Shattering the Glass Maze *

Jens Hermans, Roel Peeters, Bart Mennink
KU Leuven, ESAT/COSIC and iMinds
firstname.lastname@esat.kuleuven.be

Abstract: Template protection plays a crucial role in protecting the privacy of biometric data, by providing irreversibility and unlinkability. The Glass Maze, as presented by Trugenberger at BIOSIG 2011, is a fingerprint key binding mechanism that is claimed to provide template protection. With the correct fingerprint, the key that is entangled with the fingerprint data can be retrieved. The template protection of the Glass Maze is based on the convergence properties of a Hopfield model, a neural network. We however show how to *revert* the Glass Maze to recover the key, without requiring a correct fingerprint. This completely breaks the irreversibility property, and hence also unlinkability.

1 Introduction

Biometrics are an interesting alternative for traditional identification mechanisms, such as passwords and cryptographic tokens. With the increased adoption and consideration of biometrics, privacy concerns have been raised since biometrics provide a unique identifier for an individual. Biometric data stored in different systems might allow linking of individuals across applications. Moreover, biometric data could be abused for spoofing using an artificial sample. Biometric template protection provides a solution to these privacy issues. The key requirements for a biometric template protection mechanism are irreversibility and unlinkability [ISO11]. The former implies that it is not possible to recover the original biometric data from the protected template. The latter ensures that multiple samples from the same characteristic cannot be linked accross applications after applying template protection. There are several attack models for template protection schemes, the most common assuming a full leakage of the protected template data. We refer to Simoens *et al.* [SYZ+12] for more information on the security and privacy requirements for biometric template protection schemes.

Template protection can roughly be categorized [JNN08] in four main types of techniques: salting, non-invertible functions ("cancelable biometrics"), key binding and key generation mechanisms. The first two techniques require a separate key that needs to be presented in order to use the protected template for matching. The last two will produce a key upon presenting a matching sample. Key-binding mechanisms directly bind a separately generated cryptographic key to a biometric template, while key-generating mechanisms on the other hand directly generate a key from the template.

*Fragile, handle with care.

There are various fingerprint key binding schemes [NNJ08, CKL03, NJP07, LYC+10], of which several are based on the Fuzzy Vault scheme [JS02, JS06]) or Fuzzy commitment [JW99]. This paper discusses the template protection properties of the Glass Maze, which unlike the previous is not based on Fuzzy Commitment or the Fuzzy Vault, but on a completely new concept originating from research on neural networks.

The Glass Maze

Trugenberger [Tru11, Tru12] proposed the Glass Maze as a fingerprint key-binding mechanism. The term Glass Maze stems from the spin glasses, networks of spins (binary values with symmetric interaction) that underly the mechanism. The protection mechanism is based on the convergence (or divergence) of a neural network towards either one of the stored states (if a close enough fingerprint is provided) or random behavior (if there is no match with one of the stored fingerprints). [Tru11, Tru12] claims that the Glass Maze is *robust* against both brute force and cross matching attacks and that it can be used for identification purposes. Since no "naked" template is ever stored, the privacy issue is claimed to be solved. A key can be stored in the Glass Maze by flipping random bits of the fingerprint.

It is important to note that [Tru11, Tru12] foresaw two usage scenarios cases for the Glass Maze. In the first case only a single fingerprint/key is stored per neural network, while in the second multiple fingerprints/keys are stored. The remaining storage space in the first case is filled with random data. The second scenario has the advantage of requiring less storage per fingerprint, as the storage expansion factor in the first can be quite high. More details on the Glass Maze are given in Sect. 2.

Springer [Spr13] already pointed out potential security issues with the Glass Maze by noting a high false acceptance rate in experiments for the single fingerprint/key scenario. Essentially, when storing a single fingerprint in the Glass Maze and setting the other states to random vectors, the Glass Maze tends to evolve easily to the stored fingerprint when presenting a random fingerprint. The random vectors are uniformly random whereas fingerprints are not random at all. The multiple fingerprint scenario was not analyzed.

