t) = \mathbf{Gen}_t(X_t, S_t)$. S_t is reproduced from AD_t iff a large biometric feature Y_t such that $\mathrm{dis}_t(X_t, Y_t) \leq \delta_t$ is input.

Then, it outputs $\mathbf{AD} = (\mathrm{AD}_1, \cdots, \mathrm{AD}_T)$ as auxiliary data, and $\mathbf{PI} = (\mathrm{PI}_1, \cdots, \mathrm{PI}_T)$ as pseudo identifiers.

For example, if the fuzzy commitment scheme is used as a biometric cryptosystem (i.e. if biometric features are expressed as n-bit binary strings) and a bit concatenation is used as f_t, $\mathbf{Gen}_t$ receives a large biometric feature $X_t \in \{0,1\}^{tn}$ (tn-bit binary string), and generates a codeword $C(S_t) \in \mathcal{C} \subseteq \{0,1\}^{tn}$ ($\mathcal{C}$ is a set of codewords) by passing a secret key $S_t \in \{0,1\}^k$ (k-bit random string) through an error-correcting encoder. $C(S_t)$ is a codeword of an (tn, k, d_{min}) linear ECC ($d_{min} = 2\delta_t + 1$). It then generates auxiliary data AD_t as $\mathrm{AD}_t = X_t \oplus C(S_t)$, and makes PI_t by computing the hash value $H(S_t)$ of the secret key S_t (or a public key corresponding to S_t).

3.3.2. Authentication algorithm

The enrollment algorithm uses auxiliary data **AD**, pseudo identifiers **PI**, and biometric features $Y^1, \cdots, Y^T$ that are input sequentially as input data, and carries out the following procedure:

1. $t = 1$.
2. Combine biometric features $Y^1, \cdots, Y^t$ into a large biometric feature $Y_t = f_t(Y^1, \cdots, Y^t)$.
3. Reproduce a secret key $S'_t = \mathbf{Rep}_t(Y_t, \mathrm{AD}_t)$.
4. Verify whether $S_t = S'_t$ or not using PI_t.
5. If $S_t = S'_t$, accept a user and terminate the authentication process.
6. If $S_t \neq S'_t$ and $t < T$, increment t by 1 ($t = t + 1$) and go back to the step 2.
7. If $S_t \neq S'_t$ and $t = T$, reject a user and terminate the authentication process.

For example, if the fuzzy commitment scheme is used as a biometric cryptosystem and a bit concatenation is used as f_t, $\mathbf{Rep}_t$ receives a large biometric feature $Y_t \in \{0,1\}^{tn}$ and AD_t ($= X_t \oplus C(S_t)$), and computes S'_t by passing $Y_t \oplus \mathrm{AD}_t = C(S_t) \oplus \epsilon_t$ ($\epsilon_t = X_t \oplus Y_t$) through an error-correcting

decoder. If the Hamming distance between X_t and Y_t is less than or equal to δ_t (i.e. if $||\epsilon|| \leq \delta_t$, where $||\cdot||$ denotes the Hamming weight), a correct secret key is reproduced (i.e. $S_t = S'_t$). The system can determine whether $S_t = S'_t$ by checking whether $\mathrm{PI}_t = H(S'_t)$ or not.

3.4. Optimality

The proposed algorithm minimizes the average number of inputs while keeping FAR less than the required value α_{max}. We prove this optimality by showing that the proposed algorithm is equivalent to SPRT [15, 16].

3.4.1. SPRT

Although the original SPRT, which was proposed by Wald [16], assumes that the observed data are i.i.d. (independent and identically distributed), we describe the algorithm that is extended to the case where the observed data are non-i.i.d [15].

Assume that either of a null hypothesis H_0 and an alternate hypothesis H_1 is true, and the data $Y^1, Y^2, \cdots$ are sequentially observed according to the true hypothesis. Let $p_0(t) = P(Y^1, \cdots, Y^t \mid H_0)$ be the probability that $Y^1, \cdots, Y^t$ are observed when H_0 is true, $p_1(t) = P(Y^1, \cdots, Y^t \mid H_1)$ be the probability that $Y^1, \cdots, Y^t$ are observed when H_1 is true, and $Z_t = p_1(t)/p_0(t)$ be a ratio between the two probabilities (i.e. likelihood-ratio).

After the t-th data Y^t is observed, SPRT computes a likelihood-ratio Z_t and makes the following decision: if $Z_t \geq A$, accept H_1; otherwise if $Z_t \leq B$, accept H_0; otherwise, continue observing data (A and B are thresholds).

It is proved that SPRT minimizes the average number of observations among all binary hypothesis tests with the same error probabilities in the asymptotic case where the error probabilities are sufficiently small [15]. Let α be the probability that H_1 is accepted when H_0 is true, and β be the probability that H_0 is accepted when H_1 is true. It is also proved that SPRT keeps these error probabilities as follows [16]:

$$\alpha \leq 1/A, \quad \beta \leq B. \tag{7}$$

3.4.2. Optimality of the proposed algorithm

We now prove the optimality of the proposed algorithm by showing that it is equivalent to SPRT:

Theorem 1. *Under the assumptions (i)(ii)(iii), the proposed sequential fusion algorithm described in Section 3.3 minimizes the average number of*

inputs while keeping FAR less than or equal to α_{max} in the asymptotic case where the error probabilities (FAR and FRR) are sufficiently small.

Proof. Assume that a user inputs the t-th biometric feature Y^t. The likelihood-ratio after observing $Y^1, \cdots, Y^t$ can be expressed as $Z_t = P(Y^1, \cdots, Y^t \mid H_1)/P(Y^1, \cdots, Y^t \mid H_0)$.

Under the assumption (i), the user is accepted iff the following inequality holds:

$$\mathrm{dis}_t(X_t, Y_t) \leq \delta_t. \tag{8}$$

Under the assumption (ii)(iii), this inequality can be further written as follows:

$$\begin{aligned}
(8) &\Leftrightarrow \rho_t(\mathrm{dis}_t(X_t, Y_t)) \geq \rho_t(\delta_t) && (9)\\
&\Leftrightarrow \rho_t(d_t) \geq \rho_t(\delta_t) && (10)\\
&\Leftrightarrow P(H_1 \mid Y^1, \cdots, Y^t) \geq \rho_t(\delta_t) && (11)\\
&\Leftrightarrow \frac{P(H_1 \mid Y^1, \cdots, Y^t)}{1 - P(H_1 \mid Y^1, \cdots, Y^t)} \geq \frac{\rho_t(\delta_t)}{1 - \rho_t(\delta_t)} && (12)\\
&\Leftrightarrow \frac{P(H_1 \mid Y^1, \cdots, Y^t)}{P(H_0 \mid Y^1, \cdots, Y^t)} \geq \frac{\rho_t(\delta_t)}{1 - \rho_t(\delta_t)} && (13)\\
&\Leftrightarrow \frac{P(Y^1, \cdots, Y^t \mid H_1)P(H_1)}{P(Y^1, \cdots, Y^t \mid H_0)P(H_0)} \geq \frac{\rho_t(\delta_t)}{1 - \rho_t(\delta_t)} && (14)\\
&\Leftrightarrow Z_t \geq \frac{P_0 \rho_t(\delta_t)}{P_1(1 - \rho_t(\delta_t))} && (15)
\end{aligned}$$

(recall that $P_1 = P(H_1)$, and $P_0 = P(H_0)$). From (13) to (14), we used Bayes' theorem.

The equality (4) can be written as follows:

$$\delta_t = \rho_t^{-1}\left(\frac{P_1}{P_1 + P_0\alpha_{max}}\right) \quad \Leftrightarrow \quad \frac{P_1(1 - \rho_t(\delta_t))}{P_0\rho_t(\delta_t)} = \alpha_{max}. \tag{16}$$

By (15)(16), we have

$$Z_t \geq \frac{1}{\alpha_{max}}. \tag{17}$$

This means that the proposed algorithm is equivalent to SPRT in the case where we set thresholds as $A = 1/\alpha_{max}$ and $B = 0$. Since α and β in

(7) are FAR and FRR in this case, the proposed algorithm minimizes the average number of inputs while keeping FAR less than or equal to α_{max} in the asymptotic case where the error probabilities (FAR and FRR) are sufficiently small. □

It should be noted that the proposed sequential fusion scheme is optimal only when FAR and FRR are sufficiently small. This can be achieved by setting the required FAR α_{max} sufficiently small and the maximum number of inputs T sufficiently large (e.g. $T = 10$ fingers). Otherwise, the optimality of the proposed algorithm is not theoretically guaranteed. However, since the same holds true for the conventional score level fusion schemes based on SPRT or MSPRT [12, 13], the proposed scheme has the same theoretical property as the conventional schemes. We emphasize again that the proposed scheme has the advantage that it can be applied to biometric cryptosystems.

3.5. Setting parameters of the error-correcting code

We now describe how to set parameters k (the number of information symbols) and d_{min} $(= 2\delta_t + 1)$ (the minimum distance) in the (tn, k, d_{min}) linear ECC to satisfy the assumption (i) when we use the fuzzy commitment scheme.

If the codeword length (i.e. the dimension of large biometric features) tn can be expressed as $tn = 2^m - 1$ (m is a positive integer), we can use a cyclic code such as the BCH code [32]. Specifically, we can satisfy the assumption (i) by computing the number of information symbols k corresponding to the codeword length $tn = 2^m - 1$ and the minimum distance $d_{min} = 2\delta_t + 1$, and using the (tn, k, d_{min}) BCH code. For example, in our experiments in Section 4, we used finger-vein features represented as 511-bit binary strings (i.e. $n = 511 = 2^9 - 1$), and used the $(511, 448, 15)$ BCH code at the 1st input $(t = 1)$ in the case when the distance threshold is $\delta_1 = 7$.

If tn cannot be expressed as $tn = 2^m - 1$, we can use a *shortened cyclic code* [32], which forms a (tn, k, d_{min}) linear code from a $(tn + i, k + i, d_{min})$ cyclic code (i is a positive integer) by omitting the first i rows and columns of the generator matrix. For example, in our experiments in Section 4, we used the $(1022, 142, 253)$ shortened cyclic code, which is obtained from the $(1023, 143, 253)$ BCH code $(i = 1)$, at the 2nd input $(n = 511, t = 2)$ when the distance threshold is $\delta_2 = 126$.

However, it must be noted that the number of information symbols k (i.e. codeword space) needs to be sufficiently large to defend against an

attack that exhaustively searches codewords offline to recover an original biometric feature X_t from auxiliary data AD_t (referred to as the *codeword search attack*). For example, if the fuzzy commitment scheme is used as a biometric cryptosystem (i.e. $\mathrm{AD}_t = X_t \oplus C(S_t)$), the attacker can recover X_t from AD_t by computing, for every $C' \in \mathcal{C}$ ($\mathcal{C}$ is a set of codewords), $\mathrm{AD}_t \oplus C'$ (if $C' = C(S_t)$, then $X_t = \mathrm{AD}_t \oplus C'$). In other words, we need to defend against both the FAR attack and the codeword search attack to achieve high-level security (e.g. we need to satisfy FAR $\leq 2^{-64}$ and $k \geq 64$ to achieve the 64-bit security).

If the number of omitted rows/columns i in the shortened cyclic code (which forms the (tn, k, d_{min}) linear code from the $(tn+i, k+i, d_{min})$ linear code) is large, the number of information symbols k can be small. Similarly, if the minimum distance d_{min} in the (tn, k, d_{min}) BCH code is large (i.e. the distance threshold δ_t is large), the number of information symbols k can be small. In such cases, we can use the LDPC (Low-Density Parity-Check) code [33], which provides a performance very close to the Shannon limit. Since the LDPC code is more efficient than the BCH code (especially in the case when the codeword length tn is very large), we can increase the number of information symbols k by using the LDPC code. Also, we can construct the LDPC code for any codeword length tn (unlike a cyclic code such as the BCH code). Specifically, we can satisfy the assumption (i) by determining the parameters k and d_{min} of the LDPC code with the codeword length tn so that the distance threshold δ_t ($d_{min} = 2\delta_t + 1$) is as close as the value determined by the proposed scheme (using (4)(5)(6)). Here, we can estimate the minimum distance d_{min} for a given tn and k by using, for example, the nearest nonzero codeword search (NNCS) approach [34].

