Near-collisions and their Impact on Biometric Security

DURBET Axel[1], GROLLEMUND Paul-Marie[2], LAFOURCADE Pascal[1] and THIRY-ATIGHEHCHI Kevin[1]

[1]*Université Clermont-Auvergne, CNRS, Mines de Saint-Étienne, LIMOS, France*
[2]*Université Clermont-Auvergne, CNRS, LMBP, France*

Abstract: Biometric recognition encompasses two operating modes. The first one is biometric identification which consists in determining the identity of an individual based on her biometrics and requires browsing the entire database (*i.e.*, a 1:*N* search). The other one is biometric authentication which corresponds to verifying claimed biometrics of an individual (*i.e.*, a 1:1 search) to authenticate her, or grant her access to some services. The matching process is based on the similarities between a fresh and an enrolled biometric template. Considering the case of binary templates, we investigate how a highly populated database yields near-collisions, impacting the security of both the operating modes. Insight into the security of binary templates is given by establishing a lower bound on the size of templates and an upper bound on the size of a template database depending on security parameters. We provide efficient algorithms for partitioning a leaked template database in order to improve the generation of a master-template-set that can impersonates any enrolled user and possibly some future users. Practical impacts of proposed algorithms are finally emphasized with experimental studies.

1   Introduction

With the continuous growth of biometric sensor markets, the use of biometrics is becoming increasingly widespread. Biometric technologies provide an effective and user-friendly means of authentication or identification through the rapid measurements of physical or behavioral human characteristics. For biometric identification and authentication schemes, biometric templates of users are registered with the system. The first operating mode consists in determining the identity of an individual based on similarity scores calculated from all the enrolled templates and the fresh provided template. The latter corresponds to the verification of the claimed identity based on a similarity score calculated from the assigned enrolled template and a fresh template. As a consequence, service providers need to manage biometric databases in a manner similar to managing password databases.

The leak of biometric databases is more dramatic since, unlike passwords, biometric data serve as long term identifiers and cannot be easily revoked. The consequences of stolen biometric templates are impersonation attacks and the compromise of privacy. Essential security and performance criteria that must be met by biometric recognition systems are identified in ISO/IEC 2474 (ISO, 2011) and ISO/IEC 30136 (ISO, 2018): *Irreversibility*, *unlinkability*, *revocability* and *performance preservation*.

Biometric templates are generated from biometric measurements (*e.g.*, a fingerprint image). They result from a chain of treatments, an extraction of the features (*e.g.*, using Gabor filtering (Manjunath and Ma, 1996; Jain et al., 2000)) followed eventually by a Scale-then-Round process (Ali et al., 2020) to accommodate better handled representations, *i.e.,* binary or integer-valued vectors. These templates are then protected either through their mere encryption, or using a Biometric Template Protection (BTP), *e.g.,* a cancelable biometric transformation such as Biohashing (Jin et al., 2004; Lumini and Nanni, 2007) or any other salting method. For more details on BTP schemes, the reader is referred to the surveys (Nandakumar and Jain, 2015; Natgunanathan et al., 2016; Patel et al., 2015). The use of a BTP scheme is in general preferred since its goal is to address the aforementioned criteria. However, note that cancelable biometric transformations are prone to inversion attacks, at least in the sense of second-preimages (Durbet et al., 2021). They even lead sometimes to the compromise of privacy with a good approximation of a feature vector (Lacharme et al., 2013; Ghammam et al., 2020).

Recent works have also demonstrated that recog-

nition systems are vulnerable to dictionary attacks based on master-feature vectors (Roy et al., 2017; Bontrager et al., 2018). A master-feature vector is a set of synthetic feature vectors that can match with a large number of other feature vectors. This can naturally be extended to the problem of generating master-templates and masterkeys. The notion of masterkey has recently been addressed in (Gernot and Lacharme, 2021) to produce backdoors with the aim of implementing biometric-based access rights. In the same topic, the present paper analyses the security of biometric databases by making some recommandations, and by proposing attacks using the notions of master-template, master-feature and masterkey.

Contributions. Our main contribution is an efficient partitioning algorithm which accelerates attacks aiming to generate master-key or master-feature vector. Numerical studies on implementations of the proposed algorithm show a reduction of the computational time by a factor of up to 38 in certain settings. In addition, we show a link with the closest string problem with an arbitrary number of words, for which we provide a solution using Simulated ANNealing (SANN). Moreover, we determine a bound on the size of a database in function of the template space dimension and the decision threshold, thus preventing near-collisions with a high probability. Specifically, for a secure database, the recommanded template size is $n = 512$ bits with a threshold of the order of 10% of n, *i.e.,* around 50 bits. Setting these paramaters in this way rules out attacks based on master-templates and ensures a good recognition accuracy. Finally, some indications are provided for handling basic database operations such as addition or deletion of users.

Outline. In Section 2, we introduce some notations, background material as well as definitions of new notions such as master-template and ε-covering-template. In Section 3, we describe an algorithm which provides a segmentation of a database in order to focus on potential master-templates. In Section 4, we show how this algorithm can be used to improve the computation of masterkey-set and of the master-feature-set. Moreover, we describe how near-collisions can be used to define a secure parameter k which depends on the template space dimension and a threshold. We also explain why the secure parameter is a countermeasure and, the case of a user which is added or removed from the database are studied. In Section 5, we provide some experimentations in order to assess the performance of the proposed algorithm and to detail in practice how the near string problem is solved.

2 Preliminaries

A biometric system is a method of authentication or identification based on biometric data. The main idea is to transform the biometric data into a template to match the four aforementioned criteria, i.e., irreversible, unlinkable, revocable and performance preservation. It must be able to compare template and determine if they belong to the same person. The template is constructed by combining a feature vector derived from the biometric data and a secret parameter named token which can be for example a password. A biometric authentication or identification system always starts by using a *feature extraction scheme* to extract some information from the biometric image to construct a feature vector (Ratha et al., 2001). A database partitioning method can be applied to each biometric system for this. In this paper, we focus on templates expressed as binary vectors, but the results below can be adapted to every template representations.

In the following, we let $(\mathcal{M}_I, \text{Dist}_I)$, $(\mathcal{M}_F, \text{Dist}_F)$ and $(\mathcal{M}_T, \text{Dist}_T)$ be three metric spaces, where $\mathcal{M}_I$, $\mathcal{M}_F$ and $\mathcal{M}_T$ represent the image space, the feature space and the template space, respectively; and Dist_I, Dist_F and Dist_T are the respective distance functions. Note that Dist_I and Dist_F are instantiated with the Euclidean distance, while Dist_T is instantiated with the Hamming distance.

