Long-lived nearby-template preimages on biometric transformation with genetic algorithm

Tanguy Gernot, Patrick Lacharme

Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC, 14000 Caen, France

Abstract

We study the security of biometric data's transformations, especially about BioHashing. We perform several preimage attacks with genetic algorithm. Long-lived nearby-template preimage are fairly precisely approximated in a reasonable time.

Keywords : Biometric transformation security, Biohashing, Genetic algorithms

1 Introduction

The use of biometric systems has exploded in recent years. The data handled there is sensitive, because biometric data cannot been changed like passwords. Several protection systems were invented to prevent biometric data recovery. The goal of biometric transformation is to ensure security properties like revocability, noninvertible transformation (oneway transform), cross application and so on. For example, biohashing is a popular cancelable biometric scheme, used as biometric transformation [11].

Without loss of generality, a biometric transformation maps a real-values vector into binary template using random seed. The comparison of binary templates uses the Hamming distance. A long-lived nearby-template preimage x is recovered by an attacker from the knowledge of a template and the corresponding seed, if the map of x using a second seed is near to a second template (unknown to the attacker).

A long-lived nearby-template preimage is recovered by performing different attacks based on genetic algorithm. We compare the performance of our genetic attacks with different choices of parameters and regarding to an adaptative-plaintext attack.

On section 2, we discuss about the different idea find in the literature, especially about the input and the parameters to give to genetic algorithm. On section 3, we will talk about the implementation of the algorithm which will allow use to compare and estimate the precision of the constructed image, and we will show our experiments results. We will conclude in section 4.

2 State of the art

Genetic algorithms have been used to construct preimage in several papers as [6] [2] [3] [9]. There is four main steps in genetic algorithm : generation, selection, crossover, mutation.

The generation of the initial population uses two major methods : random generation of each people, or low discrepancy sequence [5]. It's currently difficult to use quasi random sequence in our context of high dimension vector (512 values). In fact, in genetic algorithm, the vector don't need to be random, but a good uniform distribution is required [7].

Concerning the selection step, we have a population of candidates, and we must select a subset of these peoples. There is three major strategies : the roulette wheel selection, the tournament selection and the rank selection [4] :

- Roulette wheel : it's a proportionate reproduction in terms of fitness score.
- Tournament selection : randomly draw 2 individuals and take the most fit with probability p. Here, we also use a scale factor : "the best individuals get a lower probability than they should, and the worst ones see their probability increased. (in the selection process) In the last iterations, the scale factor is increased to make sure that the algorithm effectively converges (SF 1 in the last iteration)." [1]
- Rank selection : take the individuals in order of their fitness score

The crossover and the mutation step is less discuss, but we find some information about simple and double crossover [1]. The crossover step allows us to mix two vectors. We cut vector in two or three pieces [1], and we exchange them between the two vectors like in 1.

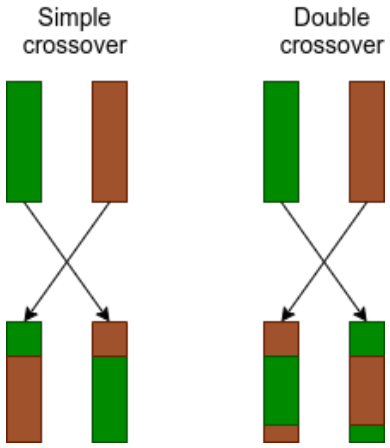


Figure 1: Simple and double crossover

There is also different methods of mutation, like single point and multi points mutation [4].

3 Results

3.1 Development

We use the FVC2002 database [8] of 800 fingerprints, 8 fingerprints of the same finger of 100 different peoples, and the biohashing transformation providing a 128 bits template.
Next we have the following algorithm to attack :

- First, we pop 2 vectors of the same people, and compute their template with 2 different seeds.
- The goal is to generate a preimage, that its template is accepted with the first template, ie the hamming distance is less than a threshold, and its template is as close as possible of the second template.

Experiments are presented with curves that represent the minimum distance distance over time (number of distance computed) between the generated template and the second legitimate template. We also have a table who give the minimum distance obtained for each threshold.