In our experiments in Section 4, we used the LDPC code at the 3rd input ($n = 511, t = 3$) and the 4th input ($n = 511, t = 4$), since the number of information symbols k in the shortened cyclic code was small at the 3rd and 4th inputs. We obtained an enough large k by using the LDPC code, while satisfying the assumption (i) (see Sections 4.2 and 4.5 for details). Note that the LDPC code uses an iterative decoding algorithm such as the sum-product algorithm [33] to correct errors (noise) in a code. It can happen that the LDPC code (whose minimum distance is $d_{min} = 2\delta_t + 1$) does not correct δ_t errors within a small number of iterations, depending on the noise strength or the noise pattern. In our experiments, we assumed, for simplicity, that a user is accepted with overwhelming probability if a distance $d_t = \mathrm{dis}_t(X_t, Y_t)$ (i.e. the number of errors in a code) is less than or equal to the threshold δ_t,

and rejected with overwhelming probability if d_t is more than δ_t when the LDPC code is used. The experimental evaluation implementing the iterative decoding algorithm is left as future work (i.e. although the LDPC code may not correct δ_t errors within a small number of iterations, we leave the investigation of its effects on the performance as future work).

4. Experimental evaluation

We carried out experiments to evaluate the security and convenience of the proposed sequential fusion scheme when we store AD and PI into a single place, or disclose them to the public (i.e. public template model described in Section 2.1). We first describe the experimental set-up: the finger-vein dataset used in our experiments (Section 4.1), protected template generation (Section 4.2), training score distributions (which is necessary in setting the distance threshold δ_t) (Section 4.3), and an evaluation procedure (Section 4.4). We then report the experimental results (Section 4.5). We finally discuss storage requirements and authentication time (Section 4.6).

4.1. Dataset

We used finger-vein, which is one of the most accurate modalities, as biometric information. There are publicly available finger-vein databases such as the SDUMLA-HMT database [35] and the Hong Kong Polytechnic University Finger Image Database Version 1.0 [36]. However, we used the finger-vein database that is used in [17], because it contains more subjects and more finger-vein images than [35, 36] (note that a large number of subjects and finger-vein images are required in our experiments, since the required FAR is very small).

The finger-vein database in [17] contains 505 subjects (while the datasets in [35] and [36] contain 106 and 156 subjects, respectively). Each subject provided images of 6 fingers (index finger, middle finger, and ring finger of both hands), and the collection for each finger was repeated for 11 times to obtain 11 images (one image was for the enrollment phase, and 10 images were for the authentication phase). We eliminated 32 images at the authentication phase that are not appropriately captured (the corresponding fingers were not appropriately put on the sensor). Thus, we used $505 \times 6 = 3030$ images at the enrollment phase, and $505 \times 6 \times 10 - 32 = 30268$ images at the authentication phase (i.e. 33298 images in total).

Figure 4: Example of the binary finger-vein image ($32 \times 16 = 512$ pixels). While pixels and black pixels represent 1 (vein) and 0 (background), respectively.

We randomly selected 200 subjects from 505 subjects for evaluation, and used the remaining 305 subjects for training the genuine score distribution $g_{t,1}(d_t)$ and the impostor score distribution $g_{t,0}(d_t)$ (recall that $g_{t,1}(d_t)$ and $g_{t,0}(d_t)$ are necessary in setting the distance threshold δ_t; see (4)(5)(6)). We set the number of enrolled fingers per user (i.e. the maximum number of biometric inputs) T as $T = 4$. Here we tried all the 360 ($= {}_6P_4$) ways to choose the 1st, 2nd, 3rd, and 4th fingers (each of the 200 subjects for evaluation input his/her fingers in this order at the authentication phase), and carried out, for each case, the experiments described in Sections 4.2, 4.3, and 4.4.

4.2. Protected template generation

We extracted, for each image, a finger-vein feature represented as a 511-bit binary string (i.e. $X^t, Y^t \in \{0,1\}^{511}$ $(1 \leq t \leq 4)$). Specifically, we first extracted a finger-vein pattern from each finger-vein image using the feature extraction method in [37]. Then we resized each finger-vein pattern image to 32×16 ($= 512$) pixels and transformed it to a binary (2-level) image (each pixel takes the value of either 1 (vein) or 0 (background)). Figure 4 shows an example of the binary image. For each binary image, we eliminated a pixel of the bottom right corner, and used the remaining 511 ($= 2^9 - 1$) pixels as a 511-bit binary string to use the $(511, k, d_{min})$ BCH code at the 1st input (see Section 3.5). We used the Hamming distance as a distance metric, and evaluated the accuracy of a single biometric feature (i.e. 511-bit binary string) using the 200 subjects for evaluation. EER (Equal Error Rate; the operating point where FAR = FRR) was 8.0×10^{-4}.

We used the fuzzy commitment scheme [3] as a biometric cryptosystem. We used a bit concatenation as a mapping function f_t, and constructed a template generation algorithm $\mathbf{Gen}_t$ and a reproduce algorithm $\mathbf{Rep}_t$ using an error-correcting code. Specifically, we used the $(511, k, d_{min})$ BCH code at the 1st input, and the $(1022, k, d_{min})$ shortened cyclic code, which is obtained from the $(1023, k+1, d_{min})$ BCH code ($i = 1$), at the 2nd input. We used the

$(1533, k, d_{min})$ LDPC code at the 3rd input, and the $(2044, k, d_{min})$ LDPC code at the 4th input. We determined the parameters k and d_{min} of the LDPC code in the way described in Section 3.5. We verified that we can set the parameters k and d_{min} $(= 2\delta_t + 1)$ for the distance threshold δ_t that is very close to the value determined by the proposed scheme (using (4)(5)(6)). In other words, we verified that the assumption (i) is satisfied in our experiments.

4.3. Training score distributions

We trained the genuine score distribution $g_{t,1}(d_t)$ and the impostor score distribution $g_{t,0}(d_t)$ using genuine scores and impostor scores obtained from the 305 subjects. Specifically, we assumed that a distance $d = \mathrm{dis}(X^i, Y^i)$ between two finger-vein features $X^i \in \{0,1\}^{511}$ and $Y^i \in \{0,1\}^{511}$ is generated from a beta-binomial distribution if H_1 is true (i.e. X^i and Y^i are input by the same user), and generated from a normal distribution $N(\mu, \sigma^2)$ otherwise (irrespective of the type of the fingers), in the same way as [17]. The beta-binomial distribution is a binomial distribution $B(n, p)$ ($n = 511$ in our experiments) where p is not fixed but randomly generated from a beta distribution $\mathrm{Beta}(a, b)$. Thus, we denote it by BB(n, a, b) ($n = 511$) in this paper. We trained the parameters a, b in BB(n, a, b) ($n = 511$) using the method of moments [38], and the parameters μ, σ^2 in $N(\mu, \sigma^2)$ using the ML (Maximum Likelihood) estimation method. Figure 5 shows how well BB(n, a, b) ($n = 511$) and $N(\mu, \sigma^2)$ trained from the 305 subjects fit the genuine scores and impostor scores from the 200 subjects for evaluation, respectively.

After training BB(n, a, b) ($n = 511$) and $N(\mu, \sigma^2)$, we can compute the genuine score distribution $g_{t,1}(d_t)$ and the impostor score distribution $g_{t,0}(d_t)$ between two large finger-vein features (i.e. concatenated binary strings) $X_t, Y_t \in \{0,1\}^{tn}$ $(n = 511)$ as follows:

$$g_{t,1}(d_t) = \underbrace{(\mathrm{BB}(n,a,b) * \cdots * \mathrm{BB}(n,a,b))}_{t\text{-times convolution}}(d_t) \quad (n = 511) \tag{18}$$

$$g_{t,0}(d_t) = \underbrace{(N(\mu,\sigma^2) * \cdots * N(\mu,\sigma^2))}_{t\text{-times convolution}}(d_t). \tag{19}$$

We set the required FAR α_{max} as $\alpha_{max} = 2^{-128}, 2^{-80}, 2^{-64}$, or $2^{-13.7}$ (i.e. 128-bit, 80-bit, 64-bit, or 13.7-bit security; we explain the reason we evaluated the case of 13.7-bit security later in details). Then we set the prior

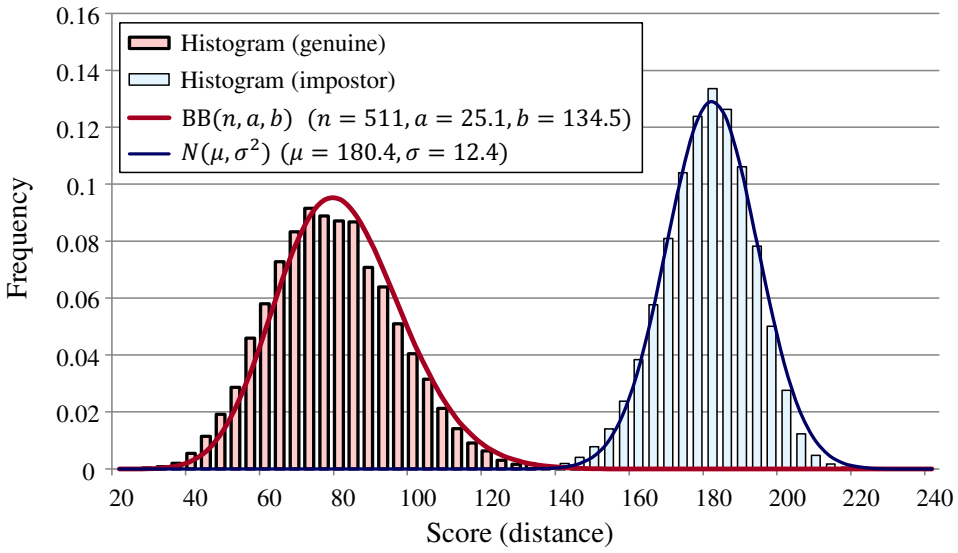

Figure 5: The beta-binomial distribution BB(n, a, b) ($n = 511, a = 25.1, b = 134.5$) and the normal distribution $N(\mu, \sigma^2)$ ($\mu = 180.4, \sigma = 12.4$) trained from the 305 subjects. They fit the histograms of genuine scores and impostor scores from the 200 subjects for evaluation, respectively.

probabilities P_1 and P_0 in (4) as $P_1 = P_0 = 0.5$, and a distance threshold δ_t using (4)(5)(6).

It should be noted that since $g_{t,1}(d_t)$ and $g_{t,0}(d_t)$ in (5) can be computed from BB(n, a, b) ($n = 511$) and $N(\mu, \sigma^2)$, respectively, the assumption (ii) is satisfied. We also verified that the assumption (iii) was satisfied in most cases (when the FAR value computed from $g_{t,0}(d_t)$ is more than 2^{-148}). Indeed, we can see from Figure 6 that the function ρ_1 ($p_1 = \rho_1(d_1)$) at the 1st input is monotonically decreasing (similar curves were obtained for the 2nd, 3rd, and 4th inputs). Since we also verified that the assumption (i) is satisfied (see Section 4.2), we can say that all of the assumptions (i)(ii)(iii) are satisfied in our experiments.

