Supplementary Materials for "IronMask: Modular Architecture for Protecting Deep Face Template"

Sunpill Kim[1] Yunseong Jeong[2] Jinsu Kim[3] Jungkon Kim[3] Hyung Tae Lee[2] Jae Hong Seo[1*]

[1]Department of Mathematics & Research Institute for Natural Sciences, Hanyang University
[2]Division of Computer Science and Engineering, College of Engineering, Jeonbuk National University
[3]Security Team, Samsung Research, Samsung Electronics

A. Template Attack against ArcFace

A template attack is one of the severest attacks against biometric recognition systems [2, 4, 3, 1, 5]. In particular, attacks using deep neural network successfully recover an analog of the original facial image from its template of target recognition systems. The neighborly de-convolutional neural network (NbNet) is a deep neural network based template attack that successfully shows vulnerability of a deep face recognition, Facenet in [5]. It achieves over 95.2% TAR at 0.1% FAR on the LFW dataset. The original NbNet was designed for the neural network architecture of Facenet using the triplet-loss function. We adjust and adapt it to the neural network architecture of ArcFace using additive angular margin loss and obtain a superior result than that applied to Facenet in [5]. This result implies that templates of ArcFace are more exposed to threats than those of FaceNet.

The NbNet consists of 6 neighborly de-convolution blocks (NbBlock), similar to Densenet, and 1 convolution operation. It uses two different loss functions. One of them is a pixel-wise loss function and the other is the L2-norm of difference between the outputs of feature mapping function. They empirically determined the second activation function of the third layer as the mapping function.

In order to adapt the NbNet to ArcFace, we reduce the size of one NbBlock and change the perceptual loss to the angular distance between the outputs of ArcFace model. Since the Arface model uses a small variation of margin loss for intra-class and a large variation of margin loss for inter-class, ArcFace uses larger threshold than Facenet. Under this setting, we experiment the vulnerability of ArcFace against template attack and obtain the results in Table 3. According to our experimental results, at FAR of 0.1%, TAR of templates generated by the NbNet for ArcFace is about 2.5% higher than that for Facenet. It is evident that the more advanced face recognition may cause severer threats to user privacy.

FAR	0.0001%	0.1%	1.0%
Facenet	NA	95.20%	98.63%
ArcFace	81.5%	97.74%	NA

Table 3: TARs for attacks against Facenet and ArcFact at three different FARs.

B. Theoretical Complements for ECC

In this section, we provide proofs of propositions that were missing in the main body of this paper due to the space constraints.

Proposition 1. *Let $\mathcal{C}_\alpha$ be the set of codewords defined in Definition 1.*

1. *The minimum angular distance between any two distinct codewords in $\mathcal{C}_\alpha$ is $\cos^{-1}(1-\frac{1}{\alpha})$.*
2. *The output of the* USample *algorithm is uniformly distributed over $\mathcal{C}_\alpha$.*
3. *For any $\mathbf{u} \in S^{n-1}$, the output $\mathbf{c}$ of the* Decode *algorithm satisfies the following inequality:*
$$\langle \mathbf{c}, \mathbf{u} \rangle \geq \langle \mathbf{c}', \mathbf{u} \rangle \text{ for all } \mathbf{c}' \in \mathcal{C}_\alpha.$$
4. *The number of all codewords in $\mathcal{C}_\alpha$ is $\binom{n}{\alpha} \cdot 2^\alpha$.*

Proof. By Definition 1, the fourth statement is straightforward. Now, we prove the other three statements.

1. It is sufficient to show that the maximum inner product value is $1-\frac{1}{\alpha}$ since the cosine function is decreasing between 0 and π. By Definition 1, each codeword in $\mathcal{C}_\alpha$ has exactly α non-zero elements with the same absolute value $\frac{1}{\sqrt{\alpha}}$. Thus, it is obvious that two codewords sharing $\alpha - 1$ non-zero positions with the same sign

*Corresponding author: J. H. Seo (e-mail: jaehongseo@hanyang.ac.kr)

for each entry have the maximum inner product value $\frac{\alpha-1}{\alpha} = 1 - \frac{1}{\alpha}$.