Our contributions

We present two distinct attacks on the Glass Maze, one for every usage scenario. We attack the template protection properties and hence assume that the database is leaked, i.e. we have full access to the protected template. The first attack (Sect. 3) targets the case that only a single fingerprint is stored in a neural network. By applying a simulated annealing based attack, similar to [Pas13], we can easily recover the original data stored in the Glass Maze in the majority of cases. Essentially, this attack bypasses the potentially chaotic behavior of the neural network. In simulated annealing we directly evaluate the energy function for the proposed solution and apply random modifications to the state. The second attack (Sect. 4) targets the case where multiple fingerprints are stored in the same neural network. A simulated annealing based attack does not work in this case, but the skewed distribution of the input data can now be abused. Again, we can recover all original

biometric data. In Sect. 5 we present the results of the experiments that were performed with the simulated annealing based attack, demonstrating that our first attack is practical. As the second attack is deterministic, no extensive experimentation was required. We also give some insight in some other fundamental issues with the Glass Maze that became apparent during experimentation.

2 Preliminaries - the Glass Maze

The Glass Maze is based on a Hopfield model [Hop82], a neural network. The Hopfield model has N neurons, each of which can have a state $s_i \in \{-1, 1\}$. Neurons are connected with synapsis that have a weight $w_{i,j} = w_{j,i}$ (and $w_{i,i} = 0$). The state of the neural network is updated from time t to $t + 1$ as follows:

$$s_i(t+1) = sign(h_i(t)) \quad (1)$$

$$h_i(t) = \sum w_{i,j} s_j(t) \quad (2)$$

The weights $w_{i,j}$ are defined using the Hebb rule:

$$w_{i,j} = \sum_{\mu=1}^{p} x_i^\mu x_j^\mu \,, \quad (3)$$

where $x_i^\mu \in \{-1, 1\}$. Define $\mathbf{x}^\mu = [x_1^\mu \dots x_N^\mu]^T$, for $\mu \in [1, p]$. The $\mathbf{x}^\mu$ are patterns that need to be 'memorized' by the neural network. The neural network is thus uniquely represented by the values $w_{i,j}$.

The idea behind using a Hopfield model to protect fingerprint templates is that it is not possible to recover the original template from the representation of the neural network (i.e. $w_{i,j}$). However, when given a matching fingerprint, this can be used as initial state for the neural network. Assuming that the original fingerprint is close to the template stored in the neural network, it will be in the basin of attraction of the original template and hence the neural network will converge to this original state.

The behaviour of the neural network can be characterized by the loading factor $\alpha = p/N$. Depending on its value, the network will converge to one of the states $\mathbf{x}^\mu$ or exhibit chaotic behaviour. In [Tru11, Tru12] a loading factor of $\alpha \simeq 0.1$ is proposed to ensure convergence to one of the states $\mathbf{x}^\mu$.

The Glass Maze goes one step further by encoding cryptographic keys in the original state. This is done by flipping certain bits of the fingerprint template and storing the resulting vector in the neural network. [Tru11, Tru12] remains vague on the exact procedure for this, but argues that if the changes are sufficiently small, a matching fingerprint will still be in the basin of attraction of the original pattern. Hence, the original pattern and the key could be recovered. [1] In the remainder of this paper we will refer to the resulting vector(s), containing the fingerprint(s) with certain bits flipped, as the *key* vector(s).

[1] The original paper remains vague regarding the number of bits k that can be changed, but simply mentions

In this paper we will make abstraction of the behavior of the neural network, as it is not required for our attacks. We show how to directly recover the keys (i.e. $\mathbf{x}^\mu$) from the weights $w_{i,j}$.

3 Attack - Single fingerprint Glass Maze

3.1 Analysis

As suggested by [Tru11, Tru12], we set $p = 25$ and $N = 256$ to obtain a load factor $\alpha \simeq 0.1$. This will thus allow for storing p key vectors $\mathbf{x}^\mu \in \{-1, 1\}^N$. Fingerprints are rasterized into a 16×16 matrix, where a value $x_i^\mu = -1$ represents the presence of one or more minutia in the pixel, and $x_i^\mu = 1$ the absence. To simplify notation, we will omit the range of indices when the index μ is in the range $[1, p]$ and when i or j are in the range $[1, N]$.