4.4. Evaluation procedure

We evaluated the performance of the proposed scheme and the OR rule [11] using the 200 subjects for evaluation. Specifically, we evaluated the performance in the case when each of the 200 subjects attempts verification against each of them by sequentially inputting 4 fingers. Here, it must be

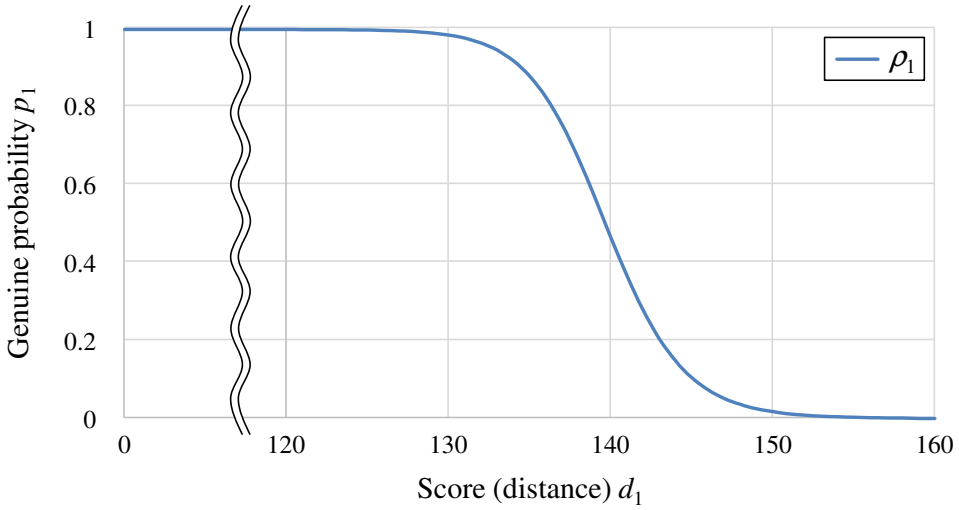

Figure 6: Function ρ_1 at the 1st input ($p_1 = \rho_1(d_1)$, where p_1 is a genuine probability and d_1 is a score (distance)). ρ_1 was computed from $g_{t,1}(d_t) = \mathrm{BB}(n, a, b)$ ($n = 511, a = 25.1, b = 134.5$) and $g_{t,0}(d_t) = N(\mu, \sigma^2)$ ($\mu = 180.4, \sigma = 12.4$) using (5).

noted that we cannot evaluate too small FAR such as FAR $= 2^{-128}, 2^{-80}$, and 2^{-64} with high statistical confidence based only on "real" data, even if we use the database in [17] that contains a large number of finger-vein images[2]. According to the rule of three [39], more than 3α independent impostor attempts are required to conclude that FAR $= \alpha$ with 95% confidence. Assuming that all of the 200 subjects are independent, the least value of FAR that can be estimated with 95% confidence based only on the "real" data is $3/(200 \times 199) = 2^{-13.7}$ (i.e. 13.7-bit security).

Thus, in the case of 128-bit, 80-bit, or 64-bit security, we evaluated FAR through *simulation experiments*. Specifically, we estimated a normal distribution $N(\mu', \sigma'^2)$ using impostor scores from the 200 subjects for evaluation, and randomly generated impostor scores from $N(\mu', \sigma'^2)$ to obtain enough impostor scores. Figure 7 shows the normal distribution $N(\mu', \sigma'^2)$, whose parameters μ', σ'^2 were estimated using the ML estimation method ($\mu' = 180.2, \sigma' = 12.3$). It can be seen that $N(\mu', \sigma'^2)$ fits a histogram of im-

[2]We also confirmed that there were "no" false accepts when we set $\alpha_{max} = 2^{-128}, 2^{-80}$, or 2^{-64} (i.e. 128-bit, 80-bit, or 64-bit security) and used "real" impostor scores from the 200 subjects.

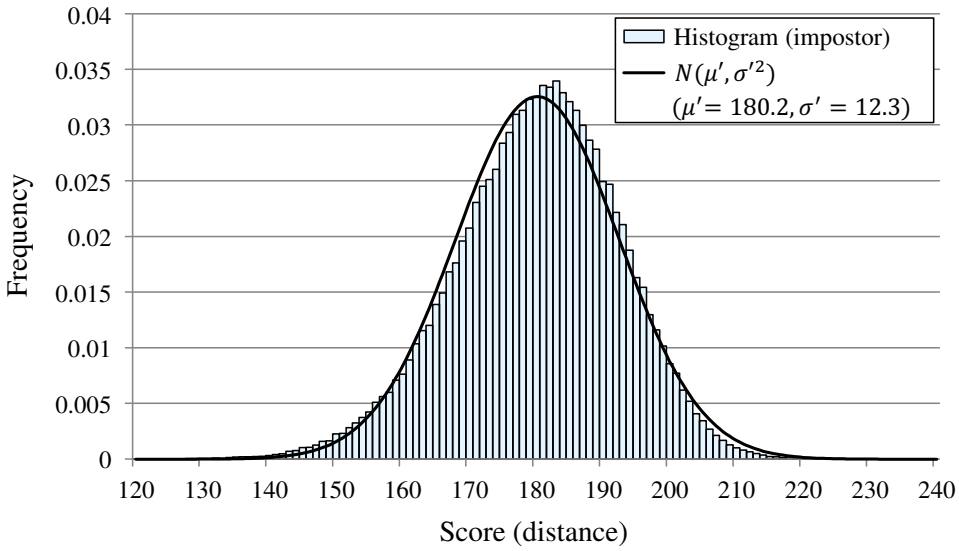

Figure 7: Histogram of impostor scores from the 200 subjects for evaluation and the fitted normal distribution $N(\mu', \sigma'^2)$ ($\mu' = 180.2, \sigma' = 12.3$). In the case of 128-bit, 80-bit, or 64-bit security, we randomly generated impostor scores from $N(\mu', \sigma'^2)$ to obtain enough impostor scores. In the case of 13.7-bit security, we used the impostor scores from the 200 subjects to evaluate the performance based completely on the "real" data.

postor scores (from the 200 subjects for evaluation) well[3]. We evaluated FAR by randomly generating impostor scores from this normal distribution. We emphasize that we evaluated FRR based only on "real" genuine scores even in this case (i.e. 128-bit, 80-bit, or 64-bit security). We also evaluated the performance in the case of 13.7-bit security (not by simulation but) by using the "real" impostor scores from the 200 subjects to compare the proposed scheme with the OR rule based completely on the "real" data.

As a performance measure, we used the rate that a genuine user is not still

[3]Note that impostor score values that cause false accepts are much lower than 120 (leftmost value in Figure 7) in the proposed scheme, since the distance threshold δ_t is very small. For example, the threshold δ_t at the first input is $\delta_t = 63$, 47, and 7 in the case of the 64-bit, 80-bit, and 128-bit security, respectively (see Figure 10 for details). However, since there is "no" impostor score whose value is less than such a threshold, we cannot fit an impostor distribution to an area that causes false accepts. Thus, we fit an impostor distribution to the histogram using the ML estimation method, which we believe the best that could be done.

Fusion scheme	Required FAR	Unaccepted rate				Average number of inputs	FAR
		$t = 1$	$t = 2$	$t = 3$	$t = 4$		
Proposed	2^(-128)	1.0	0.90	0.32	5.5×10^{-2}	3.22	2^(-133)
	2^(-80)	0.98	0.27	3.3×10^{-2}	5.3×10^{-3}	2.28	2^(-87)
	2^(-64)	0.83	0.13	1.1×10^{-2}	2.3×10^{-3}	1.97	2^(-70)
	2^(-13.7)	1.5×10^{-2}	1.5×10^{-4}	0	0	1.01	2^(-16.3)
OR	2^(-128)	1.0	1.0	1.0	1.0	4.00	2^(-133)
	2^(-80)	0.98	0.95	0.93	0.91	3.86	2^(-86)
	2^(-64)	0.84	0.72	0.62	0.54	3.18	2^(-69)
	2^(-13.7)	1.5×10^{-2}	5.2×10^{-4}	2.5×10^{-5}	0	1.02	2^(-16.3)

Figure 8: The unaccepted rate, the average number of inputs, and FAR in the proposed scheme and the OR rule.

accepted after the t-th input ($1 \leq t \leq 4$), which we call the *unaccepted rate* (it is equivalent to FRR when $t = 4$). We also evaluated the *average number of inputs* over all genuine attempts, which measures how many biometric samples on average a genuine user needs to input until acceptance or rejection at the authentication phase[4]. We further evaluated FAR to see whether FAR is less than the required value α_{max}. We averaged these performances over all the 360 ($= {}_6P_4$) ways to choose the 1st, 2nd, 3rd, and 4th fingers to obtain a stable performance.

4.5. Experimental results

Figure 8 shows the unaccepted rate, the average number of inputs, and FAR. We also show in Figure 9 the graphs representing the unaccepted rate after the 1st, 2nd, 3rd, and 4th input, respectively. In the case of the 13.7-bit security, there were "no" false rejects after the 3rd and 4th input in the proposed scheme and the OR rule, respectively.

It can be seen from Figures 8 and 9 that the proposed scheme significantly

[4]The average number of inputs can be computed from the unaccepted rate. For example, if the maximum number of inputs is $T = 4$ and the unaccepted rate at the 1st, 2nd, and 3rd input is 0.4, 0.3, and 0.1, respectively, the average number of inputs is $1 \times (1 - 0.4) + 2 \times (0.4 - 0.3) + 3 \times (0.3 - 0.1) + 4 \times 0.1 = 1.8$.

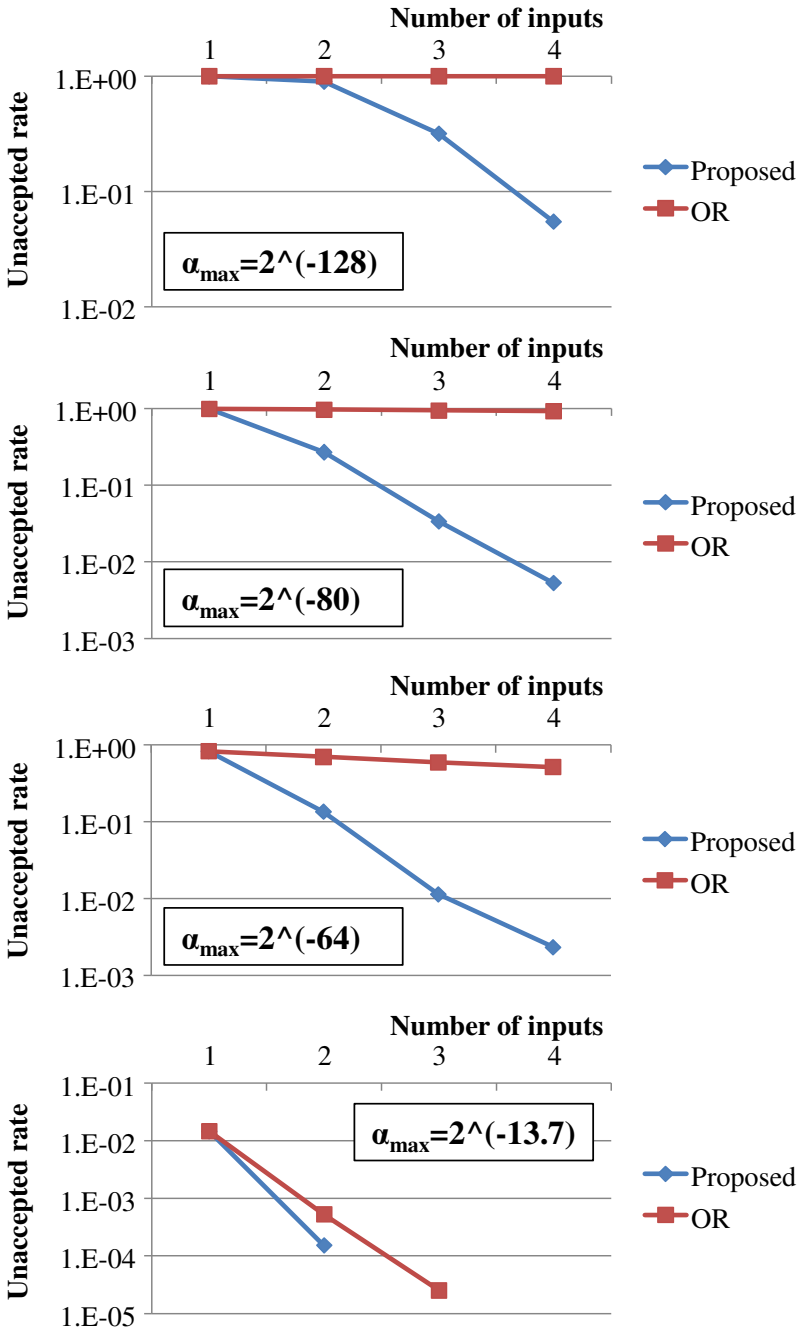

Figure 9: Unaccepted rate after the 1st, 2nd, 3rd, and 4th input in the proposed scheme and the OR rule.

