Proactive Deepfake Defence via Identity Watermarking

Supplementary Material

A. Preliminary

Pseudorandom Noise (PN) sequence is widely used in signal processing, which is usually a binary sequence with a spectrum similar to a random sequence but generated by a deterministic algorithm. Linear feedback shifter register (LFSR) is one of the simplest ways to generate a PN sequence. In an LFSR, any bit is determined by a linear combination of the previous n bits, which can be formulated as:

$$B_n = A_0B_0 \oplus A_1B_1 \oplus A_2B_2 \oplus ... \oplus A_{n-1}B_{n-1} \quad (12)$$

Since any bit is a function of the previous n bits, every LFSR can produce a sequence of bits that appears random and has a period of $2^n - 1$. There are several commonly used PN sequences which are: maximum length sequence (MLS), Gold sequences, Kasami sequences and JPL sequences.

MLS sequence is the most representative PN sequence which is generated using maximal LFSR. They are periodic and reproduce every binary sequence (except the zero vector) that can be represented by the shift registers, i.e., for m length registers they produce a sequence of with length $2m - 1$. The auto-correlation of an MLS is approached to unit impulse function as MLS length increases. This property makes the MLS suitable for synchronization and in the detection of information in single-user Direct Sequence Spread Spectrum systems.

The Gold sequence is another well-known PN sequence that belongs to the category of product codes for they produced by XOR'ing two same length MLS. The two MLS must maintain the same phase relationship till all the additions are performed. A slight change of phase even in one of the MLS produces a different Gold sequence altogether. Gold codes are non-maximal and therefore they have poor auto-correlation property when compared to MLS. However, it possesses good cross-correlation properties which are useful when multiple devices are broadcasting in the same frequency range, such as the applications like CDMA and satellite navigation.

B. Application Scenario

Fig. 8 demonstrate how to deploy our method in the real world. In the scenario without applying our method (Fig. 8 top panel), a user shared his/her images to a social network, such as Facebook or Instagram. Once the image is uploaded online, the user cannot verify its authenticity anymore. Therefore, malicious users can not only pick victims' photos and manipulate them to create Deepfakes but also release the synthesized results while falsely claiming these images are authentic. The misinformation will cause severe reputation loss for the victim and raise security and privacy concerns.

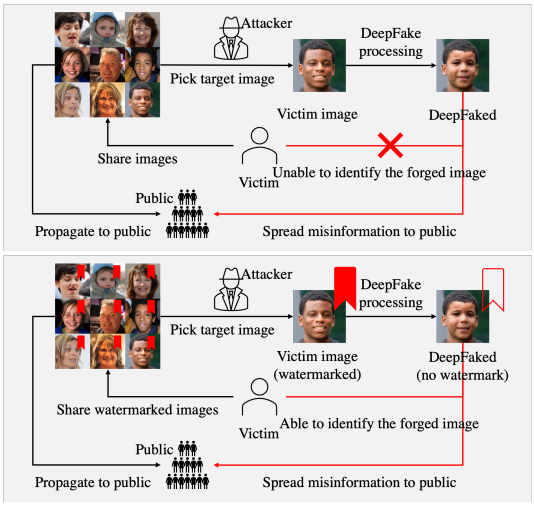

Figure 8. Comparison between the unprotected social network (top panel) and the social network protected by our method (bottom panel) in handling misinformation spreading by Deepfake attacker.

On the contrary, in the scenario where the user employed our method (Fig. 8 bottom panel), the images shared online will be embedded with the user's personalized watermark. The embedded watermark is invisible to humans and robust to conventional manipulations. Thus there is negligible impact on the image's visual quality and application utility. Nevertheless, unlike other visible or invisible watermarking techniques, our watermark is sensitive to Deepfake manipulations. Once the malicious users apply Deepfake techniques on watermarked images, the corresponding embedded message will be destroyed. According to the existence of the watermark, the real and fake images can be effectively differentiated. Therefore, the user can utilize our method to reduce the negative impact of Deepfake by identifying the forged information and authenticating the real information.

C. Network Architecture

Our framework is built upon FaceShifter's AEI-Net [24] but modified by us to implement face identity watermarking rather than face-swapping. The entire network, illustrated in Fig. 9, consists of an Identity Encoder, a U-Net style Attributes Encoder, and a Generator composed of 8 cascaded AAD Residual Blocks. Detailed structures of each component are in Fig 10.