The weights of the neural network are computed as

$$w_{i,j} = \sum_\mu x_i^\mu x_j^\mu \tag{4}$$

which in matrix notation becomes $\mathbf{W} = \mathbf{X}\mathbf{X}^T$, with $\mathbf{X}$ an $n \times p$ matrix.

Initially, [Tru11, Tru12] suggests storing only a single fingerprint/key in a network. In this case $\mathbf{x}^1$ contains the key. On average a fingerprint $\mathbf{x}^1$ contains about $\beta = 40$ entries that are -1. The other $\mathbf{x}^\mu$, with $\mu > 1$, are filled with uniformly random data, i.e. $\Pr[\mathbf{x}_i^\mu = -1] = \frac{1}{2}$.

The attack we will present breaks the template protection scheme, i.e. when given $\mathbf{W}$ it returns the key vectors $\mathbf{x}^\mu$, in this case $\mathbf{x}^1$. Before we present the attack algorithm we have a closer look at how the Glass Maze functions. Let's assume we have a candidate vector $\mathbf{x} = \mathbf{x}^1$. Clearly $\mathbf{X}^T\mathbf{x}^1 \approx [N\,0\,0\,\ldots\,0]^T$, since $\mathbf{x}^{1,T}\mathbf{x} = N$ and $E(\mathbf{x}^{i,T}\mathbf{x}) = 0$ (for $i \neq 1$) since $\mathbf{x}^{i,T}$ is uniformly random. Multiplying the resulting vector with $\mathbf{X}$ gives

$$\mathbf{W}\mathbf{x}^1 = \mathbf{X}\mathbf{X}^T\mathbf{x}^1 \approx \mathbf{X}[N\,0\,0\,\ldots\,0]^T \approx N\mathbf{x}^1 \qquad . \tag{5}$$

Simply put, we can detect whether a candidate vector $\mathbf{x}$ is close to the target $\mathbf{x}^1$, by simply computing $\mathbf{W}\mathbf{x}$ and verifying if the result is similar to $\mathbf{x}$. Because $\mathbf{x}^1$ is so different from the other vectors $\mathbf{x}^\mu$ it is unlikely that a candidate vector $\mathbf{x}$ will match one of these $\mathbf{x}^\mu$ ($\mu \neq 1$).

In the algorithm we will use a matching (scoring) function

$$\eta(\mathbf{W}, \mathbf{x}) = -\frac{1}{N^2}\mathbf{x}^T\mathbf{W}\mathbf{x} = -\frac{1}{N^2}\sum_i \mathbf{x}_i(\mathbf{W}\mathbf{x})_i \qquad . \tag{6}$$

that the original fingerprint must still be in the basin of attraction of the modified fingerprint. Otherwise the number of flipped bits must be lowered. We have doubts whether k can be anywhere near a typical cryptographic key length, i.e. at least 80 or 128 bits. In the remainder of this paper we assume that the modified vector still has properties comparable to a fingerprint, i.e. the amount of -1 entries remains in the typical fingerprint range.

The additional multiplication with $\mathbf{x}^T$ flips the sign of the corresponding values of $\mathbf{Wx}$, before adding everything up into a final score. The entire score is normalized by division with N^2. The proposed matching function is, up to scaling, identical to the original energy function for the spin glasses [Tru11].

3.2 Algorithm

We now simply embed the matching function $\eta(\cdot)$ into the simulated annealing framework shown in Algorithm 1. In simulated annealing random modifications to a candidate solution are considered in every iteration (line 5). Based on a matching function the 'energy' E' of the modified candidate solution $\mathbf{x}'$ is computed (line 6). The energy level E' of the modified candidate solution is compared with the energy level E of the original candidate solution, taking into account the 'temperature' T. Based on this energy and some randomness, the modification is either accepted (i.e. the state is updated) or rejected (line 7). The decision is thus not solely based on the matching function, which avoids getting stuck in local minima. Throughout every iteration of the algorithm the temperature is gradually lowered, which also lowers the probability of accepting a candidate solution with a worse energy level.

