A Secure Image Watermarking Framework with Statistical Guarantees via Adversarial Attacks on Secret Key Networks: Supplemental Material

The supplemental contains the following sections:

- §S1: Proof of Eq 2: Wasserstein distance calculation.
- §S2: Proof of Eq 6: P-value formula of HT4A derivation.
- §S3: Preliminaries on adversarial examples and hypothesis test.
- §S4: Properties of normal distributions for watermark hypothesis tests.
- §S5: Experiment setup details.
- §S6: Methods of verifying model output's normality.
- §S8: Discussion on secure host implementation.
- §S9: Watermark imperceptibility and examples.
- §S10: Three different ways of defining signatures.
- §S11: Watermarking performance on two unseen datasets.
- §S12: Runtime about watermarking and detection.
- §S13: Choice of target length in adversarial loss.

S1 Proof of Eq. 2: Wasserstein Distance

When given two multivariate normal distributions q, p with mean vectors μ_q, μ_p and covariance matrices Σ_q, Σ_p, the Wasserstein distance $\mathcal{W}$ from [8] is

$$\mathcal{W}[\mathrm{q},\mathrm{p}] = \|\boldsymbol{\mu}_\mathrm{q} - \boldsymbol{\mu}_\mathrm{p}\|^2 + \operatorname{tr}(\boldsymbol{\Sigma}_\mathrm{q}) + \operatorname{tr}(\boldsymbol{\Sigma}_\mathrm{p}) - 2\operatorname{tr}\left((\boldsymbol{\Sigma}_\mathrm{q}\boldsymbol{\Sigma}_\mathrm{p})^{1/2}\right), \tag{8}$$

where $\operatorname{tr}(\cdot)$ is the trace operator. This expression is also called Fréchet distance. Another common expression for the $\mathcal{W}$ from [12, 27] is written as

$$\mathcal{W}[\mathrm{q},\mathrm{p}] = \|\boldsymbol{\mu}_\mathrm{q} - \boldsymbol{\mu}_\mathrm{p}\|^2 + \operatorname{tr}(\boldsymbol{\Sigma}_\mathrm{q}) + \operatorname{tr}(\boldsymbol{\Sigma}_\mathrm{p}) - 2\operatorname{tr}\left(\left(\boldsymbol{\Sigma}_\mathrm{q}^{1/2}\boldsymbol{\Sigma}_\mathrm{p}\boldsymbol{\Sigma}_\mathrm{q}^{1/2}\right)^{1/2}\right). \tag{9}$$

The equivalence between (8) and (9) can be proved by noting that:

$$\begin{aligned}\operatorname{tr}\left(\left(\boldsymbol{\Sigma}_p^{1/2}\boldsymbol{\Sigma}_q\boldsymbol{\Sigma}_p^{1/2}\right)^{1/2}\right) &= \operatorname{tr}\left(\left(\boldsymbol{\Sigma}_p^{1/2}\boldsymbol{\Sigma}_q\boldsymbol{\Sigma}_p^{1/2}\right)^{1/2}\boldsymbol{\Sigma}_p^{-1/2}\boldsymbol{\Sigma}_p^{1/2}\right)\\ &= \operatorname{tr}\left(\boldsymbol{\Sigma}_p^{1/2}\left(\boldsymbol{\Sigma}_p^{1/2}\boldsymbol{\Sigma}_q\boldsymbol{\Sigma}_p^{1/2}\right)^{1/2}\boldsymbol{\Sigma}_p^{-1/2}\right).\end{aligned} \tag{10}$$

Then we have

$$\left(\boldsymbol{\Sigma}_\mathrm{p}^{1/2}\left(\boldsymbol{\Sigma}_\mathrm{p}^{1/2}\boldsymbol{\Sigma}_\mathrm{q}\boldsymbol{\Sigma}_\mathrm{p}^{1/2}\right)^{1/2}\boldsymbol{\Sigma}_\mathrm{p}^{-1/2}\right)^2 = \boldsymbol{\Sigma}_\mathrm{p}\boldsymbol{\Sigma}_\mathrm{q}. \tag{11}$$

Therefore, (8) and (9) are equivalent.