Definition 2.1 (Feature extraction scheme). *A biometric feature extraction scheme is a pair of deterministic polynomial time algorithms* $\Pi := (E, V)$, *where:*

- *E is the feature extractor of the system, that takes biometric data $I \in \mathcal{M}_I$ as input, and returns a feature vector $F \in \mathcal{M}_F$.*

- *V is the verifier of the system, that takes two feature vectors $F = E(I)$, $F' = E(I')$, and a threshold τ_F as input, and returns True if $\text{Dist}_F(F, F') \leq \tau_F$, and returns False if $\text{Dist}_F(F, F') > \tau_F$.*

For the sake of privacy, biometric data (the feature vector) should be designed in a such way that it prevents information leakage. This motivates the use of a cancelable biometric transformation scheme.

Definition 2.2 (Cancelable biometric transformation scheme). *Let $\mathcal{K}$ be the token (seed) space, representing the set of tokens to be assigned to users. A cancelable biometric scheme is a pair of deterministic polynomial time algorithms* $\Xi := (\mathcal{T}, \mathcal{V})$, *where:*

- *$\mathcal{T}$ is the transformation of the system, that takes a feature vector $F \in \mathcal{M}_F$ and the token parameter $P \in \mathcal{K}$ as input, and returns a biometric template $T = \mathcal{T}(P, F) \in \mathcal{M}_T$.*

- $\mathcal{V}$ is the verifier of the system, that takes two biometric templates $T = \mathcal{T}(P,F)$, $T' = \mathcal{T}(P',F')$, and a threshold τ_T as input; and returns $True$ if $\mathrm{Dist}_T(T,T') \leq \tau_T$, and returns $False$ if $\mathrm{Dist}_T(T,T') > \tau_T$.

In this paper, the template space is, unless otherwise specified, $\mathbb{F}_2^n = \left(\mathbb{Z}/2\mathbb{Z}\right)^n$, equipped with the Hamming distance denoted by d_H. As the template space is a metric space, we denote it as $(\mathbb{F}_2^n, d_H)$. In our case, the verifier is the Hamming distance, but the transformation does not need to be specified. As we work on a set of template, we denote it as *Template DataBase* (*TDB*).

Definition 2.3 (Template database or TDB). *Let (Ω, d) be the template space equipped with the distance d. A subset $L \subset \Omega$ such that $L \neq \emptyset$ and $L \neq \Omega$ is a template database (TDB), or just a database.*

As with hash functions, an antecedent of a transform can be searched in order to steal a password or a pass tests using this hash function. This preimage can be the exact feature vector or a nearby preimage.

Definition 2.4 (Template preimage). *Let $I \in \mathcal{M}_I$ be a biometric image, and $T = \Xi.\mathcal{T}(P, \Pi.E(I)) \in \mathcal{M}_T$ for some secret parameter P. A template preimage of T with respect to P is a biometric image I^* such that $T = \Xi.\mathcal{T}(P, \Pi.E(I^*))$.*

Definition 2.5 (Nearby template preimage). *Let $I \in \mathcal{M}_I$ be a biometric image, a threshold ε_B, and $T = \Xi.\mathcal{T}(P, \Pi.E(I)) \in \mathcal{M}_T$ for some secret parameter P. A nearby template preimage of T with respect to P is a biometric image I^* such that $d(T, \Xi.\mathcal{T}(P, \Pi.E(I^*))) < \varepsilon_B$.*

The goal of an attacker can be to create a masterkey-set. This is a set of tokens that allow to build all the templates of a targeted database using the same feature vector.

Definition 2.6 (Masterkey). *Let $D = \{v_i\}_{i=1,\dots,n}$ be a template database where $v_i := \Xi.\mathcal{T}(x_i, s_i)$ generated with distinct tokens $S = \{s_i\}_{i=1,\dots,n}$ and distinct biometric features $X = \{x_i\}_{i=1,\dots,n}$, and let τ_B be a threshold. Then, m is a masterkey for D, with respect to τ_B, if $\forall i \in [\![1,n]\!], \Xi.\mathcal{V}(\Xi.\mathcal{T}(x_i,m), \Xi.\mathcal{T}(x_i,s_i), \tau_B) = True$.*

Furthermore, in this context another objective of an attacker can be to find a masterkey-set or a master-feature-set, which are defined below.

Definition 2.7 (Masterkey-set). *Let $D = \{v_i\}_{i=1,\dots,n}$ be a template database where $v_i := \Xi.\mathcal{T}(x_i,s_i)$ generated with distinct tokens $S = \{s_i\}_{i=1,\dots,n}$ and distinct biometric features $X = \{x_i\}_{i=1,\dots,n}$, and let τ_B a threshold. The set D is said covered*

by a set of r masterkeys $\{k_1, \dots, k_r\}$ with respect to τ_B, if $\forall i \in [\![1,n]\!], \exists j \in [\![1,r]\!]$ such that $\Xi.\mathcal{V}(\Xi.\mathcal{T}(x_i,k_j), \Xi.\mathcal{T}(x_i,s_i), \tau_B) = True$.

Another goal of an attacker can be to create a master-feature-set. This is a set of feature that allow to build all the templates of a targeted database using preferably the same token.

Definition 2.8 (Master-feature). *Let $D = \{v_i\}_{i=1,\dots,n}$ be a template database where $v_i := \Xi.\mathcal{T}(x_i,s_i)$ generated with distinct tokens $S = \{s_i\}_{i=1,\dots,n}$ and distinct biometric features $X = \{x_i\}_{i=1,\dots,n}$, and let τ_B a threshold. Then, m is a master-feature for D, with respect to τ_B, if $\forall i \in [\![1,n]\!], \Xi.\mathcal{V}(\Xi.\mathcal{T}(m,s_i), \Xi.\mathcal{T}(x_i,s_i), \tau_B) = True$.*

Definition 2.9 (Master-feature-set). *Let $D = \{v_i\}_{i=1,\dots,n}$ be a template database where $v_i := \Xi.\mathcal{T}(x_i,s_i)$ generated with distinct tokens $S = \{s_i\}_{i=1,\dots,n}$ and distinct biometric features $X = \{x_i\}_{i=1,\dots,n}$, and let τ_B a threshold. The set D is said covered by a set of r master-features $\{k_1, \dots, k_r\}$ with respect to τ_B, if $\forall i \in [\![1,n]\!], \exists j \in [\![1,r]\!]$ such that $\Xi.\mathcal{V}(\Xi.\mathcal{T}(k_j,s_i), \Xi.\mathcal{T}(x_i,s_i), \tau_B) = True$.*

Targeting random template to find a masterkey are often not efficient, thus, to maximize the efficiency of the research of a masterkey-set, we suggest to focus on ε-covering templates.

Definition 2.10 (ε-cover-template). *Let D be a template database and a distance d. An ε-cover-template of D is x such that $d(x,a) \leq \varepsilon, \forall a \in D$.*

Note that, there are cases for which there is no possible ε-cover-template.