2. It is sufficient to show that for each $\mathbf{c} \in \mathcal{C}_\alpha$, the USample algorithm returns $\mathbf{c}$ with the probability $1/|\mathcal{C}_\alpha|$. At Step 1 of the USample algorithm in Algorithm 4, the same positions for non-zero elements as those of $\mathbf{c}$ are selected with the probability $\frac{1}{\binom{n}{\alpha}}$. At Step 2 of the USample algorithm, the same sign as those of $\mathbf{c}$ are selected with the probability $\frac{1}{2^\alpha}$. Both steps are independent and so the probability that the Usample algorithm returns $\mathbf{c}$ is $\frac{1}{2^\alpha \cdot \binom{n}{\alpha}}$, which is equal to $\frac{1}{|\mathcal{C}_\alpha|}$ by the fourth statement.

3. Let $\mathbf{u} = (u_1, ..., u_n)$ be an element in S^{n-1} and $\mathbf{c}$ be the output of the Decode algorithm with input $\mathbf{u}$. Let $J = \{j_1, \ldots, j_\alpha\}$ be the set of indices that $\mathbf{c}$ has non-zero entries. Then, from the description of the Decode algorithm, we have

$$\begin{aligned}\langle \mathbf{u}, \mathbf{c} \rangle &= u_{j_1} \cdot \frac{u_{j_1}}{|u_{j_1}|\sqrt{\alpha}} + \cdots + u_{j_\alpha} \cdot \frac{u_{j_\alpha}}{|u_{j_\alpha}|\sqrt{\alpha}} \\ &= \frac{1}{\sqrt{\alpha}}\left(|u_{j_1}| + \cdots + |u_{j_\alpha}|\right) = \frac{\sum_{k=1}^{\alpha} |u_{j_k}|}{\sqrt{\alpha}}.\end{aligned}$$

Similarly, for any other codeword $\mathbf{c}' \neq \mathbf{c}$, the value $\langle \mathbf{u}, \mathbf{c}' \rangle$ is equal to the same sum with J' where J' is the set of indices that $\mathbf{c}'$ has non-zero entries. Thus, the inequality $|u_j| \geq |u_k|$ for $\forall j \in J, \forall k \in [1, n] \setminus J$ given in the description of the Decode algorithm guarantees that $\langle \mathbf{u}, \mathbf{c} \rangle \geq \langle \mathbf{u}, \mathbf{c}' \rangle$ for all $\mathbf{c}' \in \mathcal{C}_\alpha$ if J' is distinct from J. □

Proposition 2. *Given two unit vectors* $\mathbf{t}$ *and* $\mathbf{c}$*, let* $\mathbf{w} = \mathbf{c} - \mathbf{t}^T\mathbf{c}\,\mathbf{t}$ *and* $\mathbf{R}_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$ *where* $\theta = \cos^{-1}(\langle \mathbf{t}, \mathbf{c} \rangle)$*. Then, the following matrix* $\mathbf{R}$ *is an orthogonal matrix such that* $\mathbf{Rt} = \mathbf{c}$*:*

$$\mathbf{R} = \underbrace{\mathbf{I} - \mathbf{t}\mathbf{t}^T - \mathbf{w}\mathbf{w}^T}_{\textit{projection part}} + \underbrace{[\mathbf{t}\ \mathbf{w}]\mathbf{R}_\theta[\mathbf{t}\ \mathbf{w}]^T}_{\textit{rotation part}}$$

where $\mathbf{I}$ *is the* $n \times n$ *identity matrix.*

Proof. Since

$$\begin{aligned}\mathbf{R}\mathbf{R}^T &= \left(\mathbf{I} - \mathbf{t}\mathbf{t}^T - \mathbf{w}\mathbf{w}^T\right)\left(\mathbf{I} - \mathbf{t}\mathbf{t}^T - \mathbf{w}\mathbf{w}^T\right)^T \\ &\quad + \left(\mathbf{I} - \mathbf{t}\mathbf{t}^T - \mathbf{w}\mathbf{w}^T\right)\left([\mathbf{t}\ \mathbf{w}]\mathbf{R}_\theta[\mathbf{t}\ \mathbf{w}]^T\right)^T \\ &\quad + \left([\mathbf{t}\ \mathbf{w}]\mathbf{R}_\theta[\mathbf{t}\ \mathbf{w}]^T\right)\left(\mathbf{I} - \mathbf{t}\mathbf{t}^T - \mathbf{w}\mathbf{w}^T\right)^T \\ &\quad + [\mathbf{t}\ \mathbf{w}]\mathbf{R}_\theta[\mathbf{t}\ \mathbf{w}]^T\left([\mathbf{t}\ \mathbf{w}]\mathbf{R}_\theta[\mathbf{t}\ \mathbf{w}]^T\right)^T \\ &= \mathbf{I}\end{aligned}$$