Required FAR	Parameters (tn, k, d_{min}) in the error-correcting code			
	$t = 1$	$t = 2$	$t = 3$	$t = 4$
2^(-128)	(511, 448, 15) ($\delta_t = 7$)	(1022, 142, 253) ($\delta_t = 126$)	(1533, 245, 510) ($\delta_t = 254$)	(2044, 277, 788) ($\delta_t = 393$)
2^(-80)	(511, 166, 95) ($\delta_t = 47$)	(1022, 100, 351) ($\delta_t = 175$)	(1533, 83, 618) ($\delta_t = 308$)	(2044, 100, 876) ($\delta_t = 437$)
2^(-64)	(511, 85, 127) ($\delta_t = 63$)	(1022, 65, 379) ($\delta_t = 189$)	(1533, 66, 652) ($\delta_t = 325$)	(2044, 64, 898) ($\delta_t = 448$)

Figure 10: Parameters (tn, k, d_{min}) of the error-correcting code in the proposed scheme. We also show the corresponding distance threshold δ_t.

outperforms the OR rule. For example, FRR (the unaccepted rate after the 4-th input) of the OR rule is very high in the case of the 64-bit, 80-bit, and 128-bit security: 54% and 91% and 100%, respectively. On the other hand, in the propose scheme, the unaccepted rate is reduced to 0.23%, 0.53%, and 5.5% after the 4th input in the case of the 64-bit, 80-bit, and 128-bit security, respectively. The average number of inputs is also much lower: in the case of the 80-bit security for example, it is 3.86 in the OR rule while 2.28 in the proposed scheme. It can also be seen that FAR in the proposed scheme is always less than the required FAR[5].

We would like to emphasize that the proposed scheme also outperforms the OR rule in the case of the 13.7-bit security. For example, there are some false rejects at the 3rd input in the OR rule, while there are "no" false rejects at the 3rd input in the proposed scheme. This means that the effectiveness of the proposed scheme is shown based completely on the "real" data.

We also examined whether the number of information symbols k in the

[5]Note that FAR of the proposed scheme was $2^{-16.3}$ (less than $2^{-13.7}$) in the case of 13.7-bit security, despite the fact that the least value of FAR that can be estimated with 95% confidence is $3/(200 \times 199) = 2^{-13.7}$ (according to the rule of three). It can happen that FAR is less than $2^{-13.7}$ because we tried all the 360 ($= {}_6P_4$) ways to choose the 1st, 2nd, 3rd, and 4th fingers. For example, when there is only one false accept, FAR is computed to be $1/(200 \times 199 \times 360) = 2^{-23.8}$. However, since the 360 ways to choose four fingers are *not* independent, the least value of FAR that can be estimated with 95% confidence is $3/(200 \times 199) = 2^{-13.7}$ as explained above. Thus, when FAR was computed to be $2^{-16.3}$, we can only conclude with high confidence that *FAR was less than* $2^{-13.7}$. However, this conclusion is sufficient in our experiments, since $2^{-13.7}$ is the required FAR.

proposed scheme is more than or equal to 64, 80, and 128 in the case of the 64-bit, 80-bit, and 128-bit security, respectively (we did not examine k in the case of the 13.7-bit security since the security level is not high; the system is not secure against the FAR attack). Figure 10 shows the parameters (tn, k, d_{min}) of the error-correcting code in our experiments. It can be seen that the number of information symbols k is really more than or equal to 64, 80, and 128 in the case of the 64-bit, 80-bit, and 128-bit security, respectively, which means that the proposed scheme is secure against not only the FAR attack but also the codeword search attack described in Section 3.5.

We finally evaluated EER by changing the required FAR until we find the operating point where FAR = FRR (the unaccepted rate after the 4-th input). Here, since not only FAR but also FRR is too small to be evaluated based only on real data, we estimated a beta-binomial distribution BB(n, a', b') ($n = 511$) using genuine scores from the 200 subjects for evaluation ($a' = 23.5, b' = 126.6$), and randomly generated genuine scores from BB(n, a', b') to obtain enough genuine scores. In other words, we evaluated EER using the fitted beta-binomial distribution BB(n, a', b') ($n = 511, a' = 23.5, b' = 126.6$) and the fitted normal distribution $N(\mu', \sigma'^2)$ ($\mu' = 180.2, \sigma' = 12.3$) (see Figure 7). The results were as follows: EER $= 4.0 \times 10^{-7}$ in the OR rule, while EER $= 7.3 \times 10^{-14}$ in the proposed scheme.

These results demonstrate that the secure and convenient public template model in the biometric cryptosystem is made possible by using the proposed scheme.

4.6. Discussions on storage requirements and authentication time

It should be noted that the proposed scheme requires larger storage capacity than the conventional feature level fusion scheme for the biometric cryptosystem [6, 7, 8, 9] (i.e. parallel feature level fusion), since it requires several pairs of ADs and PIs. Time required for authentication can also increase, since multiple decoding attempts are required when a user inputs more than one biometric sample. Thus, we discuss storage requirements and authentication time in our experiments.

In our experiments, we used a finger-vein feature represented as a 511-binary string, and set the number of enrolled fingers per user (i.e. the maximum number of biometric inputs) T as $T = 4$. Thus, $511 \times 4 = 2044$ bits (about 256 bytes) per user are required for storage when we use the conventional parallel feature level fusion scheme [6, 7, 8, 9]. If the input order is not determined, the proposed scheme requires much larger storage capacity,

Required FAR	Encoding/decoding time			
	$t = 1$	$t = 2$	$t = 3$	$t = 4$
2^(-128)	73 µs (encode) 87 µs (decode)	306 µs (encode) 1.89 ms (decode)	1.32 ms (encode) 76.5 ms (decode)	5.80 ms (encode) 351 ms (decode)
2^(-80)	144 µs (encode) 500 µs (decode)	263 µs (encode) 280 µs (decode)	2.63 ms (encode) 165 ms (decode)	5.60 ms (encode) 353 ms (decode)
2^(-64)	89 µs (encode) 608 µs (decode)	168 µs (encode) 3.31 ms (decode)	2.58 ms (encode) 155 ms (decode)	6.20 ms (encode) 683 ms (decode)

Figure 11: Encoding/decoding time.

since there are $\sum_{t=1}^{T} {}_{T}C_t = 2^T - 1 = 15$ possible combinations (as described in Section 3.1). Specifically, since there are ${}_4C_1 = 4$, ${}_4C_2 = 6$, ${}_4C_3 = 4$, and ${}_4C_4 = 1$ possible combinations for the 1st, 2nd, 3rd, and 4th input, respectively, the proposed scheme requires $(4\times1+6\times2+4\times3+1\times4)\times511 = 16352$ bits (2044 bytes) per user in this case (i.e. 8 times as much as the conventional scheme). However, if the input order is determined in advance, the storage capacity required in the proposed scheme becomes much smaller: $(1 + 2 + 3 + 4) \times 511 = 5110$ bits (about 639 bytes) per user (i.e. 2.5 times as much as the conventional scheme).

We then show that the increase of the authentication time does not matter much in practice by measuring the decoding time in our experiments. Specifically, we measured the decoding time of the BCH code, the shortened cyclic code, and the LDPC code, whose parameters (tn, k, d_{min}) were set in the same way as Figure 10, on the Intel Core i5-4690 (3.5GHz) with 24GB 1600MHz DDR3 RAM (recall that we used the BCH code at the 1st input, the shortened cyclic code at the 2nd input, and the LDPC code at the 3rd and 4th inputs). We set the number of iterations in the LDPC code as 100, assuming that errors in a code can be corrected within the 100 iterations.

Figure 11 shows the results (we also show the encoding time, which is required in the enrollment phase). Note that the conventional parallel feature level fusion scheme [6, 7, 8, 9] also requires the decoding attempt after a user inputs all of the 4 fingers. Thus, the increase of the decoding time caused by using the proposed scheme (when a user is required to input all of the 4 fingers) is 78.5 ms (= 87 μs + 1.89 ms + 76.5 ms), 166 ms (= 500 μs + 280 μs + 165 ms), 159 ms (= 608 μs + 3.31 ms + 155 ms), in the case of the 128-bit, 80-bit, and 64-bit security, respectively. Since they can be regarded

as small (for example, they are much smaller than one second), we can say that the increase of the decoding time does not cause much inconvenience in our experiments (similarly, we can say that the increase of the encoding time at the enrollment phase is small).

5. Security of multiple protected templates

In Section 4, we showed that the proposed scheme can provide a convenient way of authentication while achieving security against both the FAR attack and the codeword search attack (i.e. it can simultaneously achieve a sufficiently small FAR and a sufficiently large number of information symbols k). However, since multiple protected templates $(\mathrm{AD}_t, \mathrm{PI}_t)$ $(1 \leq t \leq T)$ are enrolled per user in the proposed scheme, the attacker may try to exploit the relationship between the multiple protected templates.

Thus, we analyze the security of the proposed scheme against such attacks in this section. In our analysis, we assume that the fuzzy commitment scheme is used (in the same way as Section 4), and the input order is determined in advance for simplicity (we can analyze the security in the case when the input order is not determined in the same way). We firstly consider what we call the *partial codeword search attack* as an attack that exploits the relationship between the multiple protected templates (Section 5.1). We secondly analyze the security of the proposed scheme against this attack (Section 5.2). We thirdly consider another attack, which we call the *partial feature search attack*, and discuss the security against this attack (Section 5.3). We finally discuss the security against other possible attacks (Section 5.4).

5.1. Partial codeword search attack

In the proposed scheme applied to the fuzzy commitment scheme, multiple protected templates $(\mathrm{AD}_t, \mathrm{PI}_t)$ $(1 \leq t \leq T)$ are enrolled per user (recall that $\mathrm{AD}_t = X_t \oplus C(S_t)$ and $\mathrm{PI}_t = H(S_t)$, where $X_t \in \{0,1\}^{tn}$ is a large biometric feature, $S_t \in \{0,1\}^k$ $(k < n)$ is a secret key, C is an (tn, k, d_{min}) linear ECC, and H is a hash function). Then, the attacker may try to exploit the relationship between the multiple protected templates $(\mathrm{AD}_t, \mathrm{PI}_t)$ $(1 \leq t \leq T)$. In particular, the attacker may try to exploit the fact that *the first n-bit part of $X_1, \cdots, X_T$ are exactly identical to each other* (since they are the first biometric feature at the enrollment phase $X_1 \in \{0,1\}^n$). As one such example, we consider what we call the *partial codeword search attack*, which is described in the following.

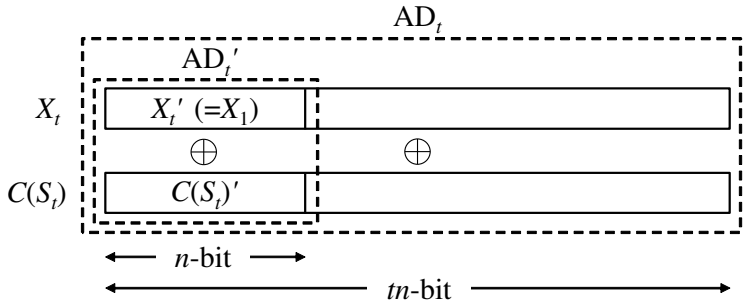

Figure 12: AD_t', X_t', and $C(S_t)'$ (the first n-bit part of AD_t, X_t,, and $C(S_t)$).