Algorithm 1 Simulated annealing algorithm

Require: $\mathbf{W}$
1: $\mathbf{x} =$ random vector with weight β
2: $\mathbf{x}_{best} = \mathbf{x}$
3: **for** $k = 1 \dots k_{max}$ **do**
4: $\quad T = k/k_{max}$
5: $\quad \mathbf{x}' = newvector(\mathbf{x})$
6: $\quad E' = \eta(\mathbf{W}, \mathbf{x})$
7: $\quad$ **if** $e^{(E-E')/T} > 1 - rand()$ **then**
8: $\quad\quad \mathbf{x} = \mathbf{x}'$
9: $\quad\quad E = E'$
10: $\quad$ **end if**
11: $\quad$ **if** $E' < E_{best}$ **then**
12: $\quad\quad \mathbf{x}_{best} = \mathbf{x}$
13: $\quad\quad E_{best} = E'$
14: $\quad$ **end if**
15: **end for**
16: **return** $\mathbf{x}_{best}$

The final component to consider is the way in which new candidate solutions are generated. Algorithm 2 shows the $newvector$ subroutine. In order to avoid converging to one of the other vectors $\mathbf{x}^i$ ($i \neq 1$), we need to ensure that the number of -1 entries remains around the expected value β. A simple way to achieve this is by introducing a lower and upper bound β_{low} and β_{high}. In case the lower bound is reached, a random entry of $\mathbf{x}$ is set to

-1, which is likely to increase the number of -1's again. If the upper bound is reached, an entry is set to 1 and otherwise a random entry is flipped sign.

Algorithm 2 Subroutine $newvector(\mathbf{x})$

Require: $\mathbf{x}$
1: $\mathbf{x}' = \mathbf{x}$
2: $i = randindex()$
3: **if** $\#\{-1 \text{ in } \mathbf{x}\} < \beta_{low}$ **then**
4: $\quad \mathbf{x}'[i] = -1$
5: **else if** $\#\{-1 \text{ in } \mathbf{x}\} > \beta_{high}$ **then**
6: $\quad \mathbf{x}'[i] = 1$
7: **else**
8: $\quad \mathbf{x}'[i] = -\mathbf{x}'[i]$
9: **end if**
10: **return** $\mathbf{x}'$

We performed several experiments with the proposed algorithm, the detailed results of which are presented in Section 5. The vast majority of the experiments resulted in a full recovery of the original key vector $\mathbf{x}^1$, although some issues remain when convergence to an alternative solution was observed that had a better energy level than the actual solution.

4 Attack - Multiple fingerprint Glass Maze

4.1 Analysis

A second instance is considered in [Tru11, Tru12], where p fingerprints are stored in the network instead of a single fingerprint. Clearly, it becomes difficult to apply the simulated annealing based attack, since convergence will be hindered as a candidate vector might match any of the p stored fingerprint vectors. We can however exploit a different property now since, overall, $\mathbf{X}$ will be very biased towards 1 entries instead of -1 entries. Instead of trying to recover the fingerprint vectors $\mathbf{x}^\mu$ one by one, we recover *pixel* vectors $\mathbf{x}_i$. A pixel vector $\mathbf{x}_i$ consists of the values of the same pixel location across all p fingerprints.

With probability $(1 - \beta/N)^p \approx 0.014$ it holds that $\mathbf{x}_i = [1\,1\,1\ldots1]$. With probability ~ 0.97 there will be two or more such pixel vectors. Two such vectors result in an entry $w_{i,j} = p$, allowing them to be detected easily. Given one such vector, we can now determine the Hamming weight (or equivalently the sum) of all other pixel vectors. Since the matrix $\mathbf{W}$ is invariant under permutations of fingerprints, we can start assigning the pixel vectors that have only a single -1 entry and, after this, compare these vectors with the others to determine the remaining entries.

4.2 Notation

We will first convert the problem to a binary form, which is characterized by pixel vectors $\mathbf{b}_i \in \{0,1\}^p$ and the Hamming distances $\delta_{i,j} = |\mathbf{b}_i, \mathbf{b}_j|_H$. This simplifies notation and the description of the algorithm.

Define

$$b_i^\mu = \frac{1}{2}(x_i^\mu + 1) \tag{7}$$

(i.e. $b_i^\mu = 0$ if $x_i^\mu = -1$ and $b_i^\mu = 1$ if $x_i^\mu = 1$). In the following we will consider vectors $\mathbf{b}_i = [b_i^1\, b_i^2 \cdots b_i^\mu]$. The elements of this vector represent the same pixel location in different fingerprints.