For the convenience of computation, we choose the form of (8). If we assume a target distribution is a standard multivariate normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ in $\mathbb{R}^d$, then (8) becomes,

$$\mathcal{L}_\mathrm{c} = \boldsymbol{\mu}_d^T\boldsymbol{\mu}_d + \operatorname{tr}(\boldsymbol{\Sigma}_d) + d - 2\operatorname{tr}\left(\boldsymbol{\Sigma}_d^{\frac{1}{2}}\right).$$

S2 Proof of Eq. 6: HT4A

Given that

$$X \sim N\left(0, I_d\right), \tag{12}$$

we have

$$\frac{X}{\|X\|} \sim U(S_{d-1}), \tag{13}$$

where

$$S_{d-1} = \frac{d \cdot \pi^{d/2}}{\Gamma\left(\frac{d}{2}+1\right)} \tag{14}$$

is the whole surface area in a unit sphere in the d dimensional space.

The p-value for the angle is given by

$$\begin{aligned} p &= \int_{\cos(w,x) \leqslant c} U(S_{d-1})\, dx \\ &= U(S_{d-1}) \int_{\cos(w,x) \leqslant c} 1\, dx. \end{aligned} \tag{15}$$

From the geometric view, the part

$$\int_{\cos(w,x) \leqslant c} 1\, dx \tag{16}$$

represents a solid angle Ω in an d-dimensional sphere.

Because the d-dimensional sphere is a unit one, the p-value formula becomes:

$$p = 1 - \frac{\text{surface area of } \Omega}{S_{d-1}} \tag{17}$$

We have the relationship:

$$\begin{aligned} S_{d-1} = g(r) &\times \int_0^{\pi} \sin^{(d-2)}\left(\phi_1\right)\, d\phi_1 \\ &\times \int_0^{\pi} \sin^{d-3}\left(\phi_2\right)\, d\phi_2 \\ &\cdots \\ &\times \int_0^{\pi} \sin\left(\phi_{d-2}\right)\, d\phi_{d-2} \\ &\times \int_0^{2\pi} d\phi_{d-1} \end{aligned} \tag{18}$$

where $g(r)$ is the part of computation about radius r.

For the solid angle Ω, we have:

$$\begin{aligned} S_\Omega = g(r) \times & \int_0^\theta \sin^{(d-2)}(\phi_1)\, d\phi_1 \\ & \times \int_0^\pi \sin^{d-3}(\phi_2)\, d\phi_2 \\ & \cdots \\ & \times \int_0^\pi \sin(\phi_{d-2})\, d\phi_{d-2} \\ & \times \int_0^{2\pi} d\phi_{d-1}. \end{aligned} \tag{19}$$

Thus, the probability becomes:

$$P = 1 - \frac{\int_0^\theta \sin^{d-2}(\phi_1)\, d\phi_1}{2\int_0^{\frac{\pi}{2}} \sin^{d-2}(\phi_1)\, d\phi_1}. \tag{20}$$

For the numerator expression, given that the length of the output vector is 32, an even number, the formula for d being even is:

$$\begin{aligned} \int_0^\pi \sin^{d-2}(\phi_1)\, d\phi_1 &= 2\int_0^{\frac{\pi}{2}} \sin^{d-2}(\phi_1)\, d\phi_1 \\ &= \frac{(d-3)!!}{(d-2)!!} \cdot \pi. \end{aligned} \tag{21}$$

And for the denominator expression, we have

$$2\int_0^{\frac{\theta}{2}} \sin^{d-2}(\phi_1)\, d\phi_1 = \frac{(d-3)!!}{(d-2)!!}\theta - \frac{(d-3)!!}{(d-2)!!}\cos\theta \sum_{k=1}^{\frac{d-2}{2}} \frac{(2k-2)!!}{(2k-1)!!}\sin^{2k-1}(\theta). \tag{22}$$