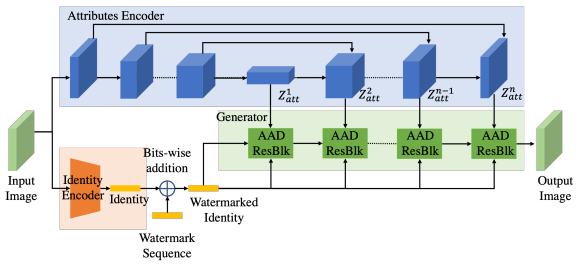

Figure 9. Detailed architecture of Watermark Injection.

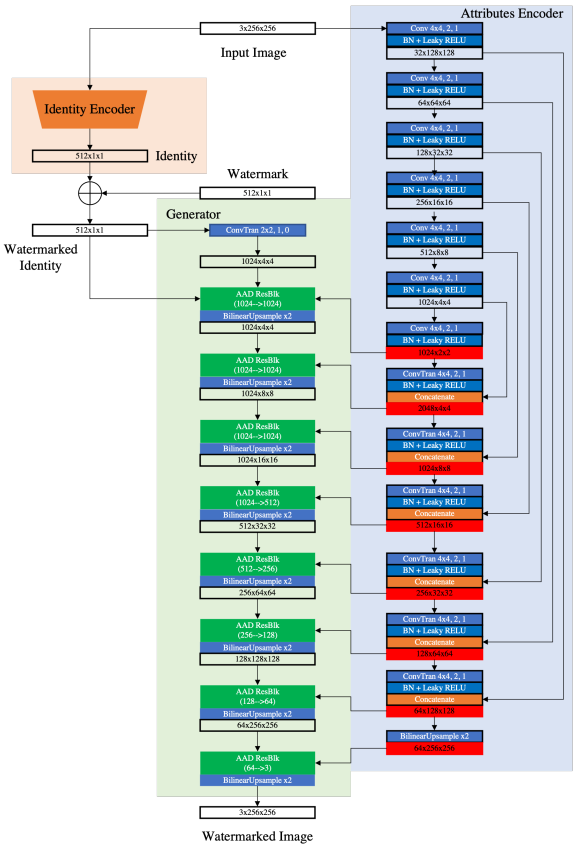

Figure 10. Network structures. Following the structures of FaceShifter [24]: *Conv k,s,p* represents a Convolutional Layer with kernel size *k*, stride *s* and padding *p*. *ConvTran k,s,p* represents a Transposed Convolutional Layer with kernel size *k*, stride *s* and padding *p*. All *Leaky ReLUs* have $\alpha = 0.1$. *AAD ResBlk*(c_{in}, c_{out}) represents an AAD ResBlk with input and output channels of c_{in} and c_{out}.

D. Training Details

The concept diagram of our framework's training procedure is illustrated in Fig. 11. In Fig. 11, black arrows refer to data flows from the input image to the watermarked image and Discriminator, while the red lines indicate data flow for each loss function. Besides, the trapezoids with a red dash represent a trainable network, while black sold line trapezoids represent fixed networks. We utilize adversarial training for our framework where the Discriminator adopts

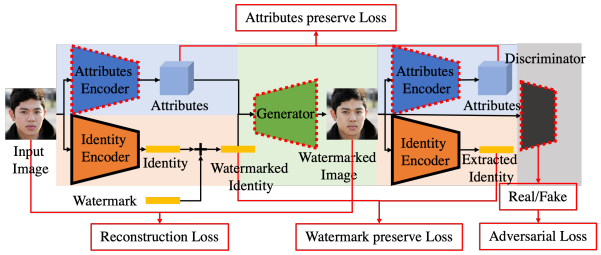

Figure 11. Training procedure

Table 7. Parameters and FLOPs of different detection methods.

	Parameters	FLOPs
BTS	45.907M	95.1557G
CD	23.5101M	5.3965G
ICPR	0.1271M	0.8869G
PF	23.51M	2.1G
RFM	20.8111M	6.0116G
SBI	0.1288M	0.1981G
Ours (Injection)	396.0907M	83.8398G
Ours (Verification)	43.7977M	6.3236G

a multi-scale network [36].