Insert this into Equation (4) we obtain

$$w_{i,j} = \sum_\mu (2b_i^\mu - 1)(2b_j^\mu - 1) \tag{8}$$

$$= \sum_\mu \left[4b_i^\mu b_j^\mu - 2(b_i^\mu + b_j^\mu)\right] + p \tag{9}$$

We define $\delta_{i,j} = \frac{p - w_{i,j}}{2}$, i.e.

$$\delta_{i,j} = \sum_\mu b_i^\mu + \sum_\mu b_j^\mu - 2\sum_\mu b_i^\mu b_j^\mu = |\mathbf{b}_i, \mathbf{b}_j|_H \tag{10}$$

The values $\delta_{i,j}$ remain unchanged under permutations of the key vectors $\mathbf{x}^\mu$.

4.3 Algorithm

As stated before, there will be several vectors $\mathbf{b}_i = [1\,1\ldots1]$ which can be easily detected, since $\mathbf{b}_i = \mathbf{b}_j$ implies $\delta_{i,j} = 0$.

Stage 1 The algorithm starts by looking for an entry $\delta_{i',j'} = 0$ (with $i' \neq j'$) and guessing that $\mathbf{b}_{i'} = \mathbf{b}_{j'} = [1\,1\ldots1]$. (The whole procedure can be repeated with a different $\delta_{i',j'}$ if this turns out not to be the case).

Let $\mathcal{I}$ denote the set of indices i such that $\delta_{i,i'} = 0$.

Stage 2 Now, since $\mathbf{b}_{i'} = [1\,1\cdots1]$, we can easily obtain the Hamming weight of every vector $\mathbf{b}_j$ from Equation (10) as

$$\delta_{i',j} = |\mathbf{b}_{i'}, \mathbf{b}_j|_H = p - |\mathbf{b}_j|_H \quad . \tag{11}$$

Let $\mathcal{J}$ denote the set of indices j such that $|\mathbf{b}_j|_H = p - 1$ (i.e. vectors with a single zero). Since $\delta_{i,j}$ does not change under permutations of the fingerprints, we can freely choose the

order of the fingerprints. For every $j \in \mathcal{J}$ this implies we can choose the position of that zero element inside the vector for the first p vectors. The only exception occurs when for both $i, j \in \mathcal{J}$ it holds that $\delta_{i,j} = 0$, which implies the zero should be at the same position in both vectors.

We thus assign a vector of the series $[0111\ldots1], [1011\ldots1], [1101\ldots1]\ldots$ to every $\mathbf{b}_j$ for $j \in \mathcal{J}$, taking into account that the same vector must be assigned when $\delta_{i,j} = 0$ and a different one otherwise. We let $\tilde{\#}\mathcal{J}$ denote the number of unique vectors $\mathbf{b}_j$ (for $j \in \mathcal{J}$), to take into account the duplicates that occur.

Stage 3 We now iterate over all remaining vectors $\mathbf{b}_i$ with $i \notin \mathcal{I} \cup \mathcal{J}$.

For every $\tilde{j} \in \mathcal{J}$ assume that $b_{\tilde{j}}^{\tilde{\mu}} = 0$ (i.e. that $\tilde{\mu}$ is the index of the zero element in $\mathbf{b}_{\tilde{j}}$). Hence from Equation (10) we get

$$\begin{aligned}
\delta_{i,\tilde{j}} &= |\mathbf{b}_i, \mathbf{b}_{\tilde{j}}|_H && (12)\\
&= \sum_{\mu} b_i^{\mu} \oplus b_{\tilde{j}}^{\mu} && (13)\\
&= b_i^{\tilde{\mu}} + \sum_{\mu \neq \tilde{\mu}} 1 \oplus b_i^{\mu} && (14)\\
&= b_i^{\tilde{\mu}} + \sum_{\mu} 1 \oplus b_i^{\mu} - (1 \oplus b_i^{\tilde{\mu}}) = 2b_i^{\tilde{\mu}} + p - |\mathbf{b}_i|_H - 1 && (15)
\end{aligned}$$

and

$$b_i^{\tilde{\mu}} = \frac{|\mathbf{b}_i|_H + \delta_{i,\tilde{j}} - (p-1)}{2} \tag{16}$$

This way, we can reconstruct the first $\tilde{\#}\mathcal{J}$ entries of every vector $\mathbf{b}_i$ and have thus already reconstructed $\#\mathcal{J}$ out of p fingerprints.