Let $\mathrm{AD}_t', X_t', C(S_t)' \in \{0,1\}^n$ be the first n-bit part of AD_t, X_t, $C(S_t) \in \{0,1\}^{tn}$, respectively. Note that $X_t' = X_1$. Figure 12 shows AD_t', X_t', and $C(S_t)'$. We refer to $C(S_t)'$ (i.e. the first n-bit part of a valid codeword $C(S_t)$) as a *valid partial codeword*. Let $\mathcal{C}'$ be a set of valid partial codewords (i.e. $C(S_t)' \in \mathcal{C}'$), and k' be the entropy of valid partial codewords (i.e. there are $2^{k'}$ valid partial codewords). If $k' \ll k$ (i.e. the entropy of valid partial codewords is much smaller than that of valid codewords), the attacker can exploit this as follows:

1. Randomly choose a valid partial codeword $\widetilde{C(S_t)'} \in \{0,1\}^n$ from $\mathcal{C}'$ ($\mathcal{C}'$ is a set of valid partial codewords).
2. Compute a guess value $\widetilde{X_1}$ of X_1 as follows: $\widetilde{X_1} = \widetilde{C(S_t)'} \oplus \mathrm{AD}_t'$ (recall that $X_1 = X_t' = C(S_t)' \oplus \mathrm{AD}_t'$).
3. Compute a guess value $\widetilde{C(S_1)}$ of $C(S_1)$ as follows: $\widetilde{C(S_1)} = \widetilde{X_1} \oplus \mathrm{AD}_1$.
4. Decode $\widetilde{C(S_1)}$. Let $\widetilde{S_1}$ be a decoded value.
5. Verify whether $\widetilde{S_1} = S_1$ or not using $\mathrm{PI}_1 = H(S_1)$.

Note that if $\widetilde{C(S_t)'} = C(S_t)'$, then $\widetilde{X_1} = X_1$ and $\widetilde{S_1} = S_1$. Thus, the attacker can recover the first biometric feature X_1 by exhaustively searching valid partial codewords (there are $2^{k'}$ valid partial codewords in total) via the above algorithm. We refer to this attack as the *partial codeword search attack*, which is much stronger than the codeword search attack (described in Section 3.5) if and only if $k' \ll k$ (i.e. the entropy of valid partial codewords is much smaller than that of valid codewords).

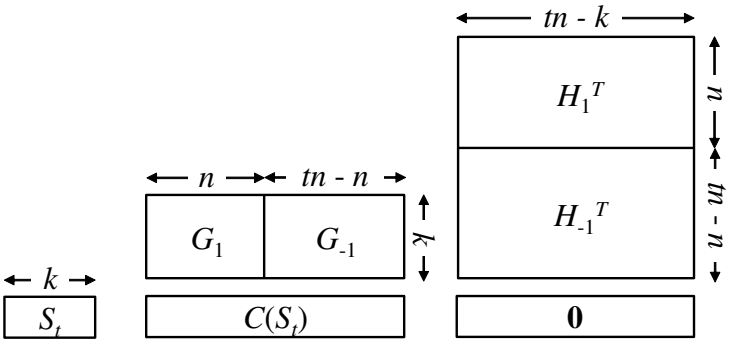

Figure 13: The generator matrix $G = [G_1|G_{-1}]$ and the parity check matrix $H = [H_1|H_{-1}]$ ($C(S_t) = S_tG$ and $C(S_t)H^T = \mathbf{0}$).

5.2. Security against the partial codeword search attack

The partial codeword search attack is much stronger than the codeword search attack if and only if the entropy of valid partial codewords k' is much smaller than that of valid codewords k. However, we show that $k' = k$ (i.e. *the partial codeword search attack is only as strong as the codeword search attack*) in most cases.

Let G ($k \times tn$ matrix that has full rank: $\mathrm{rank}(G) = k$ ($< tn$)) be a generator matrix of the (tn, k, d_{min}) linear error-correcting code C (i.e. $C(S_t) = S_tG$). Let further G_1 ($k \times n$ matrix) and G_{-1} ($k \times (tn - n)$ matrix) be the first n columns of G and the remaining part of G, respectively (i.e. $G = [G_1|G_{-1}]$). Then, a valid partial codeword $C(S_t)' \in \{0,1\}^n$ can be generated by multiplying a k-bit random string (i.e. secret key) $S_t \in \{0,1\}^k$ by G_1: $C(S_t)' = S_tG_1$. When we attempt 2^k possible random strings as S_t, we obtain $2^{\mathrm{rank}(G_1)}$ valid partial codewords. Thus, the entropy of valid partial codewords k' can be expressed as $k' = \mathrm{rank}(G_1)$, and we have $k' = k$ if and only if G_1 has full rank (i.e. $\mathrm{rank}(G_1) = k$ ($< n$)).

Taking this into account, we focus on the LDPC code for simple analysis, and show both theoretically and experimentally that the matrix G_1 has full rank in most cases.

5.2.1. Theoretical analysis

We begin by a theoretical analysis. Let H ($(tn - k) \times tn$ matrix) be a parity check matrix of the (tn, k, d_{min}) linear error-correcting code C (i.e. for any valid codeword $C(S_t) \in \{0,1\}^{tn}$, $C(S_t)H^T = \mathbf{0}$ ($\mathbf{0}$ is a zero vector)).

Let further H_1 ($(tn-k) \times n$ matrix) and H_{-1} ($(tn-k) \times (tn-n)$ matrix) be the first n columns of H and the remaining part of H, respectively (i.e. $H = [H_1|H_{-1}]$). Figure 13 shows the generator matrix $G = [G_1|G_{-1}]$ and the parity check matrix $H = [H_1|H_{-1}]$.

In the following, we firstly prove that if H_{-1} has full rank, then G_1 has full rank (Proposition 1). We secondly show that H_{-1} has full rank in most cases.

Proposition 1. *For any $\epsilon > 0$, we have*

$$\mathit{rank}(H_{-1}) > tn - n - \epsilon \Longrightarrow \mathit{rank}(G_1) > k - \epsilon. \tag{20}$$

Thus, if H_{-1} has full rank, then G_1 has full rank (i.e. $\mathit{rank}(H_{-1}) = tn-n \Longrightarrow \mathit{rank}(G_1) = k$, which is obtained by substituting $\epsilon = 1$ in (20)).

Proof. Since G and H are the generator matrix and the parity check matrix, respectively, we have

$$GH^T = \mathbf{0} \Longleftrightarrow G_1 H_1^T + G_{-1} H_{-1}^T = \mathbf{0}. \tag{21}$$

Suppose that $\mathrm{rank}(G_1) \leq k - \epsilon$ ($\epsilon > 0$). Then, there exists a $k \times k$ regular matrix M such that the bottom ϵ rows of MG_1 are $\mathbf{0}$ (such M can be found by Gaussian elimination). By multiplying the both sides of (21) by M, we have

$$MG_1 H_1^T + MG_{-1} H_{-1}^T = \mathbf{0}. \tag{22}$$

Since the bottom ϵ rows of $MG_1 H_1^T$ are $\mathbf{0}$, the bottom ϵ rows of $MG_{-1} H_{-1}^T$ are also $\mathbf{0}$. Let L_{-1} be the bottom ϵ rows of MG_{-1} (L_{-1} is an $\epsilon \times (tn-n)$ matrix). Then,

$$L_{-1} H_{-1}^T = \mathbf{0}. \tag{23}$$

Here, suppose that L_{-1} is rank deficient. Then, since the bottom ϵ rows of MG_1 are $\mathbf{0}$, the bottom ϵ rows of $MG = M[G_1|G_{-1}]$ is also rank deficient, and so is G (since M is a regular matrix). However, this violates the assumption that G has full rank, which is described in the 2nd paragraph of Section 5.2. Thus, L_{-1} has full rank (i.e. $\mathrm{rank}(L_{-1}) = \epsilon$). Then, it follows from (23) that $\mathrm{rank}(H_{-1}) \leq tn - n - \epsilon$. The contrapositive of "$\mathrm{rank}(G_1) \leq k - \epsilon \Longrightarrow \mathrm{rank}(H_{-1}) \leq tn-n-\epsilon$" is "$\mathrm{rank}(H_{-1}) > tn-n-\epsilon \Longrightarrow \mathrm{rank}(G_1) > k-\epsilon$". □

We now show that H_{-1}^T ($(tn-n)\times(tn-k)$ matrix; $tn-n<tn-k$) has full rank (i.e. $\mathrm{rank}(H_{-1}^T)=\mathrm{rank}(H_{-1})=tn-n$) in most cases. We show this in the case of the (irregular) LDPC code, since it is easy to analyze. Let $u=tn-n$ and $v=tn-k$ (i.e. H_{-1}^T is a $u\times v$ matrix; $u<v$). Assume that each element of H_{-1}^T follows the Bernoulli distribution with success probability p (i.e. each element takes the value "1" with probability p and the value "0" with probability $1-p$). Let further $P(u,v)$ be the probability that H_{-1}^T has full rank (i.e. $\mathrm{rank}(H_{-1}^T)=u$), which is evaluated in the following.

The matrix H_{-1}^T is rank deficient if and only if there exist l rows ($1\le l\le u$) in H_{-1}^T such that the summation of the l rows is $\mathbf{0}$. Consider a specific set of l rows. The summation of the first column in the l rows is 0 if and only if there are even numbers of "1"s in the column. Thus, the probability $Q_1(l)$ that the summation of the first column in the l rows is 0 is given by

$$Q_1(l)=(1-p)^l+{}_lC_2p^2(1-p)^{(l-2)}+{}_lC_4p^4(1-p)^{(l-4)}+\cdots. \tag{24}$$

Since there are v columns in H_{-1}^T, the probability $Q_2(l,v)$ that the summation of the l rows is $\mathbf{0}$ is given by

$$Q_2(l,v)=Q_1(l)^v. \tag{25}$$

Let $Q_3(l,u,v)$ ($1\le l\le u$) be the probability that there is *no* set of l rows in H_{-1}^T such that the summation of the l rows is $\mathbf{0}$. There are ${}_uC_l$ ways to choose l rows in H_{-1}^T, and for each of them, the probability that the summation is $\mathbf{0}$ is given by $Q_2(l,v)$. We assume that each of them is independent for simplicity. Then, the probability $Q_3(l,u,v)$ can be written as follows:

$$Q_3(l,u,v)=(1-Q_2(l,v))^{{}_uC_l}. \tag{26}$$

Similarly, we assume that each of $Q_3(1,u,v)$, $Q_3(2,u,v)$, $\cdots$, and $Q_3(u,u,v)$ is independent for simplicity. Then, the probability $P(u,v)$ that H_{-1}^T has full rank can be written as follows:

$$P(u,v)=Q_3(1,u,v)\cdot Q_3(2,u,v)\cdot\ \cdots\ \cdot Q_3(u,u,v). \tag{27}$$

When p is much smaller than 1 ($p\ll 1$), the probabilities $Q_1(l)$, $Q_2(l,v)$, $Q_3(l,u,v)$, and $P(u,v)$ in (24), (25), (26), and (27) can be approximated as

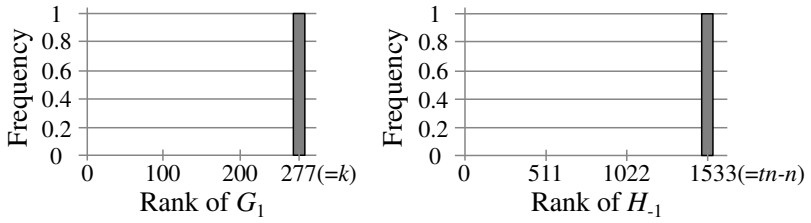

Figure 14: Frequency distribution of the rank of G_1 (left) and H_{-1} (right). G_1 and H_{-1} have full rank in all of the 1000 cases.

follows:

$$\begin{aligned} Q_1(l) &\approx (1-p)^l && (28)\\ Q_2(l,v) &\approx (1-p)^{lv} && (29)\\ Q_3(l,u,v) &\approx 1 - {}_uC_l(1-p)^{lv} && (30)\\ P(u,v) &\approx 1 - ({}_uC_1(1-p)^v + {}_uC_2(1-p)^{2v} + \cdots) \\ &\approx 1 - u(1-p)^v. && (31) \end{aligned}$$

Since the probability that elements in a specific row are $\mathbf{0}$ is $(1-p)^v$, the 2nd term of the right side of (31) approximates the probability that there exists a row in H_{-1}^T whose elements are $\mathbf{0}$. This probability is almost equal to 0 (and hence $P(u,v) \approx 1$) when v is sufficiently large. We also verified by numerical calculations that both $P(u,v)$ in (27) and $P(u,v)$ in (31) converge to 1 as v increases.