Substitute (21) and (22) into (17), and in the end, we obtain

$$P = 1 - \frac{\theta - \cos\theta \sum_{k=1}^{\frac{d-2}{2}} \frac{(2k-2)!!}{(2k-1)!!}\sin^{(2k-1)}(\theta)}{\pi}. \tag{23}$$

S3 Preliminaries

We present two techniques in our framework: generating adversarial examples and conducting hypothesis testing, viewed from a comprehensive perspective.

S3.1 Adversarial Examples

We employ a typical adversarial attack, Projected Gradient Descent (PGD) [26], for our watermarking backbone. The PGD attack generates adversarial examples by iteratively tweaking the noise η and adding it to input data to maximize the adversarial loss while keeping changes imperceptibly small. The perturbation η is updated for each iteration t using gradient ascent,

$$\eta_{t+1} = \eta_t + \alpha \cdot \operatorname{sign}\left(\nabla_I \mathcal{L}\left(f(I), y_{\text{target}}\right)\right). \tag{24}$$

Here $\mathcal{L}(f(I), y_{\text{target}})$ represents the adversarial loss, where $f(\cdot)$ is a specific DNN model, I is the input image, and y_{target} is the ground truth for a task. Mean-

while, $\nabla_I \mathcal{L}$ denotes the gradient computation based on the input image I. Subsequently, η_{t+1} is projected into an ϵ-bound to guarantee that the pixel values of the image do not vary beyond the specified range, thus the adversarial noise is imperceptible. Usually, L_∞-normal is used as the metric to measure the magnitude of this variation, *i.e.*, $\|\eta_{t+1}\|_\infty < \epsilon$. After completing all the iterations T, the perturbation η_T will be added to the image I to produce its corresponding adversarial example.

To make PGD attacks more applicable for watermarking in our proposed framework, we make some modifications. Firstly, we modify it by utilizing the gradient value directly, while (24) uses the gradient's sign for updating η,

$$\eta_{t+1} = \eta_t + \alpha \cdot \nabla_I \mathcal{L}\left(f(I), y_{\text{target}}\right). \tag{25}$$

Secondly, we expand $\|\eta_{t+1}\|_\infty < \epsilon$ into $\eta'_{t+1} = \beta \cdot \eta_{t+1}$, where β is the scale factor dependent on,

$$\beta = \text{clip}\left(\sqrt{\beta_{tg} / \text{ mean}\left(\eta_{t+1}^2\right)}, 0, 1\right). \tag{26}$$

Here, β_{tg} is a given target and another form of α, but with the same meaning and function, *i.e.*, sets an upper bound on the range of η.

Thirdly, before being inputted into the specific network $f(\cdot)$, the input image I is transformed by some data augmentation operations, like rotating and cropping, to enhance the robustness of watermarking detection. Therefore, (25) is changed to (27),

$$\eta_{t+1} = \eta_t + \alpha \cdot \nabla_I \mathcal{L}\left(f(da(I)), y_{\text{target}}\right), \tag{27}$$

where $da(\cdot)$ is the data augmentation module.

S3.2 Hypothesis Testing

Hypothesis testing is a fundamental procedure in statistics used to determine whether there is enough evidence in a sample of data to generalize a population parameter It involves two main hypotheses: null hypothesis $\mathcal{H}_0$ and alternative hypothesis $\mathcal{H}_1$. The outcome is to either reject $\mathcal{H}_0$ or fail to reject it based on the evidence presented by the sample. For this purpose, a test statistic ξ, which summarizes the sample data, must be calculated at first and assumed to follow a specific distribution when $\mathcal{H}_0$ is true. The p-value is then determined as the probability of obtaining the ξ at least as extreme as the one that was observed under the $\mathcal{H}_0$. Finally, $\mathcal{H}_0$ is rejected in favor of $\mathcal{H}_1$ if this p-value is less than a predetermined significance level α.