Stage 4 The remaining $p - \tilde{\#}J$ entries can be recovered by recursing the algorithm. Starting again from Equation (10), we divide this over the known entries (the first $\tilde{\#}J$ bits of every pixel vector) and the unknown ones

$$\begin{aligned}
\delta_{i,j} &= \sum_{\mu} (b_i^{\mu} + b_j^{\mu} - 2b_i^{\mu} b_j^{\mu}) && (17)\\
&= \sum_{\mu=1}^{\tilde{\#}J} (b_i^{\mu} + b_j^{\mu} - 2b_i^{\mu} b_j^{\mu}) + \sum_{\mu=1+\tilde{\#}J}^{p} (b_i^{\mu} + b_j^{\mu} - 2b_i^{\mu} b_j^{\mu}) && (18)\\
&= \delta'_{i,j} + \delta''_{i,j} && (19)
\end{aligned}$$

where $\delta'_{i,j}$ is the part originating from the (known) pixels $[1\ldots\tilde{\#}J]$ and $\delta''_{i,j}$ is the part from the (unknown) pixels $[\tilde{\#}J + 1\ldots p]$.

Starting from $\delta''_{i,j}$, we can execute the above algorithm again to search for the vectors $\mathbf{b}''_i \in \{0,1\}^{(p-\#J)}$ such that

$$\delta''_{i,j} = |\mathbf{b}''_i, \mathbf{b}''_j|_H \tag{20}$$

5 Experiments

In this Section we perform extensive experiments with the simulated annealing algorithm for a single fingerprint Glass Maze. Since our second attack (Sect. 4) is deterministic, no experimentation is required as the attack always succeeds.

5.1 Setup

In all experiments, the first vector $\mathbf{x}^1$ was set to a random vector with $\Pr[\mathbf{x}^1_i = 1] = 40/256$. Moreover we only retained vectors containing between 35 and 45 1-entries. All the other vectors $\mathbf{x}^\mu$ were set to uniformly random data ($\Pr[\mathbf{x}^\mu_i = 1] = 1/2$). This manner of generating synthetic fingerprints does not impact our experiments negatively. Real fingerprint data has an even more skewed distribution when comparing it with the uniformly random vectors $\mathbf{x}^\mu$. Since the attack exploits the different distributions of $\mathbf{x}^1$ and the other $\mathbf{x}^\mu$, the attack will improve when using real fingerprint data, i.e. if we can recover the artificial data in our experiments, we can definitely recover real fingerprint data.

The resulting matrix $\mathbf{W}$ was used as input to the simulated annealing algorithm. During the simulated annealing algorithm we continuously compared the best vector with the correct vector $\mathbf{x}^1$, in order to detect the iteration in which the correct solution was found. In order to avoid long computations the algorithm was terminated upon finding the correct vector, although theoretically a solution with a *better* match than the original vector could be found in subsequent iterations. The algorithm was also terminated if, after $10\,000$ iterations, the correct solution was not found.

We performed the above experiment $10\,000$ times on a machine with an Intel Xeon X7350 CPU running at 2.93GHz, of which only a single core was used.

5.2 Results

Figure 1 shows a cumulative histogram of the number of iterations required to find the correct solution. In 84.4% of experiments the correct solution was obtained within the limit of $10\,000$ iterations. The vast majority does so with less than $3\,000$ iterations which takes about 1 second. From this data, we can already conclude that the Glass Maze is critically broken, as such a high percentage is unacceptable for a template protection scheme. Nevertheless we take a closer look to the remaining 15.6% of experiments where the correct