from the following relations,

$$\begin{aligned}\left(\mathbf{I} - \mathbf{t}\mathbf{t}^T - \mathbf{w}\mathbf{w}^T\right)\left(\mathbf{I} - \mathbf{t}\mathbf{t}^T - \mathbf{w}\mathbf{w}^T\right)^T &= \mathbf{I} - \mathbf{t}\mathbf{t}^T - \mathbf{w}\mathbf{w}^T, \\ [\mathbf{t}\ \mathbf{w}]\mathbf{R}_\theta[\mathbf{t}\ \mathbf{w}]^T\left([\mathbf{t}\ \mathbf{w}]\mathbf{R}_\theta[\mathbf{t}\ \mathbf{w}]^T\right)^T &= \mathbf{t}\mathbf{t}^T + \mathbf{w}\mathbf{w}^T, \\ \left(\mathbf{I} - \mathbf{t}\mathbf{t}^T - \mathbf{w}\mathbf{w}^T\right)\left([\mathbf{t}\ \mathbf{w}]\mathbf{R}_\theta[\mathbf{t}\ \mathbf{w}]^T\right)^T &= \mathbf{0}, \text{ and}\end{aligned}$$

$\mathbf{R}$ is an orthogonal matrix. Furthermore, since $\mathbf{t}$ and $\mathbf{w}$ are orthonormal, we can represent $\mathbf{c}$ as $\mathbf{c} = \langle \mathbf{c}, \mathbf{t} \rangle \mathbf{t} + \langle \mathbf{c}, \mathbf{w} \rangle \mathbf{w} = \mathbf{t}\cos\theta + \mathbf{w}\sin\theta = \mathbf{c}$. So, we also have

$$\begin{aligned}\mathbf{Rt} &= \left(\mathbf{I} - \mathbf{t}\mathbf{t}^T - \mathbf{w}\mathbf{w}^T + [\mathbf{t}\ \mathbf{w}]\mathbf{R}_\theta[\mathbf{t}\ \mathbf{w}]^T\right)\mathbf{t} \\ &= \mathbf{t}\cos\theta + \mathbf{w}\sin\theta = \mathbf{c}.\end{aligned}$$

Therefore, $\mathbf{R}$ is an orthogonal matrix such that $\mathbf{Rt} = \mathbf{c}$. □

C. Obstacles for Solving Equation (1) using a TMTO Strategy

In this section, we explain how to apply a time-memory-trade-off (TMTO) strategy to solve Equation (1) and introduce obstacles that are occurred in the process. To help the readers' understanding, we first consider the exact version of Equation (1) that has no error part. That is, we try to find two codewords $(\mathbf{c}_1, \mathbf{c}_2)$ in $\mathcal{C}_\alpha$ such that

$$\mathbf{P}\mathbf{c}_1 = \mathbf{c}_2 \tag{2}$$

for given an orthogonal matrix $\mathbf{P}$. For simplicity, we assume that α is even, but it is naturally extended to the case that α is odd.

Let $\mathbf{p}_i$ be the i-th column of $\mathbf{P}$. Then, we can re-write Equation (2) as

$$c_{11}\mathbf{p}_1 + c_{12}\mathbf{p}_2 + \cdots + c_{1n}\mathbf{p}_n = \mathbf{c}_2 \tag{3}$$

where $\mathbf{c}_1 = (c_{11}, c_{12}, \cdots, c_{1n})$.

By using the fact that $\mathbf{c}_1$ has the exact α non-zero elements whose absolute values are the same as $\frac{1}{\sqrt{\alpha}}$, we can set two vectors of length n, $\mathbf{a}$ and $\mathbf{b}$, such that they have the exact $\alpha/2$ non-zero components whose absolute values are the same as $\frac{1}{\sqrt{\alpha}}$, and there is no location that both $\mathbf{a}$ and $\mathbf{b}$ have the non-zero components. Then, Equation (3) is equivalent to

$$\sum_{\mathbf{a}=(a_1,\cdots,a_n)} a_i\mathbf{p}_i + \sum_{\mathbf{b}=(b_1,\cdots,b_n)} b_j\mathbf{p}_j = \mathbf{c}_2 \tag{4}$$

for some $\mathbf{a}$ and $\mathbf{b}$, both of which are determined by $\mathbf{c}_1$. The goal of our TMTO attack is to reduce the searching space of $\mathbf{c}_1$ at the cost of about square-root size memory and the above equation gives a hint for our purpose. We observe that two terms $\sum_{\mathbf{a}=(a_1,\cdots,a_n)} a_i\mathbf{p}_i$ and $\sum_{\mathbf{b}=(b_1,\cdots,b_n)} b_j\mathbf{p}_j$ in the left hand side of Equation (4) share the same structure and $\mathbf{c}_2$ in the right hand side is of the special form