Note that the threshold α acts as a benchmark for decision-making and is conventionally set at 0.05, and it delineates the boundary between the rejection and acceptance regions for $\mathcal{H}_0$. Besides, the significance level is directly associated with the false positive error in hypothesis testing, which means the error occurs when the $\mathcal{H}_0$ is true, but the test incorrectly rejects it.

S4 Properties of Normal Distribution for Watermark Hypothesis Tests

In §3, our method primarily employs two types of hypothesis testing for watermark detection, each founded on distinct properties of the normal distribution. Specifically, the hypothesis test for SKN utilizes the characteristic outlined in Prop. 1, and the one for SKS is aligned with Prop. 2.

Property 1. Suppose $\mathbf{X}$ is a d-dimensional standard normal random vector, i.e., $\mathbf{x} \sim \mathcal{N}(\mathbf{0}_d, \mathbf{I}_d)$, where $\mathbf{I}_d$ is the k-dimensional identity matrix and all its components are independent standard normal random variables. Under this assumption, each x_i^2 (where $i = 1, 2, ..., d$) follows a standard chi-squared distribution with 1 degree of freedom. Hence, their sum, $\sum_i x_i^2 = \mathbf{x}^T\mathbf{x}$, is chi-squared ($\boldsymbol{\chi}^{\mathbf{2}}$) distributed with d degrees of freedom.

Property 2. Consider a Gaussian random vector $\mathbf{x}$ in $\mathbb{R}^d$. Assume it has the distribution: $\mathbf{x} \sim \mathcal{N}(\mathbf{0}_d, \mathbf{I}_d)$. Under this assumption, the probability density function (PDF) for the transformation $\mathbf{y} = \frac{\mathbf{x}}{\|\mathbf{x}\|}$ can be determined. Due to the Gaussian distribution's rotational symmetry, $\mathbf{y}$ is uniformly distributed on the $(d-1)$-dimensional unit hypersphere, denoted $\mathcal{S}_{d-1}$ in $\mathbb{R}^d$.

S5 Experiment Details

In this section, we present the details related to our experiment, including the environment, model architecture, and the training and testing procedure setup.
Experimental Setup: We use PyTorch version 2.0 and Python 3.9, with the RTX 3090 GPU for both the training and testing phases.
SKN Model Architecture: Resnet18 [14] is selected as the backbone for our model. To adapt it to our approach, the input layer at the beginning of the model is modified to a kernel size of 3, and the subsequent max-pooling is removed. After processing through all residual blocks, the feature map goes through an adaptive average pooling, resulting in a 4×4 output size. The final fully-connection layer outputs 32 dimensions. The remainder of our model aligns with the standard Resnet18 architecture.
SKN Training Procedure: For SKN training, the MSCOCO [22] is utilized. Images are initially resized to 160×160 and then randomly cropped to 128×128 before feeding into the model. The training batches consisted of 256 images each. The model is trained for 15 epochs. The optimizer is Adam, with a learning rate of 0.001 for the first 5 epochs and 0.0001 for the remaining 10. Initial weights of the model are randomly assigned. The weights of two terms in the training function $\mathcal{L}_{\text{gen}}$ in (1) are set to $\lambda_1 = 1$ and $\lambda_2 = 2.5$.
Testing Procedure: In the testing phase, individual images are processed one at a time. The PGD [26] adversarial attack settings included a perturbation bound of $\epsilon = 0.00063$, with 100 iterations and a step size of 0.01. The weights in the adversarial loss $\mathcal{L}_{\text{adv}}$ in (4) are set to $\lambda_3 = 0.1$ and $\lambda_4 = 200$.