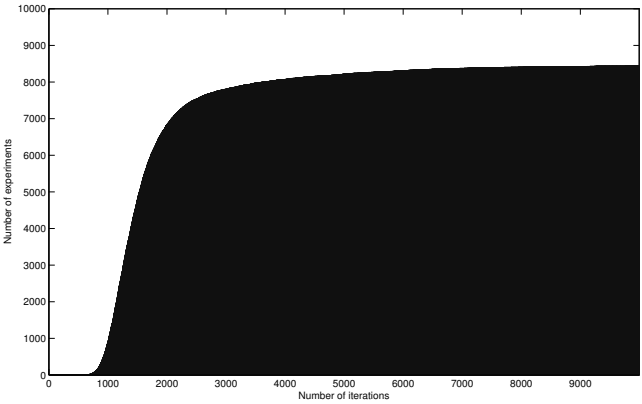

Figure 1: Cumulative histogram of the number of iterations of simulated annealing required to find the correct solution.

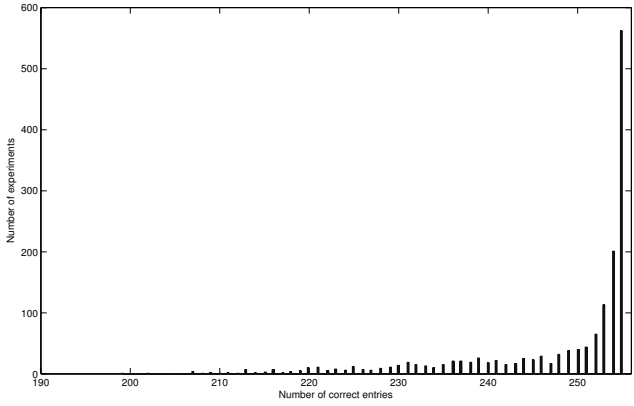

Figure 2: Histogram of the number of correct entries in the best solution obtained from simulated annealing.

solution was not found within $10\,000$ iterations. Figure 2 shows a histogram of the number of correct entries for these experiments. By measuring the number of correct entries (instead of correct minutiae), we take into account both actual minutiae that were not present in the solution as well as pixels that were incorrectly identified as minutiae in the solution. For most of the experiments, the obtained solution from simulated annealing is very close to the correct solution.

Of the *failed* experiments 1035 however obtain a solution with a lower energy than the correct solution, i.e. it finds a *better* solution than the correct one. These results hint at far more fundamental problems of the Glass Maze: by inserting random vectors $\mathbf{x}^\mu$, it is not guaranteed that the key vector is indeed a (local) minimum of the Glass Maze energy function. Likely a very close vector will become the minimum due to the noise introduced by the random vectors. Clearly, using the resulting vector as a cryptographic key becomes impossible when even the slightest variation is applied to the correct key.

In total only 527 (5.2%) experiments result in a suboptimal solution. Increasing the number of iterations or simply repeating the experiment with the same input can already resolve this, as a different descent route is taken at every execution. Even so, the fraction of experiments that fails remains low whereas the number of successes is unacceptably high.

Regarding performance, the algorithm turns out to be very efficient: the memory usage is negligible and the computational effort is about 0.3 seconds for 1000 iterations of simulated annealing.

6 Conclusion and Future Work

We presented two attacks, one for every use case, against the irreversibility property of the Glass Maze. In both cases we completely bypass the behavior of the neural network, by respectively running simulated annealing or a simple deterministic algorithm. The attacks result in the recovery of the key, without requiring a matching fingerprint to be available. We applied the attacks to random data that has the same probability density as fingerprints (i.e. about 40 pixels containing minutiae). From our experiments with the simulated annealing algorithm it is clear that the vast majority of networks can be reversed. Moreover, our experiments reveal fundamental issues with the neural networks that hinder convergence to the correct solution and hence affect the correctness of the resulting key even when the correct fingerprint is presented.

Directions for future work

The obvious endeavor would be finding a fix for the above attacks. We however failed to identify any potential correction to the Glass Maze that might resolve these issues. Fundamentally, a neural network seems to be unfit for creating template protection schemes, as the whole mechanism can easily be bypassed by using other algorithms that directly use the protected template data without evaluating the neural network.