Given an input face image X, the identity encoder and attributes encoder respectively extract the 1D $1x512$ identity representation $z_{id}(X)$ and multi-level attributes representation $z_{att}(X)$ from the image. Then we bit-wise add watermark sequence z_{seq} to the identity representation $z_{id}(X)$ to produce watermarked identity $z^{w}_{id}(X)$. The Generator finally integrates the watermarked identity $z_{id}(X)$ and original attributes representation $z_{att}(X)$ to produce the watermarked facial image $\hat{X}$. After obtained $\hat{X}$, we calculating different losses according to E.q. 7,8,9,10 and update all trainable networks. As we can see in the scheme, non-extra annotations are required in our training process.

E. Computational Overhead

According to Table 7, the primary computational cost of our method is the watermark injection, while the verification(detection) stage's overheads are close to other detection approaches. Considering the detection performance of our method, we believe it is a balanced trade-off between computational costs and detection accuracy.

F. Security Analysis

To verify the security of our method of confronting worst-case threaten, we simulated attacks to the application

Table 8. Correlation results of Attack Model 1.

Sequences types	Auto-correlation results			DR
	Peak	Average	PAR	
Gaussian	0.78	0.53	1.47	0.6%
Gold	0.95	0.53	1.79	2.98%
Laplace	1.01	0.53	1.89	0.13%
MLS	0.77	0.53	1.46	1.6%

scenarios of our method. The objective of the adversaries is to utilize the knowledge about our watermarking mechanism to forge the watermark. Here, we consider three strong attack models:

Attack Model 1: The adversary can obtain the victim's entire watermarking framework and its corresponding pre-trained models but knows nothing about the watermark sequence. Thus, the adversary tries to embed different sequences on Deepfaked images via the obtained network to deceive the watermark verification step.

Attack Model 2: The adversary knows the victim's watermark sequence but cannot obtain the corresponding framework. Thus, the adversary utilizes the knowledge about our method trying to build and train a similar watermarking network to embed the victim's watermark on Deepfaked images to deceive the watermark verification step.

Attack Model 3: The adversary stealthily collects the victim's watermarked images and employs these images as dataset to train his/her Deepfake method to generate fake images to deceive the watermark verification step.

F.1. Attack Model 1

For Attack Model 1, we simulate the forging process by utilizing one fixed watermarking framework to synthesize four groups of watermarked images according to different types of sequences. Then, we randomly generate another sequence to act as the victim's watermark and use the same framework to implement watermark verification on these watermarked images. As can be seen from Table 8, the correlation results of different watermarked images are close to non-watermarked images. Therefore, even if the victim's watermarking framework leaked, the forged watermarked image cannot pass the corresponding watermark verification step.

F.2. Attack Model 2

For Attack Model 2, we employ another face recognition network, namely circularface, as the identity encoder to constitute a new watermark framework then utilize both arcface $Model_{Arc}$ and circularface $Model_{Cir}$ watermark networks to generate watermarked images by embedded same sequence respectively. To simulate the deceive process, we use $Model_{Arc}$'s network to verify $Model_{Cir}$'s outputs and $Model_{Cir}$'s network to verify $Model_{Arc}$'s outputs. The correlation results reported in Table 9 demonstrate that different frameworks cannot generate the same watermarked results even embedded in the same sequence.

Table 9. Correlation results of Attack Model 2.

Networks	Auto-correlation results			DR
	Peak	Average	PAR	
$Model_{Arc}$	0.71	0.53	1.33	1.28%
$Model_{Cir}$	0.73	0.53	1.37	1.63%

F.3. Attack Model 3

For Attack Model 3, we train Faceshifter [24] by 40k Gold sequence watermarked images to imitate the attack scenario and test the specific performance. The Faceshifter is one of the most representative Deepfake methods which generates Deepfake image by utilizing the source image's identity and target image's attributes. Hence, the fake image from Faceshifter would preserve the source's identity feature and the target's attributes feature.