S6 Hypothesis Tests for Normality

HT 1 is employed to determine if a dataset adheres to a normal distribution by comparing the distribution of the sample data with the expected normal distribution. However, this test is limited to a single variable. In our experiment Sec. 4.2 and Supp. S7, we examine each entry of the output vector separately and then calculate the proportion of entries contributing to the acceptance of $\mathcal{H}_0$. As defined in (28), $ECDF_m(\cdot)$ represents the empirical cumulative distribution function of the m samples, while $CDF(\cdot)$ signifies the target cumulative distribution function. Our target distribution is the normal distribution characterized by mean μ and variance σ^2, expressed as $CDF(i; \mu, \sigma^2)$ for the i-th sample.

While HT 1 assesses normality, it does not specifically evaluate standard normality (i.e., zero mean, unit variance). To address this, we perform two additional hypothesis tests: one to determine if the mean vector equals the zero vector (outlined in HT 2), and another to check if the covariance matrix is the identity matrix (described in HT 3). Specifically, the $\mathbf{A}$ in (33) is the sample deviation matrix, derived from the sample covariance matrix $\mathbf{S}$ in (34). The test dataset is segmented into 500 batches of 100 images each. Each batch serves as a set of test samples for the hypothesis test, and we calculate the proportion of batches that do not reject the null hypotheses $\mathcal{H}_0$.

Hypothesis Test 1 Testing $\mathbf{x}_j \sim \mathcal{N}\left(\mu, \sigma^2\right)$?

STEP 1: Define hypotheses,
$\mathcal{H}_0$: $\mathbf{x}_j \sim \mathcal{N}\left(\mu, \sigma^2\right)$;
$\mathcal{H}_1$: $\mathbf{x}_j \not\sim \mathcal{N}\left(\mu, \sigma^2\right)$.

STEP 2: Sample from the dataset,
Samples $m = 5000$;

STEP 3: Construct statistic

$$D = \sup_{i \in \{m_j\}} |ECDF_m(i) - CDF(i; \mu, \sigma^2)|. \tag{28}$$

STEP 4: Because D follows Kolmogorov distribution,
P-value p can be computed by

$$p = 2 \sum_{k=1}^{\infty} (-1)^{k-1} e^{-2k^2 D^2}. \tag{29}$$

STEP 5: Set the confidence α and get conclusion.
If the $p < \alpha$, reject the $\mathcal{H}_0$ and accept $\mathcal{H}_1$; Otherwise accept the $\mathcal{H}_0$.

Hypothesis Test 2 Testing $\boldsymbol{\mu}_d = \mathbf{0}$?

STEP 1: Define hypotheses,
$\mathcal{H}_0$: $\boldsymbol{\mu}_d = \mathbf{0}_d$;
$\mathcal{H}_1$: $\boldsymbol{\mu}_d \neq \mathbf{0}_d$.
STEP 2: Sample from testing dataset;
Samples $m = 100$, variables $d = 32$.
STEP 3: Construct statistic

$$\xi = \frac{m(m-d)}{d(m-1)} \overline{\boldsymbol{\mu}}_d^{\mathrm{T}} \mathbf{S}_d^{-1} \overline{\boldsymbol{\mu}}_d. \tag{30}$$

STEP 4: Because $\xi \sim \boldsymbol{F}(d, m-d)$,
P-value p can be computed by

$$p = 1 - \int_0^{\xi'} t^{d/2-1}(1-t)^{(m-d)/2-1} \mathrm{d}t, \tag{31}$$

$$\xi' = \frac{d \cdot \xi}{d \cdot \xi + m - d}\,. \tag{32}$$

STEP 5: Set the confidence α and get conclusion.
If the $p < \alpha$, reject the $\mathcal{H}_0$ and accept $\mathcal{H}_1$; Otherwise accept the $\mathcal{H}_0$.

Hypothesis Test 3 Testing $\boldsymbol{\Sigma}_d = \mathbf{I}_d$?