Example 2.0.1. *Let $\varepsilon = 1$, $n = 3$ and $D = \{(0,0,0);(0,1,1);(1,0,1);(1,1,0)\} \subset \mathbb{F}_2^3$. In this case, we have $\forall a,b \in D, d_H(a,b) \leq 2$, but there is no template x such that $\forall a \in D, d_H(x,a) \leq 1$. The 1-cover template does not exist. But, if we remove $(0,0,0)$ from D then $x = (1,1,1)$ is an 1-cover template for D.*

As the ε-covering template is non-unique, we also consider ε-covering template-sets.

Definition 2.11 (ε-cover-template-set). *Let D be a template database and d a distance. An ε-cover-template-set of D is $\mathcal{C}$ such that $\forall u \in D$ and $\forall x \in \mathcal{C}$, $d(u,x) \leq \varepsilon$.*

To construct a partition of a template database, we introduce strong and weak notions of ε-master-template.

Definition 2.12 (ε-master-template or ε-MT). *Let (Ω, d) be the template space and D a template database. A template $t \in \Omega$ is an ε-master-template if $\forall t' \in D, d(t,t') \leq \varepsilon$.*

Definition 2.13 (ε-master-template-set or ε-MTS).
Let (Ω, d) *be the template space and D a template database. A subset* $T \subset \Omega$ *is an ε-master-template-set if* $\forall t' \in D, \exists t \in T$ *such that* $d(t, t') \le ε$.

Note that an ε-master-template-set is a non-empty set: D is an ε-master-template-set of itself but an ε-master-template of D could be empty. Moreover, an ε-cover-template is an ε-master-template and an ε-master-template-set is a set of ε-cover-templates which are not in the same ε-cover-template-set. We define a near-collision and more precisely multiple-near-collision.

Definition 2.14 (Near collision). *Let* (Ω, d) *be the template space and a threshold* ε *. There exists a* near-collision *if* $\exists a, b \in \Omega \mid d(a, b) \le ε$.

Definition 2.15 (*m*-near-collision). *Let* (Ω, d) *be the template space and a threshold* ε *. There exists an m-near-collision if* $\exists a_1, \ldots, a_m \in \Omega$ *such that* $\forall i$ *and* $j \in \{1, \ldots, m\}, d(a_i, a_j) \le ε$.

Thus, the search of an ε-cover-template of D a database corresponds to the search of an at least $|D|$-near-collision for which each template of D is related to the collision.

3 Database Partitioning

The aim of this part is to determine the smallest ε-covering-template-set for a given database D.

3.1 Agglomerative Clustering

Consider M_D the dissimilarity matrix of a template database D, for the Hamming distance. The dissimilarity matrix M_D is used to compute template clusters, denoted by $C_ε$, for which the distance between two templates in the same cluster is at most s. To perform this clustering, we use the agglomerative clustering method which is a type of the hierarchical clustering. This method consists in successively agglomerating the two closest groups of templates. It begins with $|D|$ groups, one for each template, and it terminates when all the groups are merged as a unique one.

A standard post-processing is required to define at which iteration the algorithm should be terminated so that a relevant set of template clusters is obtained. However, we define a termination condition so that the clustering algorithm stop when it is not possible anymore to obtain templates cluster verifying the following required property: $\forall i \in [\![1, n]\!]$, $\forall a, b \in C_i, max(d_H(a, b)) \le s$. The Agglomerative Clustering algorithm we used then corresponds to a slight

variation of the HACCLINK (Hierarchical Agglomerative Clustering Complete LINK) presented in (Defays, 1977).

By using the aforementioned clustering method, we obtain a set of template clusters, for which the inner-cluster distance suggests that it could exist at least one master-template for these templates. An additional step is described below whose aim is to determine potential master-templates, if there exists some.

3.2 Master-Template of a Template Group

We consider having a group of templates verifying $\forall i \in [\![1, n]\!]$, $\forall a, b \in C_i$, $max(d_H(a, b)) \le s$, and for which we aim at finding a master-template. We emphasize that this problem can be formulated as a modified case of closest-string problem which is defined as follows.

Definition 3.1 (Modified closest-string problem). *Given* $S = \{s_1, s_2, \ldots, s_m\}$ *a set of strings with length n and d a distance, find a center string t of length m such that for every string s in S, $d_H(s, t) \le d$*.

The closest-string problem is known as an *NP*-hard problem (Frances and Litman, 1997), and there exist algorithms to solve that kind of problem, see among others (Meneses et al., 2004; Gramm et al., 2001).

Definition 3.2 (Closest-string problem). *Given* $S = \{s_1, s_2, \ldots, s_m\}$ *a set of strings with length n, find a center string t of length m minimizing d such that for every string s in S, $d_H(s, t) \le d$*.

According to the link between both problems defined in Definition 3.2 and 3.1, we can establish that the issue addressed in this paper is a hard problem, which is specified in the following theorem, whose proof is given in the Appendix A.

Theoreme 3.1 (MCSP is NP-hard). *The modified closest-string problem is NP-hard*.

To the best of our knowledge, this problem has not been addressed in the literature, then we propose an algorithm to solve it. Moreover, with regards to Theorem 3.1, we deem that relying on brute force type algorithm could not be efficient and that more parsimonious algorithm must be investigated, notably stochastic algorithms. However, more efficient upcoming methods could replace this part without affecting the remainder of the database partitioning method proposed in Section 3.

We consider $D = \{v_1, \ldots, v_k\}$ be a template database and C the ε-cover-template-set for D. The approach described below provides a constructive

definition of the elements of C, if $C \neq \emptyset$. In particular, the following result emphasizes the link between C and the balls $B_i = \{u \in \mathbb{F}_2^n | d_H(u, v_i) \leq \varepsilon\}$. The proof is given in Appendix A.

Theoreme 3.2 (C is the intersection of the balls of radius ε). *Let $D = \{v_1, \ldots, v_k\}$ be a template database and C the ε-cover-template-set for D.*
Then, $C = \cap_{i \in \{1, \ldots, k\}} B_i$.

We denote by $p \in C$ a master-template, and Theorem 3.2 indicates that determining all the master-template p reduces to determining the intersection of k Hamming balls, which turns out to be formulated as the solutions of the following system:

$$d_H(p, v_i) \leq \varepsilon, \quad \forall i \in \{1, \ldots, k\}. \tag{1}$$

Notice that System 1 is a linear system, hence we can rely on a binary ILP (Integer Linear Programming) to solve it and then to compute C.