After training the Faceshifter model to coverage, we subdivide the Deepfake generate process into six different cases: 1) Using the non-watermarked image as source and non-watermarked image as the target, denoted as $Id_{ori}Att_{ori}$; 2) Using the non-watermarked image as source and watermarked image as the target, denoted as $Id_{ori}Att_{wat}$; 3) Using the watermarked image as source and non-watermarked image as the target, denoted as $Id_{wat}Att_{ori}$; 4) Using the watermarked image as source and watermarked image as the target, denoted as $Id_{wat}Att_{wat}$; 5) Using the non-watermarked image as source and non-watermarked image as the target but verification with different Gold sequence, denoted as $Id_{wat}Att_{ori}DS$; 6) Using the non-watermarked image as source and non-watermarked image as the target but verification with different Gold sequence, denoted as $Id_{wat}Att_{wat}DS$.

The experiment results are summarized in Table 10. We can see even the Faceshifter is trained with the watermarked images, but unless the fake image is generated from the source image which has the same watermarked sequence with the verification model, otherwise the correlation results and DR are the same with the non-watermarked image.

G. Proactive Comparison

According to the requirement of AGF, we employ AGF to fingerprint 15k Celeba images and train FaceShifter model A on these fingerprinted images. Then, we use

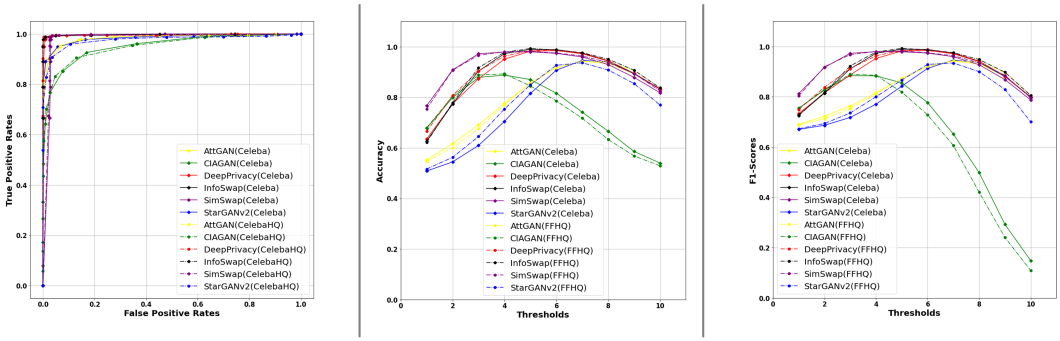

Figure 12. Our method's **ROC** ↑, **Accuracy** ↑ and **F1 Score** ↑ on different Deepfake and datasets images. The ROC curve indicates our method has excellent discriminability on both low- and high-resolution Deepfake results. Besides, according to the trend of Accuracy and F1 Scores, our method can achieve different performances under different PAR thresholds.

Table 10. Correlation results of Attack Model 3.

Networks	**Auto-correlation results**			**DR**
	Peak	**Average**	**PAR**	
$Id_{ori}Att_{ori}$	1.13	0.52	2.15	3.6%
$Id_{ori}Att_{wat}$	0.58	0.52	1.11	0.1%
$Id_{wat}Att_{ori}$	3.02	0.53	5.68	92.9%
$Id_{wat}Att_{wat}$	3.6	0.53	6.77	94.67%
$Id_{wat}Att_{ori}DS$	0.58	0.53	1.1	1.33%
$Id_{wat}Att_{wat}DS$	0.64	0.52	2.62	1.2%

trained FaceShifter model A to generate 5k Deepfake results on original Celeba images without AGF fingerprint. We employ the AGF to conduct detection on the set, which mixes the 5k Deepfaked images with another randomly selected 5k original Celeba images and summarized results in Table 4.

We train another FaceShifter model B for our method on the same 15k Celeba images but without AGF fingerprints. We employ our method to inject watermarks into 5k Celeba images, which are the same image in the AGF process. Then, we use trained FaceShifter model B to generate Deepfake results based on our watermarked images and mix another 5k original Celeba image to form a test set. We employ our method to conduct detection on this test set and report the results in Table 4.

H. Classification Analysis

To further explore our method's real and fake classification capability, we plot our method's ROC, accuracy, and F1-Score curves when adopting decision thresholds ranging from 1 to 10 with a step size of 1. As illustrated in Fig. 12, our method can achieve excellent classification capability when facing different Deepfake methods. Besides, the F1 Score and ACC curves indicate that our method has better detection performance on CIAGAN and StarGAN2 when adopting a threshold of 3 and 7, respectively.