Another direction would be to improve the attacks and the experimentation. At the moment no specific properties of fingerprints are taken into account, neither in the attack algorithms, nor in the experiments, since we used random data. Springer [Spr13] already hinted at potential security issues, since the minutiae are unevenly distributed throughout the image. For example, pixels near the edge of the image are less likely to contain minutiae. As the distribution of fingerprints is even more skewed than the random data we used, we expect that the current attacks will show even better results. Improving the algorithms by taking into account knowledge of this distribution will likely result in an additional improvement.

Acknowledgements

This work was supported by the European Commission through the FIDELITY EU-FP7 project (Grant No. SEC-2011-284862) and the Research Council KU Leuven: GOA TENSE (GOA/11/007).

References

[BB13] Arslan Brömme and Christoph Busch, editors. *2013 BIOSIG - Proceedings of the 12th International Conference of Biometrics Special Interest Group, Darmstadt, Germany, September 4-6, 2013*, volume 212 of *LNI*. GI, 2013.

[CKL03] T. Charles Clancy, Negar Kiyavash, and Dennis J. Lin. Secure Smartcardbased Fingerprint Authentication. In *Proceedings of the 2003 ACM SIGMM Workshop on Biometrics Methods and Applications*, WBMA '03, pages 45–52, New York, NY, USA, 2003. ACM.

[Hop82] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the USA*, 79(8):2554–2558, 1982.

[ISO11] ISO. ISO/IEC 24745:2011 - Information technology – Security techniques – Biometric information protection, 2011.

[JNN08] Anil K. Jain, Karthik Nandakumar, and Abhishek Nagar. Biometric Template Security. *EURASIP J. Adv. Sig. Proc.*, 2008, 2008.

[JS02] Ari Juels and Madhu Sudan. A Fuzzy Vault Scheme. *IEEE Int. Symp. Information Theory*, page 408, 2002.

[JS06] Ari Juels and Madhu Sudan. A Fuzzy Vault Scheme. *Des. Codes Cryptography*, 38(2):237–257, 2006.

[JW99] Ari Juels and Martin Wattenberg. A Fuzzy Commitment Scheme. In Juzar Motiwalla and Gene Tsudik, editors, *ACM Conference on Computer and Communications Security*, pages 28–36. ACM, 1999.

[LYC+10] Peng Li, Xin Yang, Kai Cao, Xunqiang Tao, Ruifang Wang, and Jie Tian. An alignment-free fingerprint cryptosystem based on fuzzy vault scheme. *J. Network and Computer Applications*, 33(3):207–220, 2010.

[NJP07] Karthik Nandakumar, Anil K. Jain, and Sharath Pankanti. Fingerprint-Based Fuzzy Vault: Implementation and Performance. *IEEE Transactions on Information Forensics and Security*, 2(4):744–757, 2007.

[NNJ08] Abhishek Nagar, Karthik Nandakumar, and Anil K. Jain. Securing fingerprint template: Fuzzy vault with minutiae descriptors. In *ICPR*, pages 1–4. IEEE, 2008.

[Pas13] Andreas Pashalidis. Simulated annealing attack on certain fingerprint authentication systems. In Brömme and Busch [BB13], pages 63–74.

[Spr13] Markus Springer. Protection of Fingerprint Data with the Glass Maze Algorithm. In Brömme and Busch [BB13], pages 249–256.

[SYZ+12] Koen Simoens, Bian Yang, Xuebing Zhou, Filipe Beato, Christoph Busch, Elaine M. Newton, and Bart Preneel. Criteria towards metrics for benchmarking template protection algorithms. In Anil K. Jain, Arun Ross, Salil Prabhakar, and Jaihie Kim, editors, *ICB*, pages 498–505. IEEE, 2012.

[Tru11] Carlo A. Trugenberger. The Glass Maze: Hiding Keys in Spin Glasses. In Arslan Brömme and Christoph Busch, editors, *BIOSIG*, volume 191 of *LNI*, pages 89–102. GI, 2011.

[Tru12] Carlo A. Trugenberger. Hiding Identities in Spin Glasses. In George A. Tsihrintzis, Jeng-Shyang Pan, Hsiang-Cheh Huang, Maria Virvou, and Lakhmi C. Jain, editors, *IIH-MSP*, pages 35–40. IEEE, 2012.