However, solving this system could be time-consuming in real world cases since there are as many parameters as the length of p, i.e., the dimension n of $\mathbb{F}_2^n$. Therefore, we suggest reducing System 1 by removing dependent variables. To do so, the following necessary notations are introduced:

- For $K = \{k_1, \ldots, k_{|K|}\} \subset \{1, \ldots, n\}$, the Hamming distance over K is denoted by: $\forall u, v \in \mathbb{F}_2^n, d_K = d_H((u_{k_1}, \ldots, u_{k_{|K|}}), (v_{k_1}, \ldots, v_{k_{|K|}}))$.

- Let $\mathcal{P}_D(K)$ a statement about $K \subset \{1, \ldots, n\}$. We said that $\mathcal{P}_D(K)$ holds if $\forall u, v \in D, d_K(u, v) \in \{0, |K|\}$. Examples of subsets K for which $\mathcal{P}_D(K)$ holds or not are provided in Appendix B.

- Consider I the smallest partition $\{(K_1, \ldots, K_{|I|}), K_i \subset \{1, \ldots, n\} | \forall i \in \{1, \ldots, |I|\}\}$ such that $\mathcal{P}_D(K_i)$ holds for all $i \in \{1, \ldots, n\}$. As I is the smallest possible partition, it allows us to reduce the dimension of System 1 as much as it is possible.

- For $p \in \mathbb{F}_2^n$ and $v \in D$, $n_{v,i}$ denotes $d_{K_i}(p, v)$ and n_v^I denotes the parameters vector $(n_{v,1}, \ldots, n_{v,|I|})$, written $N = (n_1, \ldots, n_{|I|})$ for short when the context is clear.

- The vector of distances $(d_H(v_1, v), \ldots, d_H(v_{|D|}, v))$ is denoted by $d(v)$ with $v \in D$ and $D = (v_1, \ldots, v_{|D|})$.

Then, with these notations, Theorem 3.3 can be established as follows, specifying a smaller version of System 1.

Theoreme 3.3. *For a given template database D and for a given $v \in D$, consider $L = \{p \in \mathbb{F}_2^n | AN \leq \varepsilon - d(v)\}$ with $N = n_v^I$, $\varepsilon = (\varepsilon, \ldots, \varepsilon)^T$, $n_{v,i}$ denotes $d_{K_i}(p, v)$, n_v^I denotes the parameters vector*

$(n_{v,1}, \ldots, n_{v,|I|})$ *and $A = (a_{i,j})$ a matrix of size $|I| \times |D|$ whose the $(i, j)^{th}$ element is*

$$a_{i,j} = \begin{cases} 1 & \text{if } d_{K_j}(v_1, v_i) = 0 \\ -1 & \text{if } d_{K_j}(v_1, v_i) = |K_j| \end{cases}$$

Then, $L = C$ the ε-cover-template-set for D.

Proof is detailed in Appendix A.

As I is required to reduce System 1, we assure with Lemma 3.1 that $I \neq \emptyset$, whatever the configuration of the set D is.

Lemme 3.1 (I is not empty). $\forall D \subset \mathbb{F}_2^n$ *such that $|D| > 1$, $I \neq \emptyset$.*

Proof is given in Appendix A. In the same vein, one can determine that $|I| \leq n$. As $|I|$ corresponds to the number of parameters, the system described in Theorem 3.3 is always smaller or equivalent to System 1.

Theorem 3.3 indicates that determining the ε-cover-template-set for D (which corresponds to an intersection of $|D|$ balls in $\mathbb{F}_n^2$ can be reduced to solving a potentially small linear system. While the resolution of the aforementioned system can be done with powerful tools (like GUROBI (Pedroso, 2011)), we deem that simpler algorithms should be used in this case. In particular, according to the configuration of D, it is possible to obtain a such system linear that it is straightforward to determine the space of the potential solutions and to find a solution with any Markovian scanning algorithm. More precisely, if $\mathcal{N}$ denotes the set of the possible solutions N for the linear system described in Theorem 3.3, we have:

$$\mathcal{N} = \prod_{k=1}^{|I|} \{0, \ldots, \min(\varepsilon, |K_k|)\}$$

since, for $k \in \{1, \ldots, |I|\}$, $n_{v,k}$ corresponds to the distance $d_{K_k}(v_k, v)$, which can not be greater than $|K_k|$, and in the other hand if $d_{K_k}(v_k, v) > \varepsilon$ then, N does not belong to L. One can then be aware that depending on the dimension of $\mathcal{N}$, finding a solution N can be efficiently done via either a brute force algorithm in case of small dimensional set $\mathcal{N}$, or via a more parsimonious algorithm if the dimension is high. As the dimension of $\mathcal{N}$ depends among other factors on D, we consider that the use of one of the both approaches should be determined with regards to pratical context-specific consideration. In this paper, we only describe an algorithm to use in case of high dimensional $\mathcal{N}$ set. We propose to rely on an efficient and simple algorithm: the Simulating Annealing algorithm (Kirkpatrick et al., 1983). Nevertheless, even if we illustrate the proposed methodology with this

algorithm, it could be replaced by any optimization algorithm based on scanning the space. Below we detail features of Simulating Annealing algorithm that we tune in order to obtain good performances in our numerical study. It is composed of the following parameters:

- *Energy:* We define the following energy so that larger it is, the closer N is to solve the linear system:

$$E(N) = \sum_{i=1}^{|I|} f((\varepsilon - d(v) - AN)_i) \qquad (2)$$

where f is a ReLU type function: $f(x) = \min(0, x)$.

- *Cooling Schedule:* In practice, we observe that finding a solution is not sensitive to the cooling of the system, see Section 5.3. Then, we propose to choose a linear decreasing temperature. The starting temperature is fixed so that at the initial iteration, all potential move must be accepted, whatever the chosen initial point is.

- *Proposal distribution:* According to computational considerations and for the sake of numerical performance, we define a proposal distribution for which the support is the neighbors set. Moreover, we choose a non-symmetric proposal that preferentially promotes neighbors that increases the energy (2).

- *Termination:* The algorithm is terminated either it reaches the maximum iteration number (about 200k iterations), or if a solution is found, which corresponds to a vector N with a null energy.

The experimentations of this part are presented in Section 5.3.

3.3 Database Partitioning Algorithm

Using the developments of the sections 3.1 and 3.2, we propose Algorithm 1 to partition the template database. It takes as inputs D a template database and a threshold ε and returns an ε-MTS.

4 Attack Scenario, Countermeasure and Case Studies

The aim of this section is to show that the method described Section 3 eases the computation of a masterkey-set or a master-feature-set. Their computations are straightforward in the absence of BTP scheme and are still possible if an invertible transformation is employed, like Biohashing or some other

---

Algorithm 1: Database partitioning algorithm

   Data: D, ε
   Result: MTS
1 Set s to 2ε.
2 Set MTS to [ ].
3 **while** $D \neq \emptyset$ **do**
4   Compute cluster *Cls* using D and s.
5   **foreach** *cluster c in Cls* **do**
6   Search the cover template t for c.
7   **if** *a cover template t is found for* $c \in C$ **then**
8   Set D to $D \backslash c$ and add t to MTS.
9   **end**
10   Set s to $s - 1$.
11   **end**
12 **end**
13 **return** *MTS*.