3.2 Genetic algorithm

In this section, we discuss about the results we obtain with different paramaters, and finally, we show our attack result with the better configuration. The original population is composed of n vectors which are tested preimages. This population is updated by m generations.

Crossover Their is no difference between the simple and double crossover regarding the minimal distance obtained (we use $n = m = 100$).

Probability of mutation Better results are obtained with probability of 0.2 or 0.3 (we use $n = m = 100$). It's note that high mutation probability bring us closer to randomness.

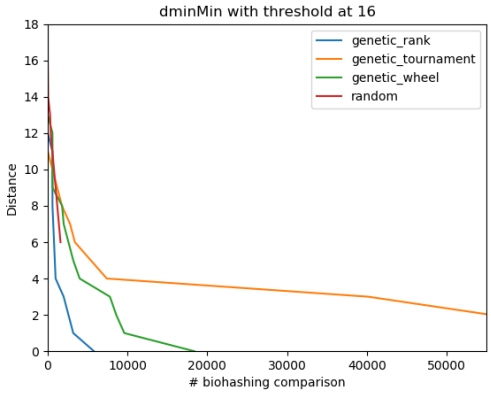
Selection method We did the following experience : we make one test for 100 peoples with population of size 200 and 500 iterations. Figure 2 shows the evolution of the minimal distance between the second genuine template and our better template candidate, where the threshold is 16 in 2a and 6 in 2b.
There are 3 curves that represent the different selection method applied to the genetic algorithm. There is also a curve that represent the efficiency of a random generation of the preimage candidate.
The curves represent the best candidate encountered according to the number of comparison (metric representing duration). Be careful that the scales are not the same on the 2 graphs.

These results are also represented in the table 1 : for all thresholds (until 16), there is the better minimal distance with the second genuine template is given. There is also the moment, representing with the number of comparison, when we found this candidate.

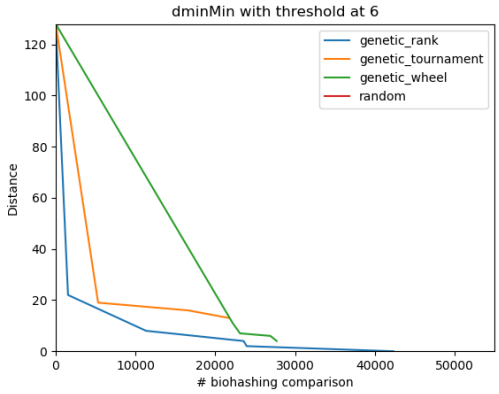
We observe that the rank selection is the better selection's method. In fact, we can see that it obtains a minimal distance of 0 from threshold at 6.
The roulette wheel selection has good score, but longer and a little worse.
The tournament selection is the worst : it's clearly longer, and it obtains worse minimal distance.
However, these 3 selection's method are really better than the random candidate's generator.

Population and iteration The size of the population and the number of iterations are decisive for the effectiveness of our program [10].

We make several experiences by changing these parameters, and we show here the results of 2 combinations. The first one has a population size of $n = 200$ and makes $m = 500$ iterations. The second one has a population size of 100 and makes 100 iterations.