(e.g., sparse vector consisting of only 0 and $\pm\frac{1}{\sqrt{\alpha}}$). When we compute $\sum_{\mathbf{d}=(d_1,\cdots,d_n)} d_i\mathbf{p}_i$ for all possible vectors $\mathbf{d}$ having the exact $\alpha/2$ non-zero components whose absolute values are $\frac{1}{\sqrt{\alpha}}$ and store them at a table $\mathcal{T}$, both $\sum_{\mathbf{a}=(a_1,\cdots,a_n)} a_i\mathbf{p}_i$ and $\sum_{\mathbf{b}=(b_1,\cdots,b_n)} b_j\mathbf{p}_j$ are elements in $\mathcal{T}$. Then, due to the special form of $\mathbf{c}_2$, Equation (4) helps us to efficiently search both $\sum_{\mathbf{a}=(a_1,\cdots,a_n)} a_i\mathbf{p}_i$ and $\sum_{\mathbf{b}=(b_1,\cdots,b_n)} b_j\mathbf{p}_j$, not a sequential-manner but at the same time from the table $\mathcal{T}$, which is roughly of square-root size.

We provide a precise description of our TMTO attack algorithm with detailed analysis.

1. Generate a table $\mathcal{T}$ that stores all possible pairs of vector $\mathbf{a}$ and the corresponding vector $\mathbf{q}_i = \sum_{\mathbf{a}} a_i\mathbf{p}_i$ where $\mathbf{a}$ is a vector of length n that has the exact $\alpha/2$ non-zero components and their absolute values are $\frac{1}{\sqrt{\alpha}}$. (Note that all possible candidates for $\sum_{\mathbf{a}} a_i\mathbf{p}_i$ are exactly the same as those for $\sum_{\mathbf{b}} b_j\mathbf{p}_j$. So, we can use the same table.)

2. From the table $\mathcal{T}$, generate a sub-table $\mathcal{T}_i$ which stores pairs of vector $\mathbf{a}$ and the i-th component of the corresponding vector $\mathbf{q}_i$ for $1 \le i \le \ell$. Then, sort tables $\mathcal{T}_i$'s each.

3. For each $\mathbf{q}_i$ stored at $\mathcal{T}$, search a table $\mathcal{T}_i$ to find elements such that the sum of the i-th component of $\mathbf{q}_i$ and the value stored at $\mathcal{T}$ is 0, $\frac{1}{\sqrt{\alpha}}$, or $-\frac{1}{\sqrt{\alpha}}$. If the condition holds, we collect the corresponding $(\mathbf{a}, \mathbf{b})$ where $\mathbf{a}$ is the vector corresponding to $\mathbf{q}_i$ in $\mathcal{T}$ and $\mathbf{b}$ is the vector corresponding to the i-th component in $\mathcal{T}_i$. Run this step for tables $\mathcal{T}_1, \cdots, \mathcal{T}_\ell$.

4. For all pairs of vectors $(\mathbf{a}, \mathbf{b})$ such that they satisfy the condition in Step 3 for all tables $\mathcal{T}_i$'s and there is no location that both $\mathbf{a}$ and $\mathbf{b}$ have non-zero components, check whether

$$\sum_{\mathbf{a}=(a_1,a_2,\cdots,a_n)} a_i\mathbf{p}_i + \sum_{\mathbf{b}=(b_1,b_2,\cdots,b_n)} b_j\mathbf{p}_j$$

is a codeword in $\mathcal{C}_\alpha$ or not. If it is a codeword, return $\mathbf{c}_1 = \mathbf{a} + \mathbf{b}$ and $\mathbf{c}_2 = \sum_{\mathbf{a}} a_i\mathbf{p}_i + \sum_{\mathbf{b}} b_j\mathbf{p}_j$.

Now, we calculate the complexity of the above algorithm. Let N be the number of elements stored at $\mathcal{T}$. Then, $N = \binom{n}{\alpha/2}2^{\alpha/2}$. Step 1 takes N additions of $\alpha/2$ n-dimensional vectors. Step 2 and Step 3 take $\mathcal{O}(\ell N \log N)$ and $O(\ell \log N)$ comparisons, respectively. Finally, Step 4 takes c additions of α n-dimensional vectors where c is the number of pairs that satisfy the conditions stated in Step 4. Therefore, if ℓ and c is sufficiently small, e.g., both are logarithmic in N, then the above algorithm takes a quasi-linear time in N.