---

salting transformations. Moreover, that kind of attack is analyzed, and a security bound is established in Section 4.2.

4.1 Attack Scenario

Consider a pair of functions $\mathcal{T}_1^{-1}$ and $\mathcal{T}_2^{-1}$ defined as follows :

Definition 4.1 (Token transformation inversion function). *The* token transformation inversion function *denoted by* $\mathcal{T}_1^{-1}$ *takes* $v \in \mathcal{M}_{\mathcal{F}}$ *a feature vector and* $t \in \Omega$ *a template and gives* p *a token such that* $\mathcal{T}(v, p) = t$.

Definition 4.2 (Feature transformation inversion function). *The* token transformation inversion function *denoted by* $\mathcal{T}_2^{-1}$ *takes* p *a token and* $t \in \Omega$ *a template and gives* $v \in \mathcal{M}_F$ *a feature vector such that* $\mathcal{T}(v, p) = t$.

Note that we focus on frameworks for which $\mathcal{T}_1^{-1}$ and $\mathcal{T}_2^{-1}$ can be computed in a reasonable time: at least linear and at most subexponential. These functions must be determined case-by-case according to the used biometric transformation. Furthermore, an attacker seeking to create a master-feature-set (resp. a masterkey-set) can do it using k calls to the inverse transformation function $\mathcal{T}_1^{-1}$ (resp. $\mathcal{T}_2^{-1}$), where k is the number of templates. However, the method developed in Section 3 can be used to reduce the computation complexity. Actually, the attacker can compute a master-feature-set or a masterkey-set in only ℓ step with $\ell \leq k$, where l is the number of clusters.

4.2 Countermeasure: Managing the Database Size

Consider a biometric system set with a template space of size n and a threshold ε. Moreover, suppose that the biometric system is unbiased i.e., each template is randomly chosen in the template space. There exists a maximum size for a database at n and ε fixed which minimizes the gain of an attacker with the method presented in Section 3 and which maximizes the size of that database. Notice that the following approach can be applied to any biometric system.

4.2.1 Prevent an Advantage

An advantage of an attacker is significant when our database partitioning method (Section 3) reduces the complexity of the initial attack by at least one. Let k be the number of clients allowed in a database and, $\mathbb{F}_2^n$ the template space. If $k \geq \left\lceil 2^n / \sum_{i=0}^{\varepsilon} \binom{n}{i} \right\rceil$ then, there is at least one cluster containing two or more templates, according to the Dirichlet's box principle. In our case, c is at most: $\left\lceil 2^n / \sum_{i=0}^{\varepsilon} \binom{n}{i} \right\rceil$ and there are two scenarios:

1. There are enough clients to find a coverage of $\mathbb{F}_2^n$ by using their clusters and any other enrollment is already compromised.

2. There is not enough clients to find a coverage of $\mathbb{F}_2^n$ and the attacker obtains an advantage for the computation.

By using birthday problem, more particularly the probability of a near collision (Lamberger et al., 2012; Lamberger and Teufl, 2012), we can establish that, the average number of template must be about $2^{(n+1)/2} S_\varepsilon(n)^{-1/2}$ so that a cluster contained two templates, where, $\sum_{i=0}^{\varepsilon} \binom{n}{i} = S_\varepsilon(n)$. Furthermore, the number of near collisions is $N_C(\varepsilon)$ and its expected value $\mathbb{E}(N_C(\varepsilon))$ is equal to $\binom{k}{2} S_\varepsilon(n) 2^{-n}$ with k the number of templates. Thus, the number k of templates which give a collision with a probability of 50% is

$$\approx 2^{n/2} S_\varepsilon(n)^{-1/2} \tag{3}$$

Figure 1 provides numerical and graphical representations based on experimentations, enlightening on how k behaves relatively to n and ε. They show that the size of a database which can provide collisions is wide smaller than the size of n. Furthermore, if ε is bigger than 20% of n, this size dramatically decreases. To keep enough room in a safe database, n must be larger than 512 and ε smaller than 51.

4.3 Case Study: Addition of Users in the Database

Let D be a database and C a master-template-set for D with respect to a threshold ε. As the proposed method of Section 3 works incrementally, the clustering can be repeated each time there is a new arrival in the database. However, considering the cost of such an operation, we propose a more efficient algorithm yielding a near-optimal solution. Let us denote by t the new enrolled template. The steps for adding t are as follows:

1. Compute the distance d_i between each center $c_i \in C$ and t.

2. If there exists $d_i \leq \varepsilon$ then nothing to do, *i.e.,* t has a representative.

3. Otherwise, add the singleton master-template-set t to C.

To minimize C once a certain number of clients have been added, the partitioning algorithm of Section 3 is applied on C and D, and the smallest resulting set is kept. Note however that since its computation is faster on C than on D due to its smaller cardinal, it is better to start with C.