(a) Threshold at 16



(b) Threshold at 6

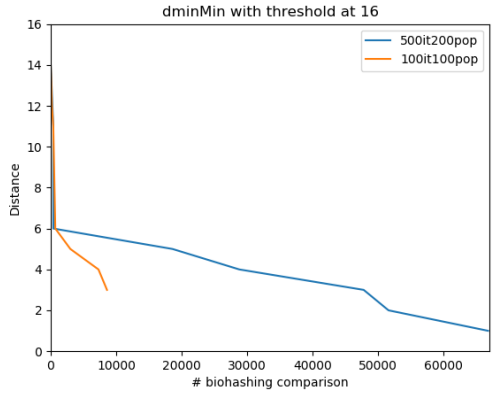
Figure 2: Selection's method

θ	Rank	Tournament	Wheel	Random
0	-	-	-	-
1	-	-	-	-
2	-	-	-	-
3	-	-	-	-
4	-	-	7 (6.9)	-
5	22 (0.15)	20 (3.3)	6 (3.9)	-
6	0 (4.2)	13 (2.1)	4 (2.7)	-
7	0 (2)	13 (2.1)	3 (3.5)	25 (3)
8	0 (1.8)	12 (4.5)	1 (5.3)	23 (1.4)
9	0 (1.6)	8 (5.3)	1 (5.3)	15 (1.3)
10	0 (1.3)	7 (6)	0 (4.8)	15 (1.3)
11	0 (0.8)	2 (6.5)	0 (3.4)	14 (0.3)
12	0 (0.6)	2 (6.5)	0 (3.2)	9 (0.6)
13	0 (0.6)	2 (6.3)	0 (2.1)	9 (0.6)
14	0 (0.6)	2 (5.9)	0 (2)	9 (0.6)
15	0 (0.6)	2 (5.6)	0 (1.8)	6 (0.1)
16	0 (0.6)	2 (5.5)	0 (1.8)	6 (0.1)

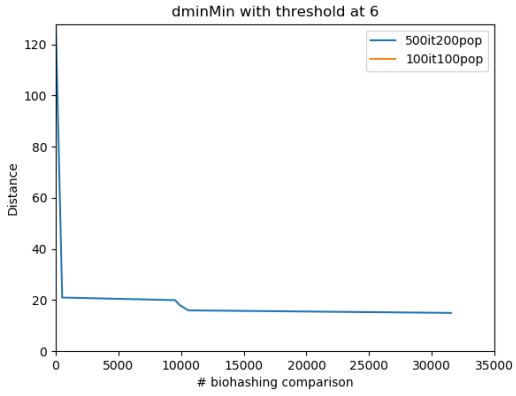
Table 1: Distance from second template in terms of selection's method (#comparison in (10^4))

As in 3.2, we give 2 graphs, the first one 3a with the threshold at 16, and the second one 3b with the threshold at 6. There are 2 curves : one represents the combination 500 iterations and 200 peoples, the other represents the combination 100 iterations and 100 peoples. Again, be careful that the scales are not the same on the 2 graphs. As we can see, the first method finally give a much better score. In fact, the second method (100 iterations 100 peoples) quickly converges in a local maximum.

We also have the table 2, which gives the minimal distance obtained with a specific threshold, and when. It also shows the best accuracy of the first combination.



(a) Threshold at 16



(b) Threshold at 6

Figure 3: Genetic population and iteration comparison

θ	500it200pop	100it100pop
0	-	-
1	-	-
2	-	-
3	-	-
4	-	-
5	17 (30326)	-
6	15 (31546)	-
7	10 (25373)	-
8	10 (25373)	-
9	3 (63592)	33 (360)
10	3 (61795)	10 (2254)
11	3 (61795)	9 (6577)
12	3 (58140)	8 (6097)
13	1 (66948)	7 (4867)
14	1 (66797)	7 (3096)
15	1 (66797)	3 (8591)
16	1 (66797)	3 (8591)
17	1 (64996)	2 (8899)
18	1 (63395)	2 (8900)
19	1 (62400)	2 (7898)
20	1 (62400)	1 (8898)
21	1 (62400)	1 (6599)
22	1 (62400)	1 (5898)
23	1 (62400)	1 (5899)
24	1 (62400)	1 (5900)
25	1 (62400)	1 (5900)
26	1 (62400)	1 (5900)

Table 2: Distance from second template in terms of population and iteration combination

3.3 Adaptative-plaintext attack algorithm

To have a better element of comparison for genetic algorithm compared to random generation, we try 3 different adaptative-plaintext attacks :

- the first one : we have a preimage candidate, we generate all its neighbors. To do that, for each vector index one by one, we create an other preimage for each possible values for this vector index with a given step. Then, we select the better neighbor according to the fitness function. And we start again with the new preimage.

- the second one : henceforth, we do the same, but for each vector index, we save the better value without changing the others. The new preimage is the vector formed by all the better saved values for each index.

- the last one : we do the same of the second method, but for each vector index, we save the better value taking into account the previous better value for each previous index. So we gradually build the new preimage.