One may try to check Equation (4) with all candidates after Step 1. We remark that there is no efficient way to execute it since $\mathbf{c}_2$ is also hidden. That is, the key idea to reduce the complexity of solving by using our method is Steps 2 and 3 which exploits the fact that $\mathbf{c}_2$ is of the special form that has the exact α non-zero elements and their absolute values are the same.

Next, we move our attention to solving Equation (1), which can be regarded as the erroneous version of Equation (2). Similarly to the above, we may apply to the TMTO strategy by mitigating the condition at Step 3 so that it collects the vectors such that the result sum is in some range, not one of the exact values 0, $\frac{1}{\sqrt{\alpha}}$ and $-\frac{1}{\sqrt{\alpha}}$, to reflect the errors generated by $\mathbf{P}_2\mathbf{e}$ in Equation (1). However, in this case, there are still remained many $(\mathbf{a}, \mathbf{b})$'s after Step 3, that is, c is too large in the analysis of the above algorithm. So, we cannot reduce the complexity of the algorithm as we expect.

D. Discussions

D.1. Matching Score

The verification of IronMask outputs only the binary score, 'match'/'no match'. We would like to note that most biometric cryptographic schemes, which do not use cryptographic encryption schemes, employ cryptographic hash functions. Thus, their verification processes output the binary score as ours.

Nevertheless, one can indirectly control a threshold for TAR-FAR rate by adjusting parameters of the underlying face recognition system. There is a more direct method for handling this issue by loosening requirements in the matching process: In the registration, a user stores several hashed codewords in close proximity to each other, unlike storing one hashed codeword only in the current registration. Thereafter, in the verification, we may use a modified decoding algorithm that outputs a set of close codewords, instead of only the closest codeword, and check their hashed values with stored ones. From these relaxations, one can control a threshold since those enable to check more approximate matches. Although this direct solution is a plausible candidate, more plentiful analyses and experiments are necessary for resolving this issue completely. We leave a detailed analysis as further study.

D.2. CosFace Experimental Phenomenon

The performance degradation of CosFace with IronMask is considerable compared to the ArcFace case. (See 'CF+I' row in Table 1.) Let us present our interpretation for this experimental phenomenon. First, we note that we used hyperparameters of ArcFace and CosFace, which were set to get the best performance only, regardless of IronMask. In fact, in order to understand this phenomenon by our-

selves, we have evaluated the average cosine values for the same person for both ArcFace and CosFace. As a result, we found that the cosine value of ArcFace is higher than that of CosFace: For example, the average cosine values in ArcFace and CosFace for CMU-MultiPIE dataset are 0.89 and 0.80, respectively. We have inferred that IronMask's decoding process relatively well harmonizes with recognition systems with smaller intra-class variation and its boundary lies somewhere between 0.80 and 0.89. This causes considerable performance degradation for the CosFace case. To complement this effect, we devise a new centering method using multiple images. In CosFace (and ArcFace), each template is considered as a vector in S^{n-1} with cosine similarity distance and thus simple averaging with the Euclidean distance cannot generate a meaningful center. To overcome this obstacle, we exploited the multiple linear regression to find a center and it improves the intra-class variation: For example, we confirm by experiments that the cosine value of CosFace in the above setting increases from 0.80 to 0.87. Moreover, as results shown in Table 1, IronMask is well harmonized with both CosFace and ArcFace for several datasets with our new centering method.

References

[1] Kai Cao and Anil K Jain. Learning fingerprint reconstruction: From minutiae to image. *IEEE Transactions on information forensics and security*, 10(1):104–117, 2015. 1

[2] R Cappelli, A Lumini, D Maio, and D Maltoni. Fingerprint image reconstruction from standard templates. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 29, pages 1489–1503. arXiv preprint, 2007. 1

[3] M Fredrikson, S Jha, and T Ristenpart. Model inversion attacks that exploit confidence information ad basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333, 2015. 1

[4] Javier Galbally, Arun Ross, Marta Gomez-Barrero, Julian Fierrez, and Javier Ortega-Garcia. Iris image reconstruction from binary templates: An efficient probabilistic approach based on genetic algorithms. *Computer Vision and Image Understanding*, 117(10):1512–1525, 2013. 1

[5] Guangcan Mai, Kai Cao, Pong C Yuen, and Anil K Jain. On the reconstruction of face images from deep face templates. *IEEE transactions on pattern analysis and machine intelligence*, 41(5):1188–1202, 2018. 1