4.4 Case Study: Removing a User from the Database

Removing a user from the database is a more complex problem. Indeed, if the template t_u of a user u is removed from the database, that may mean that she has unsubscribed or she has been banned from the service. In any case, all the templates in her ball must be systematically rejected. Thus, there is a problem if the templates $t_{u'}$ of other users u' are such that $B_{u'} \cap B_0 \neq \emptyset$ with $B_0 = B(t_u, \varepsilon)$ and $B_{t_{u'}} = B(t_{u'}, \varepsilon)$. If the number of templates in the database plus the number of removed templates is at most $2^{n/2} S_\varepsilon(n)^{-1/2}$, this case happens with a small probability. In this case, all users have an increased False Rejection Rate (FRR) proportionally to the intersection between their ball and B_0. Then, as this case is uncommon, for the comfort of the users and to keep the initial FRR, it is recommended to re-enroll the affected user(s). Note that if a large amount of persons are unsubscribed from the database, the size of the template space is reduced, precisely by $\sum_{i=0}^{\varepsilon} \binom{n}{i}$ for each removed user. Furthermore, when the number of removed clients k_1 and the number of clients in the database k_2 are such that $k_1 + k_2 \approx 2^{n/2} S_\varepsilon(n)^{-1/2}$, the system must be changed, or no more users should be accepted as recommended in Section 4.2.
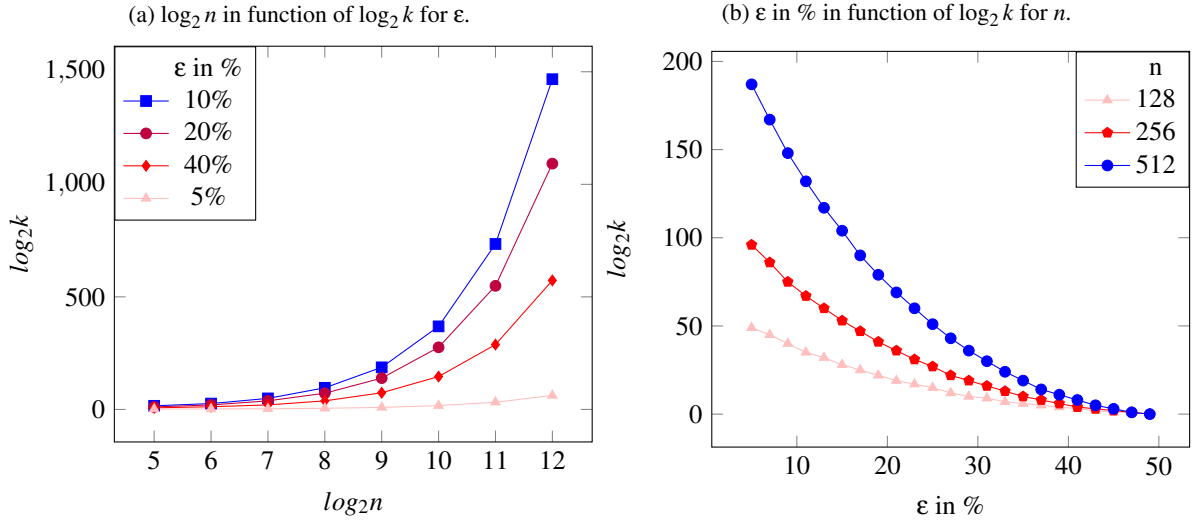
(a) $\log_2 n$ in function of $\log_2 k$ for ε.   (b) ε in % in function of $\log_2 k$ for n.

Figure 1: Link between k and n or ε.

5 Attack Evaluation

In this section we provide some experimental evaluations of Algorithm 1 and, we discuss our results. In our experiments, the passwords are assumed unique for each individual. The hashed passwords serve as seeds for the generation of the matrices. Thus, the produced templates are uniformly distributed.

5.1 Naive Greedy Approach

To compare the efficiency of our proposal with a baseline, we propose a naive algorithm based on a greedy strategy. First, a template is picked from the template database. Then, all templates in the template database which are at a distance of at most ε from the chosen template are removed. These steps are repeated as long as there are templates in the database. As a result, the chosen templates form the MTS.

5.2 Evaluation of the Database Partitioning

Templates are randomly drawn from $\mathbb{F}_2^n$. For each configuration, experimentations are replicated $10\,000$ times and averaged results are computed. The average results are presented in Table 1 and Table 2 with the following notations:

- n: the space dimension.
- ε: the threshold.
- #*clients*: the number of templates in the TDB.
- #*clust*: the number of clusters found with Algorithm 1.

- #*clust*(G): the number of clusters found using the greedy Algorithm 5.1.
- *Efficiency* is the ratio #*clust*(G)/#*clust*.
- *Time* is the running time of Algorithm 1.
- *Time*(G) is the running time of the greedy Algorithm 5.1.

As the computation of the ε-cover-template 3.2 is the most expensive part of Algorithm 1, an experimentation Table 2 is dedicated to the ε-cover-template search 3.2. In fact, we remark that the gain of the attacker is greater when the value of k is greater that what we recommend in Section 4.2.

5.3 Evaluation of Simulated Annealing

Keeping the same notations, the average experimentations are stored in Table 3. In the case where the simulated annealing is used as a sub-routine of the algorithm 1, this latter is slower and less efficient. The main reason of this loss of performance is the error rate of the simulated annealing which forces doing more calculations. However, it is quicker and more efficient than solving a system to answer to the near string problem given in Section 3.1.

Moreover, we use several cooling functions (from (Aarts et al., 2005; Kirkpatrick et al., 1983)) to determine what is preferable. We draw randomly $10\,000$ times some templates in a ball of radius ε and a template as a center. These templates except the center form the simulated database. By doing so, we are sure that the database enables an ε-cover-template. Then, for each database generated, an ε-cover-template is sought. The results of Table 4

n	ε	#clients	#clust	#clust(G)	Efficiency	Time (ms)	Time G (ms)
15			1.000	27.352	×**27.35**	1 239.629	7.182
20			2.700	35.433	×**13.12**	8 415.270	10.714
25			5.260	44.965	× 8.55	7 085.071	16.302
30	10	50	8.709	48.977	× 5.70	8 775.802	18.940
35			12.838	49.872	× 3.90	6 870.666	21.636
40			18.087	49.986	× 2.77	6 417.596	23.762
45			23.217	49.998	× 2.14	6 773.971	25.629
	3		200.000	200.000	× 1.00	42.252	558.165
	5		200.000	200.000	× 1.00	43.969	449.166
	10		198.280	200.000	× 1.00	799.243	329.992
70	15	200	90.000	200.000	× 2.22	47 016.050	337.082
	20		46.543	199.985	× 4.30	94 364.038	350.846
	25		22.109	198.982	× **9.00**	222 386.614	346.420
	30		10.000	182.891	×**18.29**	37 925.217	326.385
	35		3.600	137.583	×**38.20**	949 691.855	380.933
		30	29.96	30		24.318	16.446
		50	49.83	50		46.362	40.018
		70	69.78	70		78.995	80.496
		90	89.67	90		136.572	137.186
50	10	110	109.60	110	×1	208.685	189.203
		130	129.30	130		428.885	251.221
		150	148.93	150		670.493	335.790
		170	168.79	170		531.363	434.727
		190	188.46	190		625.553	562.672

Table 1: Summary of the experiments of the space partitioning algorithm.

n	ε	#clients	Time (ms)	n	ε	#clients	Time (ms)	n	ε	#clients	Time (ms)
15			1 277.473		3		24 901.480			70	8 474.519
20			1 592.213		5		24 949.724			90	11 087.893
25			2 033.807		10		22 279.933			110	17 464.348
30	10	50	2 428.682	70	15	200	20 978.806	70	10	130	18 330.508
35			2 758.428		20		21 028.472			150	19 009.142
40			3 887.738		25		29 089.280			170	20 887.950
45			3 866.077		30		31 586.784			190	19 539.516

Table 2: Summary of the experiments of the ε-cover-template search algorithm ILP version.

in Appendix indicate that finding a solution is not strongly sensitive to the cooling method of the system. However, all cooling methods give similar results.