θ	First	Second	Third
0	-	-	-
1	-	-	-
2	-	-	-
3	-	-	-
4	-	-	-
5	-	-	-
6	19 (2)	128 (1.2)	-
7	18 (2)	20 (1.2)	20 (1.4)
8	16 (2.7)	20 (1.2)	20 (1)
9	14 (1.9)	17 (1.2)	17 (1)
10	12 (1.9)	8 (1.4)	8 (1.2)
11	12 (1.9)	8 (1.4)	8 (1.2)
12	10 (2.3)	8 (1.9)	8 (1.2)
13	10 (2.3)	7 (1.9)	7 (1.2)
14	9 (2.1)	7 (1.6)	7 (1.2)
15	9 (2.1)	7 (1.6)	7 (1.2)
16	9 (2.1)	7 (1.6)	7 (1.2)

Table 3: Comparison of the 3 adaptative-plaintext attack (#comparison in (10^6))

We make an experience to find out which method is the best. To do that, we make a test and start an attack with each method. We can see on 3 that the second and the third method has a better minimal score, and faster, than the first one. Comparing to the genetic algorithm results in 1, it's obvious that genetic algorithms are much faster and much accurate than adaptative-plaintext attack algorithm.

4 Conclusion and discussion

Genetic algorithms allow us to design attack against biometric transformation. For example, experiments have showed that a mutation probability at 0.2, a rank selection's method, and a population size of 200 (with 500 iterations) give the best long-lived nearby-template preimage.

Perspectives This may be an idea to improve the enrollment step, by compute an average vector instead of taking the most accurate vector of the enrollment.
This work should be generalized to generic transformations and several biometric modalities.

References

[1] Andre, J., Siarry, P., Dognon, T.: An improvement of the standard genetic algorithm Ⓡghting premature con-

vergence in continuous optimization. Advances in Engineering Software p. 12 (2001)

[2] Dong, X., Jin, Z., Jin, A.T.B.: A Genetic Algorithm Enabled Similarity-Based Attack on Cancellable Biometrics. arXiv:1905.03021 [cs] (May 2019), arXiv: 1905.03021

[3] Galbally, J., Ross, A., Gomez-Barrero, M., Fierrez, J., Ortega-Garcia, J.: Iris image reconstruction from binary templates: An efficient probabilistic approach based on genetic algorithms. Computer Vision and Image Understanding **117**(10), 1512–1525 (Oct 2013)

[4] Hsu, W.H.: Introduction to Genetic Algorithms. CIS 732: Machine Learning and Pattern Recognition p. 18 (2002)

[5] Kocis, L., Whiten, W.J.: Computational investigations of low-discrepancy sequences. ACM Transactions on Mathematical Software **23**(2), 266–294 (Jun 1997)

[6] Lacharme, P., Cherrier, E., Rosenberger, C.: Preimage Attack on BioHashing. In: International Conference on Security and Cryptography (SECRYPT). p. 8 p. (2013)

[7] Maaranen, H., Miettinen, K., Mäkelä, M.: Quasi-random initial population for genetic algorithms. Computers & Mathematics with Applications **47**(12), 1885–1895 (Jun 2004)

[8] Maio, D., Maltoni, D., Cappelli, R., Wayman, J., Jain, A.: FVC2002: Second Fingerprint Verification Competition. In: Object recognition supported by user interaction for service robots. vol. 3, pp. 811–814. IEEE Comput. Soc (2002)

[9] Rozsa, A., Glock, A.E., Boult, T.E.: Genetic algorithm attack on minutiae-based fingerprint authentication and protected template fingerprint systems. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pp. 100–108. IEEE (Jun 2015)

[10] Stanhope, S.A., Daida, J.M.: Optimal mutation and crossover rates for a genetic algorithm operating in a dynamic environment. In: Evolutionary Programming VII, vol. 1447, pp. 693–702. Springer Berlin Heidelberg (1998)

[11] Teoh, A.B., Ngo, D.C., Goh, A.: Personalised cryptographic key generation based on FaceHashing. Computers & Security **23**(7), 606–614 (Oct 2004)