Thus, the matrix H_{-1} has full rank (and hence the matrix G_1 also has full rank) in most cases when we use the LDPC code with sufficiently large $v = tn - k$. In the next subsection, we show that the matrices G_1 and H_{-1} indeed have full rank in most cases through experiments.

5.2.2. Experimental analysis

We carried out experiments to verify whether the matrices G_1 and H_{-1} indeed have full rank in most cases. In our experiments in Section 4, we used the $(2044, 277, 788)$ LDPC code at the 4th input in the case of the 128-bit security (see Figure 10). In this setting, we randomly generated both the generator matrix G and the parity check matrix H for 1000 times, and examined the rank of G_1 and H_{-1} for each case. Figure 14 shows the results. It can be seen that G_1 and H_{-1} have full rank in all of the 1000 cases, which

supports the theoretical analysis in Section 5.2.1. As a consequence, we can say that the partial codeword search attack in Section 5.1 is only as strong as the codeword search attack in our experiments.

Note that it is (very unlikely but) still theoretically possible that the matrix H_{-1} is rank deficient, as analyzed in Section 5.2.1. However, even in such cases, the proposed scheme is secure against the partial codeword search attack if the rank of H_{-1} is very large. Recall that the entropy of valid partial codewords k' is $k' = \mathrm{rank}(G_1)$. By Proposition 1, if $\mathrm{rank}(H_{-1}) > tn - n - \epsilon$ ($\epsilon > 0$), then $k' = \mathrm{rank}(G_1) > k - \epsilon$. Thus, *if ϵ is very small, the strength of the partial codeword search attack is almost the same as that of the codeword search attack*. For example, it is (very unlikely but) theoretically possible that there is a row in H_{-1}^T whose elements are $\mathbf{0}$ (which makes H_{-1} rank deficient). However, even such a event increases ϵ by only one. Thus, we can conclude that the proposed scheme, whose security against the codeword search attack is shown in Section 4, is also secure against the partial codeword search attack.

5.3. Security against the partial feature search attack

We have so far considered the security of the proposed scheme against the partial codeword search attack. However, we can consider another attack against multiple protected templates as follows. Suppose that the attacker guesses some n' ($< n$)-bit part of the first biometric feature $X_1 \in \{0,1\}^n$, which we call a *partial feature*. Then, he/she adds the partial feature to the corresponding n' bits of AD_t ($1 \leq t \leq T$), and checks whether the resulting n'-bit string (i.e. the n'-bit part of $X_1 \oplus \mathrm{AD}_t$) is *valid* (i.e. it is a part of a valid codeword $C(S_t)$) or not. Figure 15 shows the n'-bit part of X_1, AD_t, and $C(S_t)$ (gray area). If the n'-bit part of $X_1 \oplus \mathrm{AD}_t$ is not valid, the attacker discards the guess, and attempts another guess for the partial feature. By doing so, the attacker may narrow down the search for X_1. We refer to this attack as the *partial feature search attack*.

We analyze the security of the proposed scheme against this attack. Recall that G ($k \times tn$ matrix) is a generator matrix of the (tn, k, d_{min}) linear error-correcting code C (i.e. $C(S_t) = S_t G$). Let G'_1 ($k \times n'$ matrix) be a *partial generator matrix* corresponding to the n'-bit part of $C(S_t)$ (i.e. $S_t G'_1$ generates the n'-bit part of $C(S_t)$), and G'_{-1} ($k \times (tn - n')$ matrix) be the remaining part of G. We can theoretically show that G'_1 has full rank (i.e. $\mathrm{rank}(G'_1) = \min(k, n')$) in most cases in the same way as Section 5.2.1 (we omit the proof). Thus, in the following, we explain that the proposed scheme

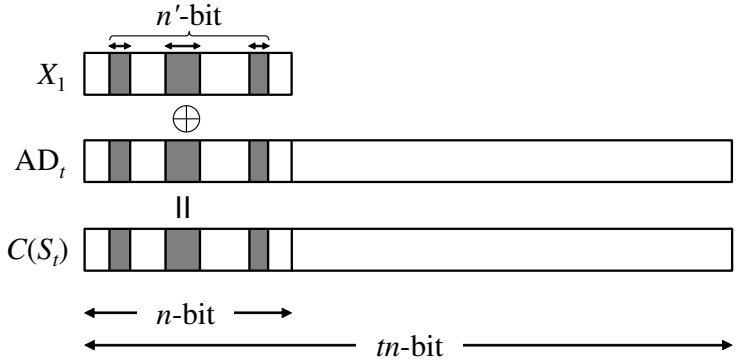

Figure 15: n'-bit part of X_1, AD_t, and $C(S_t)$ (gray area).

is secure against the partial feature search attack if G'_1 has full rank and the number of information symbols k is sufficiently large.

Firstly, suppose that G'_1 has full rank and $n' < k$. Then, $\mathrm{rank}(G'_1) = \min(k, n') = n'$. Thus, there are $2^{n'}$ possible strings as an n'-bit part of a valid codeword $C(S_t)$. In other words, *the n'-bit part of $X_1 \oplus AD_t$ is always valid, and hence the attacker cannot narrow down the search for X_1 using the partial feature search attack.*

Secondly, suppose that G'_1 has full rank and $n' \geq k$. Then, $\mathrm{rank}(G'_1) = \min(k, n') = k$, and there are 2^k possible strings as an n'-bit part of a valid codeword $C(S_t)$. In other words, *the entropy of the n'-bit part of $C(S_t)$ is k, which is the same as the entropy of $C(S_t)$.* Thus, the attacker cannot sufficiently reduce the entropy of the partial feature (i.e. n'-bit part of X_1) if k is sufficiently large. For example, suppose that the attacker prepares a large number of biometric features $Z^{(1)}, \cdots, Z^{(M)} \in \{0,1\}^n$ (M is the number of biometric features), and uses the n'-bit part ($n' \geq k$) of each biometric feature $Z^{(m)}$ ($1 \leq m \leq M$) as a guess value. Then, by the partial feature search attack, he/she can obtain only a set of valid codewords computed from $Z^{(m)} \oplus \mathrm{AD}_t$ (since the entropy of the n'-bit part of $C(S_t)$ ($= k$) is the same as the entropy of $C(S_t)$, a set of valid codewords computed from the n'-bit part of $Z^{(m)} \oplus \mathrm{AD}_t$ is equal to a set of valid codewords computed from $Z^{(m)} \oplus \mathrm{AD}_t$). If he/she chooses the most likely codeword from them, then this attack is equivalent to the statistical attack [18, 20], which is described in Section 6.2. If k is sufficiently large, the proposed scheme is secure against the statistical attack, as discussed in Section 6.2. Similarly, if k is sufficiently

large, the proposed scheme is secure against the partial feature search attack.

To sum up, if G'_1 has full rank and k is sufficiently large, the proposed scheme is secure against the partial feature search attack. Since G'_1 has full rank in most cases (which can be shown in the same way as Section 5.2.1), and k is very large ($k = 128$, 80, or 64) in our experiments in Section 4, we can say that the proposed scheme is secure against the partial feature search attack.

5.4. Security against other attacks

We finally discuss the security of the proposed scheme against other possible attacks. We partition a generator matrix G ($k \times tn$ matrix) and a parity check matrix H ($(tn-k) \times tn$ matrix) as follows: $G = [G_1|G_2|\cdots|G_t]$ and $H = [H_1|H_2|\cdots|H_t]$, where G_i is a $k \times n$ matrix and H_i is a $(tn-k) \times n$ matrix ($1 \leq i \leq t$). The partial codeword search attack, which is described in Sections 5.1 and 5.2, is threatening if the rank of G_1 is low (and the entropy of valid partial codewords k' is small). The attacker may try to generalize this attack to an attack that is threatening if the rank of G_i ($1 \leq i \leq t$) is low. In other words, the attacker may try to utilize the generator matrix G_i with the lowest rank for his/her attack.

However, we can show that each G_i ($1 \leq i \leq t$) has full rank (i.e. $\mathrm{rank}(G_i) = k, \forall i = 1, \cdots, t$) in most cases as follows. Let G_{-i} ($k \times (tn-n)$ matrix) be a matrix that is obtained by concatenating $G_1, \cdots, G_t$ except for G_i (i.e. $G_{-i} = [G_1|\cdots|G_{i-1}|G_{i+1}|\cdots|G_t]$). Similarly, let H_{-i} ($(tn-k) \times (tn-n)$ matrix) be a matrix that is obtained by concatenating $H_1, \cdots, H_t$ except for H_i (i.e. $H_{-i} = [H_1|\cdots|H_{i-1}|H_{i+1}|\cdots|H_t]$). Then, we can prove that for any $\epsilon > 0$ and any i ($1 \leq i \leq t$), we have

$$\mathrm{rank}(H_{-i}) > tn - n - \epsilon \Longrightarrow \mathrm{rank}(G_i) > k - \epsilon \tag{32}$$

(and therefore, $\mathrm{rank}(H_{-i}) = tn - n \Longrightarrow \mathrm{rank}(G_i) = k$). We can prove this in exactly the same way as Proposition 1 in Section 5.2.1 (and hence we omit the proof). We can also extend the argument in Section 5.2.1 to show that each H_{-i} ($1 \leq i \leq t$) has full rank (i.e. $\mathrm{rank}(H_{-i}) = tn - n, \forall i = 1, \cdots, t$) in most cases. Thus, each G_i ($1 \leq i \leq t$) has full rank (i.e. $\mathrm{rank}(G_i) = k, \forall i = 1, \cdots, t$) in most cases. It is (very unlikely but) still theoretically possible that H_{-i} is rank deficient. However, even in such cases, we can say that the rank of G_i is very close to k (since $\mathrm{rank}(G_i) > k - \epsilon$ and ϵ is very small), as discussed in Section 5.2.2. Thus, we can conclude that the proposed scheme,

whose security against the codeword search attack is shown (i.e. k is very large) in Section 4, is also secure against an attack that utilizes the generator matrix G_i with the lowest rank.

We can also generalize the partial feature search attack, which is described in Section 5.3, to an attack that tries to narrow down the search for the i-th biometric feature $X^i \in \{0,1\}^n$ $(1 \leq i \leq t)$ as follows: (i) Guess some n' $(< n)$-bit part of X^i (i.e. partial feature); (ii) Add the partial feature to the corresponding n' bits of AD_t; (iii) Check whether the resulting n'-bit string is *valid* (i.e. it is a part of a valid codeword $C(S_t)$) or not. We can easily extend the arguments in Section 5.3 to show that the proposed scheme is also secure against this attack.

Finally, we consider what we call the *partial FAR attack*, which is somewhat similar to the partial feature search attack (described in Section 5.3). We partition auxiliary data $\mathrm{AD}_t \in \{0,1\}^{tn}$ and a codeword $C(S_t) \in \{0,1\}^{tn}$ as follows: $\mathrm{AD}_t = [\mathrm{AD}_t^1 | \mathrm{AD}_t^2 | \cdots | \mathrm{AD}_t^t]$ and $C(S_t) = [C(S_t)^1 | C(S_t)^2 | \cdots | C(S_t)^t]$, where $\mathrm{AD}_t^i, C(S_t)^i \in \{0,1\}^n$ $(1 \leq i \leq t)$. In the partial FAR attack, the attacker first guesses a value of the i-th biometric feature $X^i \in \{0,1\}^n$. Let $\widetilde{X^i} \in \{0,1\}^n$ a guess value of X^i. Then, the attacker computes a guess value $\widetilde{C(S_t)^i} \in \{0,1\}^n$ of $C(S_t)^i$ as follows: $\widetilde{C(S_t)^i} = \widetilde{X_i} \oplus \mathrm{AD}_t^i$. Finally, the attacker tries to recover S_t from $\widetilde{C(S_t)^i}$ using, for example, G_i and H_i (i.e. the i-th submatrix of G and H, respectively). This attack differs from the partial feature search attack in that the attacker attempts to decode $\widetilde{C(S_t)^i}$ to obtain S_t (instead of verifying whether $\widetilde{C(S_t)^i}$ is valid).