6 Concluding Remarks

In this paper, we have performed an in-depth analysis of the Hamming space as template space. We first have introduced some formal definitions such as multiplicative near-collision, master-template, ε-covering template and some technical terms and concepts. We then have proposed an algorithm to perform a partition of the set of templates. This partitioning can be used to improve either the masterkey-set search or the master-feature-set search. The proposed center search algorithm using simulated annealing is also a result of independent interest for solving the near-string-problem. If an additional and secure authentication factor is put into place, this partitioning may be used in the authentication mode to compress the template database. This compression is achieved by replacing each template by a reference of its cluster center.

By relying on the properties of near-collisions and the partitioning algorithm, we have also shown there exists a security bound on the size of a database that depends on both the space dimension and the decision threshold. Beyond that limit on the size, there is a high probability of a near collision that impacts

n	ε	#clients	Error in %	Time (ms)	n	ε	#clients	Error in %	Time (ms)	n	ε	#clients	Error in %	Time (ms)
15			16.34	1201		3		0.15	36			70	1.95	6
20			0.64	17		5		0.00	36			90	0.14	12
25			0.00	1		10		0.00	35			110	0.00	18
30	10	50	0.00	1	70	15	200	0.00	36	70	10	130	0.00	22
35			0.00	1		20		0.00	37			150	0.00	27
40			0.05	1		25		0.00	40			170	0.00	31
45			0.36	2		30		0.00	40			190	0.00	34

Table 3: Summary of the experiments of the ε-cover-template search algorithm SANN version.

both security and efficiency. Identification and authentication systems are negatively affected by near-collisions, especially identification in terms of recognition accuracy. A countermeasure when the number of templates exceeds the recommended security parameter is to extend the size of the template space. This can be transparently done for the user if the risk has been anticipated in the design of the protocol. For instance, in the case of projection-based transformations, the enrollment template can easily be extended using a part of an extended fresh template.

Acknowledgement

REFERENCES

(2011). ISO/IEC24745:2011: Information technology – Security techniques – Biometric information protection. Standard, International Organization for Standardization.

(2018). ISO/IEC30136:2018(E): Information technology – Performance testing of biometrictemplate protection scheme. Standard, International Organization for Standardization.

Aarts, E., Korst, J., and Michiels, W. (2005). *Simulated Annealing*, pages 187–210.

Ali, S., Karabina, K., and Karagoz, E. (2020). Formal accuracy analysis of a biometric data transformation and its application to secure template generation. In Samarati, P., di Vimercati, S. D. C., Obaidat, M. S., and Ben-Othman, J., editors, *Proceedings of the 17th International Joint Conference on e-Business and Telecommunications, ICETE 2020 - Volume 2: SECRYPT, Lieusaint, Paris, France, July 8-10, 2020*, pages 485–496. ScitePress.

Bontrager, P., Roy, A., Togelius, J., Memon, N., and Ross, A. (2018). Deepmasterprints: Generating masterprints for dictionary attacks via latent variable evolution. In *2018 IEEE 9th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, pages 1–9. IEEE.

Defays, D. (1977). An efficient algorithm for a complete link method. *The Computer Journal*, 20(4):364–366.

Durbet, A., Lafourcade, P., Migdal, D., Thiry-Atighehchi, K., and Grollemund, P.-M. (2021). Authentication attacks on projection-based cancelable biometric schemes.

Frances, M. and Litman, A. (1997). On covering problems of codes. *Theory of Computing Systems*, 30(2):113–119.

Gernot, T. and Lacharme, P. (2021). Biometric masterkeys.

Ghammam, L., Karabina, K., Lacharme, P., and Thiry-Atighehchi, K. (2020). A cryptanalysis of two cancelable biometric schemes based on index-of-max hashing. *IEEE Transactions on Information Forensics and Security*, PP:1–12.

Gramm, J., Niedermeier, R., and Rossmanith, P. (2001). Exact solutions for closest string and related problems. pages 441–453.

Jain, A., Prabhakar, S., Hong, L., and Pankanti, S. (2000). Filterbank-based fingerprint matching. *IEEE Transactions on Image Processing*, 9(5):846–859.

Jin, A. T. B., Ling, D. N. C., and Goh, A. (2004). Biohashing: two factor authentication featuring fingerprint data and tokenised random number. *Pattern Recognition*, 37(11):2245–2255.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

Lacharme, P., Cherrier, E., and Rosenberger, C. (2013). Preimage Attack on BioHashing. In *SECRYPT 2013 - Proceedings of the 10th International Conference on Security and Cryptography, Reykjavík, Iceland, 29-31 July, 2013*, pages 363–370.

Lamberger, M., Mendel, F., Rijmen, V., and Simoens, K. (2012). Memoryless near-collisions via coding theory. *Designs, Codes and Cryptography*, 62(1):1–18.

Lamberger, M. and Teufl, E. (2012). Memoryless near-collisions, revisited.

Lumini, A. and Nanni, L. (2007). An improved BioHashing for human authentication. *Pattern Recognition*, 40(3):1057 – 1065.

Manjunath, B. S. and Ma, W. Y. (1996). Texture features for browsing and retrieval of image data. 18(8):837–842.

Meneses, C., Lu, Z., Oliveira, C., and Pardalos, P. (2004). Optimal solutions for the closest string problem via

integer programming. *Informs Journal on Computing - INFORMS*, 16:419–429.

Nandakumar, K. and Jain, A. K. (2015). Biometric template protection: Bridging the performance gap between theory and practice. *IEEE Signal Processing Magazine*, 32:88–100.

Natgunanathan, I., Mehmood, A., Xiang, Y., Beliakov, G., and Yearwood, J. (2016). Protection of privacy in biometric data. *IEEE Access*, 4:880–892.

Patel, V. M., Ratha, N. K., and Chellappa, R. (2015). Cancelable biometrics: A review. *IEEE Signal Processing Magazine*, 32(5):54–65.

Pedroso, J. P. (2011). Optimization with gurobi and python. *INESC Porto and Universidade do Porto, Porto, Portugal*, 1.

Ratha, N. K., Connell, J. H., and Bolle, R. M. (2001). An analysis of minutiae matching strength. In Bigun, J. and Smeraldi, F., editors, *Audio- and Video-Based Biometric Person Authentication*, pages 223–228, Berlin, Heidelberg. Springer Berlin Heidelberg.

Roy, A., Memon, N., and Ross, A. (2017). Masterprint: Exploring the vulnerability of partial fingerprint-based authentication systems. *IEEE Transactions on Information Forensics and Security*, 12(9):2013–2025.

A Omitted Proofs

Proof of Theorem 3.1. Let A be the oracle for the modified closest-string problem and (S) a closest-string problem instance. Thus, on at most n calls to A, the closest-string problem can be solved. In fact, the solver B of the closest-string problem sends to A the following instances: $(S,1),(S,2),\ldots,(S,i \leq n)$ and stops at the i-th instance for which A finally comes with the solution t. Then, B returns the pair (t,i), solution to the initial problem. Since B can be reduced in polynomial time to A and B is NP-hard, A is also NP-hard. The reduction is trivial in the other direction. $\square$

Proof of Theorem 3.2. Let $p \in \cap_{u \in D} B(u, \varepsilon)$. Then, $\forall u \in D, p \in B(u, \varepsilon)$. Which implied that $\forall u \in D, d_H(p,u) \leq \varepsilon$. And so, p is an ε-cover-template for D. Then, $p \in \mathcal{C}$ which implies that $\cap_{u \in D} B(u, \varepsilon) \subset \mathcal{C}$. Moreover, let $p \in \mathcal{C}$. Then, $\forall u \in D, d_H(p,u) \leq \varepsilon$. So, $\forall u \in D, p \in B(u, \varepsilon)$. Thus, $P \in \cap_{u \in D} B(u, \varepsilon)$ and, $\mathcal{C} \subset \cap_{u \in D} B(u, \varepsilon)$. Then, using both inclusion, $\mathcal{C} = \cap_{u \in D} B(u, \varepsilon)$. $\square$

Proof of Lemma 3.1. Let $D \subset \mathbb{F}_2^n$ be a template database such that $|D| > 1$ and $K_i = \{i\}, \forall i \in \{1,\ldots,n\}$. Therefore, $\sqcup_{i \in \{1,\ldots,n\}} K_i = \{1,\ldots,n\}$ and, $\forall i \in \{1,\ldots,n\}, \mathcal{P}_D(K_i) = \text{True}$. Then, $I = \{K_i, \forall i \in \{1,\ldots,n\}\} \neq \emptyset$. $\square$

Proof of Theorem 3.3. Let D be a database, $u \in D$ a template, $p \in \mathbb{F}_2^n$ a template and, $A_K(u) = \{v \in D | d_K(u,v) = 0\}$. There are two cases:

1. If $v \in A_K(u)$ then, $d_k(u,v) = 0$.
2. Else, $v \in A_K(u)^c$ then, $d_k(u,v) = |K|$.

If $v \in A_K(u)^c$ then, $d_K(v,p) = |K| - d_K(p,u)$. However, as I is a partition of $\{1,\ldots,n\}$, $d_H(u,p) = \sum_{K \in I} d_K(u,p)$.

Suppose that $p \in C$ the ε-cover-template-set for D. As $max_{u \in D} d_H(u,p) \leq \varepsilon$ then, $\sum_{K \in I} d_K(u,p) \leq \varepsilon$.

Thus, for $v \in D$, $d_K(v,p) = d_K(p,u)\mathbb{1}_{A_K(u)}(v) + (|K| - d_K(u,p))\mathbb{1}_{A_K(u)^c}(v)$. Then, for a given couple (u,v), we have:

$$
\begin{aligned}
\sum_{K \in I} d(v,p) &= \sum_{K \in I} d_K(p,u)\mathbb{1}_{A_K(u)}(v) \\
&\quad + (|K| - d_K(u,v))\mathbb{1}_{A_K(u)^c}(v) \\
&= \left(\sum_{K \in I} d_K(p,u)(\mathbb{1}_{A_K(u)}(v) - \mathbb{1}_{A_K(u)^c}(v)) \right) \\
&\quad + \sum_{K \in I} |K|\mathbb{1}_{A_K(u)^c}(v)
\end{aligned}
$$

Moreover, $d_H(u,v) = \sum_{K \in I} |K|\mathbb{1}_{A_K(u)^c}(v)$ then,

$$
\sum_{K \in I} d(v,p) = \sum_{K \in I} d_K(p,u)(\mathbb{1}_{A_K(u)}(v) - \mathbb{1}_{A_K(u)^c}(v)) + d_H(u,v).
$$

Then,

$$
\sum_{K \in I} d_K(v,p) \leq \varepsilon
$$
$$
\Leftrightarrow \sum_{K \in I} d_K(p,u)(\mathbb{1}_{A_K(u)}(v) - \mathbb{1}_{A_K(u)^c}(v)) \leq \varepsilon - d_H(u,v)
$$
$$
\Leftrightarrow A(u)d_K(p,u) \leq \varepsilon - d_H(u,v)
$$
$$
\Leftrightarrow p \in L
$$
$\square$ $\square$

B Illustration for the Partitioning of the Set of Indices

The following example serves as an illustration for Section 3.2. Let $D = \{(1,0,1,1,0,1,1), (1,0,0,1,0,1,1), (1,0,1,1,1,1,1), (1,0,0,1,1,1,0)\}$ be a database represented as a matrix with the templates in rows. The identical or opposite columns are labelled with the same symbol, as follows:

	1	2	3	4	5	6	7
v_1	1	0	1	1	0	1	1
v_2	1	0	0	1	0	1	1
v_3	1	0	1	1	1	1	1
v_4	1	0	0	1	1	1	0
	♠	♠	■	♠	★	♠	▲

We remind the statement $\mathcal{P}_D(K)$ which holds if for all templates of D, their pairwise distance is $|K|$ or 0. Let $K = \{1, 2, 4, 6\}$ be the set of columns marked with a ♠. Then, $\mathcal{P}_D(K)$ holds. However, for $K = \{3, 7\}$, $\mathcal{P}_D(K)$ does not hold. In fact, if K is uniquely comprised of columns having the same symbol, the statement holds. If the columns which are identical or opposite are merged together, the property $\mathcal{P}_D(K)$ holds. Finally, in this example, the partition I is $\{\underbrace{\{1, 2, 4, 6\}}_{♠}, \underbrace{\{3\}}_{■}, \underbrace{\{5\}}_{★}, \underbrace{\{7\}}_{▲}\}$.

C Comparison of Cooling Strategies

Some experimentations with our Simulated Annealing process have been done using different temperatures and the same number of steps. The objective is to figure out which cooling strategy is preferable. Table 4 indicates that the different options do not significantly impact the efficiency of the Simulated Annealing algorithm.

Cooling method	n	ε	#*clients*	Error in %	Time (ms)
	45	10	50	0.1	7
	50	10	50	0.6	11
Additive cooling	55	10	50	3.1	9
	60	10	50	8.3	11
	65	10	50	47.2	19
	45	10	50	0.6	12
	50	10	50	0.8	11
Linear multiplicative	55	10	50	3.7	5
	60	10	50	5.3	16
	65	10	50	35.3	18
	45	10	50	0.2	7
	50	10	50	1.2	10
Exponential	55	10	50	4.3	3
	60	10	50	6.9	4
	65	10	50	40.8	15
	45	10	50	0.5	6
	50	10	50	1.4	3
Logarithmic	55	10	50	2.9	3
	60	10	50	6.6	10
	65	10	50	40.8	10

Table 4: Comparison of cooling methods for our simulated annealing.