If G_i is a generator matrix of some (efficient) linear ECC, the attacker can decode $\widetilde{C(S_t)^i}$ and recover S_t. However, a linear ECC does not generally have a property that a submatrix G_i of a generator matrix G is also a generator matrix of some linear ECC (if G_i should be a generator matrix of some linear ECC, we could obtain a very efficient linear ECC that corrects total errors while correcting partial errors). In other words, in general, G_i is not a generator matrix of some linear ECC, and therefore the attacker cannot decode $\widetilde{C(S_t)^i}$ to recover S_t. For example, if $G_i H_i^T = \mathbf{0}$, G_i is a generator matrix and H_i a parity check matrix corresponding to G_i. However, as we can see below, $G_i H_i^T$ *is not* $\boldsymbol{0}$ in general. Since G and H are the generator matrix and the parity check matrix, respectively, we have

$$GH^T = \mathbf{0} \Longleftrightarrow \sum_{i=1}^{t} G_i H_i^T = \mathbf{0}. \tag{33}$$

However, *(33) does not imply* $G_i H_i^T = \boldsymbol{0}$. Thus, in general, $G_i H_i^T$ is not $\mathbf{0}$, which means that H_i *is not a parity check matrix corresponding to* G_i. Therefore, the attacker cannot recover S_t from $\widetilde{C(S_t)}^i$ using G_i and H_i.

From another perspective, we can regard $\widetilde{C(S_t)}^i \in \{0,1\}^n$ as a guess value of the codeword $C(S_t) \in \{0,1\}^{tn}$ that has $(t-1)n$*-bit erasures* (i.e. we can regard guess values of $C(S_t)^1, \cdots, C(S_t)^{i-1}, C(S_t)^{i+1}, \cdots, C(S_t)^t$ as erasures). For example, in our experiments in Section 4, we can regard $\widetilde{C(S_4)}^i$ $(1 \leq i \leq 4)$ as a guess value of $C(S_4)$ that has $(4-1) \times 511 = 1533$-bit erasures $(n = 511)$. Since there are too many erasures, the attacker cannot recover S_4 from the guess value of $C(S_4)$. For example, in the case of the 64-bit security, we used the $(2044, 64, 898)$ LDPC code $(\delta_t = 448)$ at the 4th input (see Figure 10). Since the number of erasures is more than $2\delta_t$ (i.e. $1533 > 2 \times 448$), the attacker fails to recover S_4 from the guess value of $C(S_4)$. Thus, the attacker cannot perform the partial FAR attack.

As a conclusion, we can say that the proposed scheme is secure against the partial codeword search attack, the partial feature search attack, their generalizations, and the partial FAR attack.

6. Security of the fuzzy commitment scheme

In our experiments in Section 4, we used the fuzzy commitment scheme. However, some studies proposed attacks against the fuzzy commitment scheme, such as the soft-decoding attack [18, 19], the statistical attack (or ECC (error-correction code) histogram attack) [18, 20], and the decodability attack (or cross-matching attack) [21, 22]. Thus, we discuss the security of the proposed scheme applied to the fuzzy commitment scheme against these attacks. We discuss the security against the soft-decoding attack, the statistical attack, and the decodability attack in Sections 6.1, 6.2, and 6.3, respectively.

6.1. Security against the soft-decoding attack

In the soft-decoding attack [18, 19], the attacker prepares a large number of biometric features $Z^{(1)}, \cdots, Z^{(M)} \in \{0,1\}^n$ (M is the number of biometric features), and matches each of them with a protected template (AD, PI) (AD $= X \oplus C(S)$ and PI $= H(S)$, where $X \in \{0,1\}^n$ is an enrolled biometric feature, $S \in \{0,1\}^k$ $(k < n)$ is a secret key, C is an (n, k, d_{min}) linear ECC, and H is a hash function). Specifically, the attacker adds $Z^{(m)}$ $(1 \leq m \leq M)$ to AD (i.e. $Z^{(m)} \oplus$ AD), and runs a *soft decoder* that always outputs

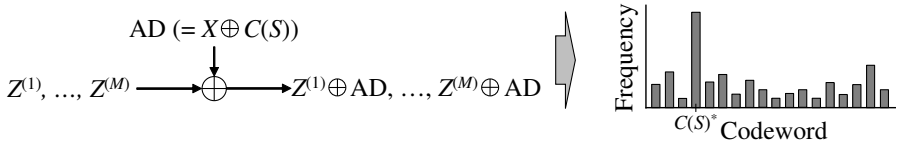

Figure 16: Statistical attack. The attacker computes a histogram of codewords from a commitment AD ($= X \oplus C(S)$) using M biometric features $Z^{(1)}, \cdots, Z^{(M)}$ that he/she prepares, and chooses the most likely codeword $C(S)^*$ as an estimate of $C(S)$.

one or more nearest codewords (even if the number of errors exceeds the distance threshold). Then, the attacker checks whether each of the nearest codewords is a correct codeword $C(S)$ or not by using PI ($= H(S)$). If the attacker finds a correct codeword $C(S)$, he/she can recover the biometric feature X as follows: $X = \mathrm{AD} \oplus C(S)$. In the FAR attack, the attacker fails to decode $Z^{(m)} \oplus \mathrm{AD}$ (and hence, fails to obtain a codeword) if the number of errors exceeds the distance threshold. On the other hand, in the soft-decoding attack, the attacker always obtains one or more nearest codewords even in such cases by using a soft decoder. Thus, the soft-decoding attack has a higher success probability (i.e. the probability of obtaining a correct codeword $C(S)$) than the FAR attack when the attacker searches the same biometric features $Z^{(1)}, \cdots, Z^{(M)}$. However, since nearest neighbor decoding (or maximum likelihood decoding) takes much time in general, the soft-decoding attack takes much more time than the FAR attack.

One way to defend against this attack is to increase the number of information symbols k (as well as the codeword search attack described in Section 3.5). This is because the Voronoi region (also called the Voronoi cell) of a correct codeword $C(S)$ becomes very small (and hence the probability that a soft decoder outputs $C(S)$ becomes very small) if the number of possible codewords 2^k is very large. Since k is very large ($k = 128$, 80, or 64) in our experiments in Section 4, we consider that the proposed scheme is secure against the soft-decoding attack. Also, we note again that the soft-decoding attack takes much more time than the FAR attack.

6.2. Security against the statistical attack

In the statistical attack [18, 20], the attacker runs, for each of $Z^{(m)} \oplus \mathrm{AD}$ ($1 \leq m \leq M$), a soft decoder that always outputs one or more nearest codewords in the same way as the soft-decoding attack. Then, the attacker counts

the number of appearances for each possible codeword, and computes a histogram of codewords. Using the histogram, the attacker chooses a codeword $C(S)^*$ corresponding to the most likely codeword (i.e. histogram maximum). If $C(S)^* = C(S)$, the attacker can recover the biometric feature X as follows: $X = \mathrm{AD} \oplus C(S)^* = \mathrm{AD} \oplus C(S)$. Figure 16 shows an overview of the statistical attack.

In the soft-decoding attack, the attacker checks whether each of the nearest codewords (computed from $Z^{(m)} \oplus \mathrm{AD}$ $(1 \leq m \leq M)$) is correct or not using a pseudo identifier PI. On the other hand, in the statistical attack, the attacker computes the most likely codeword $C(S)^*$ from the nearest codewords, and tries to recover the biometric feature X *without using PI*. As described in Section 1, one of the possible models for storing a protected template (AD, PI) is to store AD and PI separately. In this model, it can happen that the attacker (does not obtain PI but) obtains only AD, and hence cannot perform the soft-decoding attack. The statistical attack can be a threat even in this case, since this attack does not require PI. It should be noted, however, that this paper focuses on the public template model in which both AD and PI are stored into a single place (or made public). In this model, since the attacker can obtain both AD and PI simultaneously, he/she can perform both the soft-decoding attack and the statistical attack.

Also, it is important to note that the statistical attack has a lower success probability than the soft-decoding attack when the attacker searches the same biometric features $Z^{(1)}, \cdots, Z^{(M)}$. The soft-decoding attack checks, for each of the nearest codewords (computed from $Z^{(m)} \oplus \mathrm{AD}$ $(1 \leq m \leq M)$), whether it is a correct codeword $C(S)$ or not using PI. Thus, this attack results in success if the number of appearance for $C(S)$ is more than or equal to 1. On the other hand, the statistical attack results in success only if the number of appearance for $C(S)$ is the highest among all possible codewords. In other words, the statistical attack has a lower success probability than the soft-decoding attack, because it does not perform a checking process using PI. Since the proposed scheme is secure against the soft-decoding attack (as discussed in Section 6.1), we consider that the proposed scheme is also secure against the statistical attack.

As described in Section 6.1, the soft-decoding attack takes much more time than the FAR attack, since nearest neighbor decoding (or maximum likelihood decoding) takes much time in general. Taking this into account, the attacker may run a *hard-decoder* that decodes $Z^{(m)} \oplus \mathrm{AD}$ $(1 \leq m \leq M)$ only if the number of errors is within the distance threshold (as well as the

FAR attack) in the statistical attack. In this case, the statistical attack has a lower success probability than the FAR attack (because it does not perform a checking process using PI). Since the proposed scheme in our experiments in Section 4 is secure against the FAR attack, it is also secure against the statistical attack that runs the hard-decoder.

6.3. Security against the decodability attack

Consider a scenario where the same person uses two biometric authentication systems A and B, both of which use the fuzzy commitment scheme. Let X_A (resp. X_B) $\in \{0,1\}^n$ be the biometric feature at the enrollment phase in the system A (resp. B). The system A (resp. B) computes a commitment $\mathrm{AD}_A = X_A \oplus C(S_A)$ (resp. $\mathrm{AD}_B = X_B \oplus C(S_B)) \in \{0,1\}^n$ ($S_A, S_B \in \{0,1\}^k$ ($k < n$) are secret keys, and C is an (n, k, d_{min}) linear ECC), and stores it into the database. Kelkboom *et al.* [21] proposed the decodability attack (or cross-matching attack), which determines whether AD_A and AD_B are from the same user or not to break the *unlinkability* [40] of the protected templates.

In the decodability attack in [21], the attacker adds AD_A to AD_B, which can be written as follows:

$$\begin{aligned} \mathrm{AD}_A \oplus \mathrm{AD}_B &= (X_A \oplus C(S_A)) \oplus (X_B \oplus C(S_B)) \\ &= \epsilon_{AB} \oplus (C(S_A) \oplus C(S_B)), \end{aligned} \tag{34}$$

where $\epsilon_{AB} = X_A \oplus X_B$. Then, the attacker checks whether $\mathrm{AD}_A \oplus \mathrm{AD}_B$ is decodable (i.e. the number of errors is within the distance threshold δ $(= (d_{min} - 1)/2)$) or not, and decides that AD_A and AD_B are from the same user if and only if it is decodable. Since C is linear, $C(S_A) \oplus C(S_B)$ in (34) is a codeword. Thus, if $\epsilon_{AB} (= X_A \oplus X_B) \leq \delta$, the attacker successfully decodes $\mathrm{AD}_A \oplus \mathrm{AD}_B$, and decides that AD_A and AD_B are from the same user (if $\epsilon_{AB} > \delta$ and ϵ_{AB} is not decodable, the attacker decides that AD_A and AD_B are from different users). Since the distance between X_A and X_B from the same user is small in general, the attacker breaks the unlinkability with high accuracy.

To prevent this attack, Kelkboom *et al.* [21] proposed a bit-permutation randomization process, which shuffles a biometric feature vector using a random permutation matrix that is made public. Let P_A (resp. P_B) be a random permutation matrix ($n \times n$ matrix) in the system A (resp. B). The system A (resp. B) generates a commitment AD'_A (resp. AD'_B) as follows:

$\mathrm{AD}'_A = P_A X_A \oplus C(S_A)$ (resp. $\mathrm{AD}'_B = P_B X_B \oplus C(S_A)$), and stores ($\mathrm{AD}'_A$, P_A) (resp. (AD'_B, P_B)) into the database (at the authentication phase, the system shuffles a biometric feature in the same way). Then, the XOR of the commitments AD'_A and AD'_B can be written as follows:

$$\mathrm{AD}'_A \oplus \mathrm{AD}'_B \;=\; \epsilon'_{AB} \oplus (C(S_A) \oplus C(S_B)), \tag{35}$$

where $\epsilon'_{AB} = P_A X_A \oplus P_B X_B$. Since P_A and P_B are random, we can assume that $P_A X_A$ and $P_B X_B$ are also random, and hence ϵ'_{AB} does not have discriminative information. Kelkboom *et al.* [21] showed that the performance of the decodability attack was close to that of a random classifier when the systems use the bit-permutation randomization process.

The proposed scheme applied to the fuzzy commitment scheme can also prevent the decodability attack by using the bit-permutation randomization process as follows. Let $X_{A,t} \in \{0,1\}^{tn}$ (resp. $X_{B,t} \in \{0,1\}^{tn}$) be the t-th large biometric feature at the enrollment phase in the system A (resp. B). Let further $P_{A,t}$ (resp. $P_{B,t}$) be a random permutation matrix ($tn \times tn$ matrix) at the t-th input in A (resp. B). The system A (resp. B) shuffles $X_{A,t}$ (resp. $X_{B,t}$) using $P_{A,t}$ (resp. $P_{B,t}$), and computes a commitment $\mathrm{AD}'_{A,t} \in \{0,1\}^{tn}$ (resp. $\mathrm{AD}'_{B,t} \in \{0,1\}^{tn}$) at the t-th input. That is, $\mathrm{AD}'_{A,t}$ and $\mathrm{AD}'_{B,t}$ can be written as follows:

$$\mathrm{AD}'_{A,t} \;=\; P_{A,t} X_{A,t} \oplus C(S_{A,t}) \tag{36}$$

$$\mathrm{AD}'_{B,t} \;=\; P_{B,t} X_{B,t} \oplus C(S_{B,t}), \tag{37}$$

where $S_{A,t}, S_{B,t} \in \{0,1\}^k$ ($k < tn$) are secret keys, and C is an (tn, k, d_{min}) linear ECC. The system A (resp. B) stores ($\mathrm{AD}'_{A,t}, P_{A,t}$) (resp. ($\mathrm{AD}'_{B,t}, P_{B,t}$)) ($1 \le t \le T$) into the database. Since $P_{A,1}, \cdots, P_{A,T}, P_{B,1}, \cdots, P_{B,T}$ are random, we can assume that $P_{A,1}X_{A,1}, \cdots, P_{A,T}X_{A,T}, P_{B,1}X_{B,1}, \cdots, P_{B,T}X_{B,T}$ are also random, and hence the proposed scheme is secure against the decodability attack.

We also analyze storage requirements in the proposed scheme when we use the bit-permutation randomization process. In our experiments in Section 4 ($n = 511, T = 4$), the conventional parallel feature level fusion scheme [6, 7, 8, 9] requires a random permutation matrix whose size is 522 kB ($= (4 * 511)^2/8$) per user. On the other hand, the proposed scheme requires random permutation matrices whose size is, in total, 979 kB ($= (1^2 + 2^2 + 3^2 + 4^2) \times 511^2/8$) per user (i.e. 1.9 times as much as the conventional scheme). Since the size of finger-vein data in the proposed scheme is 639 bytes per user (as

discussed in Section 4.6), the size of random permutation matrices (= 979 kB) is relatively large. However, since the storage capacity of the HDD is much larger in recent years (e.g. more than terabytes) and is still increasing, we consider the storage requirements for the proposed scheme do not matter much in practice.

Finally, we note that Tams [22] showed that the binary fuzzy commitment scheme with the bit-permutation randomization process is vulnerable to an attack called the *generalized decodability attack*. To prevent this attack, Tams [22] proposed to use a modification of the bit-permutation randomization process to use a random bijection (instead of random permutation) in a non-binary case, and to use the improved fuzzy vault scheme [1] in a binary case. Similarly, the proposed scheme can also prevent this attack by using the modified randomization process or the improved fuzzy vault scheme.

7. Conclusions

In this paper, we proposed a framework for feature level sequential fusion, and a sequential fusion algorithm that minimizes the average number of inputs, which can be applied to multibiometric cryptosystems. We proved its optimality under three assumptions (Theorem 1). We applied the proposed scheme to the fuzzy commitment scheme, and showed its security (against the FAR attack and the codeword search attack) and convenience through experiments using the finger-vein dataset that contains 6 fingers from 505 subjects (33298 finger-vein images in total). We also analyzed the security of the proposed scheme applied to the fuzzy commitment scheme against various attacks: attacks that exploit the relationship between multiple protected templates (e.g. the partial codeword search attack, the partial feature search attack), the soft-decoding attack [18, 19], the statistical attack [18, 20], and the decodability attack [21, 22]. We believe these results significantly contribute to the secure and convenient public template model.

References

[1] Y. Dodis, R. Ostrovsky, L. Reyzin, A. Smith, Fuzzy extractors: How to generate strong keys from biometrics and other noisy data, SIAM J. Comput. 38 (1) (2008) 97–139.

[2] A. Juels, M. Sudan, A fuzzy vault scheme, in: Proc. IEEE Int. Symposium on. International Symposium on Information Theory (ISIT'02), 2002.

[3] A. Juels, M. Wattenberg, A fuzzy commitment scheme, in: Proc. the 6th ACM conference on Computer and communications security (CCS'99), 1999, pp. 28–36.

[4] U. Uludag, S. Pankanti, S. Prabhakar, A. Jain, Biometric cryptosystems: Issues and challenges, Proc. IEEE 92 (6) (2004) 948–960.

[5] A. Ross, K. Nandakumar, A. K. Jain, Handbook of Multibiometrics, Springer, 2006.

[6] E. Kelkboom, X. Zhou, J. Breebaat, R. Veldhuis, C. Busch, Multi-algorithm fusion with template protection, in: Proc. IEEE 3rd International Conference on Biometrics: Theory, Applications, and Systems (BTAS'09), 2009, pp. 1–7.

[7] J. Merkle, T. A. M. Kevenaar, U. Korte, Multi-modal and multi-instance fusion for biometric cryptosystems, in: Proc. of the International Conference Biometrics Special Interest Group (BIOSIG'12), 2012.

[8] A. Nagar, K. Nandakumar, A. K. Jain, Multibiometric cryptosystems based on feature-level fusion, IEEE Trans. Information Forensics and Security 7 (1) (2012) 255 – 268.

[9] C. Rathgeb, A. Uhl, P. Wild, Reliability-balanced feature level fusion for fuzzy commitment scheme, in: Proc. IEEE/IAPR International Joint Conference on Biometrics (IJCB'11), 2011, pp. 1–7.

[10] G. L. Marcialis, F. Roli, Serial fusion of fingerprint and face matchers, in: Proc. the Seventh International Workshop on Multiple Classifiers Systems (MCS'07), Vol. 4472 of Lecture Notes in Computer Science, 2007, pp. 151–160.

[11] R. M. Bolle, J. H. Connell, S. Pankanti, N. K. Ratha, A. W. Senior, Guide to Biometrics, Springer, 2003.

[12] T. Murakami, K. Takahashi, Accuracy improvement with high convenience in biometric identification using multihypothesis sequential probability ratio test, in: Proc. the First IEEE International Workshop on Information Forensics and Security (WIFS '09), 2009, pp. 66–70.

[13] K. Takahashi, M. Mimura, Y. Isobe, Y. Seto, A secure and user-friendly multimodal biometric system, in: Proc. SPIE, Vol. 5404, 2004, pp. 12–19.

[14] C. J. Hill, Risk of masquerade arising from the storage of biometrics (2001).
URL `http://www.enhyper.com/content/biometricmasquerading.pdf`

[15] T. L. Lai, Asymptotic optimality of invariant sequential probability ratio tests, Ann. Statist 9 (2) (1981) 318–333.

[16] A. Wald, Sequential Analysis, Wiley & Sons, New York, 1947.

[17] T. Yanagawa, S. Aoki, T. Ohyama, Diversity of human finger vein patterns and its application to personal identification, Bulletin of Informatics and Cybernetics 41 (2007) 1–9.

[18] A. Stoianov, T. Kevenaar, M. van der Veen, Security issues of biometric encryption, in: Proceedings of the 2009 IEEE Toronto International Conference on Science and Technology for Humanity (TIC-STH'09), 2009, pp. 34–39.

[19] A. Stoianov, Security of error correcting code for biometric encryption, in: Proceedings of the 8th Annual International Conference on Privacy Security and Trust (PST'10), 2010, pp. 231–235.

[20] C. Rathgeb, A. Uhl, Statistical attack against fuzzy commitment scheme, IET Biometrics 1 (2) (2012) 94–104.

[21] E. J. Kelkboom, J. Breebaart, T. A. Kevenaar, I. Buhan, R. N. Veldhuis, Preventing the decodability attack based cross-matching in a fuzzy commitment scheme, IEEE Transactions on Information Forensics and Security 6 (107–121).

[22] B. Tams, Decodability attack against the fuzzy commitment scheme with public feature transforms, CoRR abs/1406.1154.
URL http://arxiv.org/abs/1406.1154

[23] A. K. Jain, K. Nandakumar, A. Nagar, Biometric template security, EURASIP Journal on Advances in Signal Processing (2008) 1–17.

[24] J. Daugman, How iris recognition works, IEEE Trans. Circuits and Systems for Video Technology 14 (2004) 21–30.

[25] F. Hao, R. Anderson, J. Daugman, Combining crypto with biometrics effectively, IEEE Trans. Computers 55 (9) (2006) 1081–1088.

[26] A. Ross, R. Govindarajan, Feature level fusion using hand and face biometrics, in: Proc. the SPIE Conference on Biometric Technology for Human Identification, 2005, pp. 196–204.

[27] K. Nandakumar, A. K. Jain, A. Ross, Fusion in multibiometric identification systems: What about the missing data?, in: Proc. 3rd International Conference on Advances in Biometrics (ICB '09), 2009, pp. 743–752.

[28] S. Ben-Yacoub, Y. Abdeljaoued, E. Mayoraz, Fusion of face and speech data for person identity verification, IEEE Trans. Neural Networks 10 (5) (1999) 1065–1074.

[29] J. Kittler, M. Hatef, R. P. Dulin, J. Matas, On combining classifiers, IEEE Trans. Pattern Analysis and Machine Intelligence 20 (3) (1998) 226–239.

[30] L. Lam, C. Y. Suen, Application of majority voting to pattern recognition: An analysis of its behavior and performance, IEEE Trans. Systems, Man, and Cybernetics, Part A: Systems and Humans 27 (5) (1997) 553–568.

[31] V. P. Dragalin, A. G. Tartakovsky, V. V. Veeravalli, Multihypothesis sequential probability ratio tests, part I: Asymptotic optimality, IEEE Trans. Information Theory (1999) 2448–2461.

[32] W. W. Peterson, E. J. Weldon, Error-Correcting Codes, MIT Press, 1972.

[33] D. J. MacKay, Information Theory, Inference and Learning Algorithms, Cambridge University Press, 2003.

[34] X.-Y. Hu, M. P. Fossorier, E. Eleftheriou, On the computation of the minimum distance of low-density parity-check codes, in: Proceedings of the 2004 IEEE International Conference on Communications (ICC'04), Vol. 2, 2004, pp. 767 – 771.

[35] Y. Yin, L. Liu, X. Sun, SDUMLA-HMT: a multimodal biometric database, in: Proceedings of the 6th Chinese Conference on Biometric Recognition (CCBR'11), 2011, pp. 260–268.

[36] A. Kumar, Y. Zhou, Human identification using finger images, IEEE Trans. Image Processing 21 (4) (2012) 2228–2244.

[37] N. Miura, A. Nagasaka, T. Miyatake, Feature extraction of finger-vein patterns based on repeated line tracking and its application to personal identification, Machine Vision and Applications 15 (2004) 194–203.

[38] P. J. Bickel, K. A. Doksum, Mathematical Statistics, Prentice Hall New Jersey, 1976.

[39] J. Hanley, A. Lippman-Hand, If nothing goes wrong, is everything alright?, Journal of the American Medical Association 249 (1984) 1743–1745.

[40] ISO/IEC 24745:2011: Information technology – security techniques – biometric information protection.