STEP 1: Define hypotheses,
$\mathcal{H}_0$: $\boldsymbol{\Sigma}_d = \mathbf{I}_d$;
$\mathcal{H}_1$: $\boldsymbol{\Sigma}_d \neq \mathbf{I}_d$.
STEP 2: Sample from testing dataset;
Samples $m = 100$, variables $d = 32$.
STEP 3: Construct statistic

$$\xi = \mathrm{tr}(\mathbf{A}) - m \cdot \ln|\mathbf{A}| - m \cdot d \cdot (1 - \ln m)\,, \tag{33}$$

$$\mathbf{A} = m \cdot \mathbf{S}_d\,. \tag{34}$$

STEP 4: Because $\xi \sim \boldsymbol{\chi}^{\mathbf{2}}(d')$ with $d' = \frac{d(d+1)}{2}$,
P-value p can be computed by

$$p = 1 - \int_0^{\xi^2} \frac{1}{2^{d'/2}\Gamma(d'/2)} t^{d'/2-1} e^{-t/2} \mathrm{d}t. \tag{35}$$

STEP 5: Set the confidence α and get conclusion.
If the $p < \alpha$, reject the $\mathcal{H}_0$ and accept $\mathcal{H}_1$;Otherwise accept the $\mathcal{H}_0$.

S7 Normality of Secret Key Network

We first assess the normality of the SKN's output after training by examining the covariance matrix and mean vector of the outputs. The test dataset is divided into 50 batches, each with 100 images, and the covariance matrix and mean vector of the SKN outputs for each batch are calculated. The average covariance matrix over all batches is visualized in Fig. 6a and resembles an identity matrix. We used kernel density estimation (KDE) to approximate the PDF of each

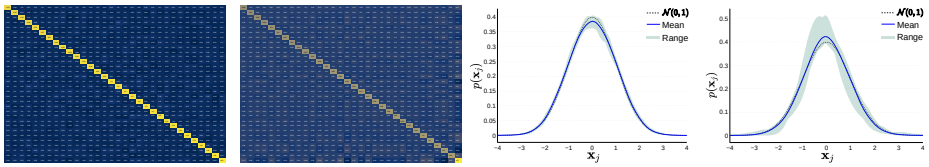

Fig. 6: Qualitative results for assessing normality. The top row shows the average covariance matrix; the bottom row shows the estimated PDF using KDE. (a) and (c) are the results of training w/ $\mathcal{L}_{\text{v}}$, while (b) and (d) are trained w/o $\mathcal{L}_{\text{v}}$.

	w/ $\mathcal{L}_{\text{v}}$		w/o $\mathcal{L}_{\text{v}}$	
	mean	std	mean	std
diagonal entries of covariance matrix (variance terms)	0.9984	0.0200	1.4181	0.2687
off-diagonal entries of covariance matrix (covariance terms)	-0.0002	0.0017	0.0022	0.1103
entries of mean vector	-0.0019	0.0186	-0.0015	0.0471

Table 6: Results on assessing normality of the SKN's output on clean images. Across all batches and output dimensions, we compute the mean and standard deviation (std) of the entries of the output's mean vector and covariance matrix.

dimension of $\mathbf{x}$, and then plot the average, minimum, and maximum of the 32 estimated PDFs in Fig. 6c. The plots show that the individual output dimensions will follow a standard normal distribution. Tab. 6 (left) presents the statistics of the mean vector entries and covariance matrix entries. The diagonal (variance) of the covariance matrix is close to 1, and the off-diagonal (covariance) is close to 0, while the entries of the mean vector are close to zero, indicating that the SKN output closely follows the desired SMVN distribution.

We also evaluated the effect of the proposed variance loss term $\mathcal{L}_{\text{v}}$ by using a similar analysis on an SKN trained without $\mathcal{L}_{\text{v}}$. The results are denoted by "w/o $\mathcal{L}_{\text{v}}$", and are presented in Fig. 6b, Fig. 6d, and Tab. 6 (right). The absence of $\mathcal{L}_{\text{v}}$ during SKN training led to a noticeable deviation from normality in the output vector, as seen by a covariance matrix not resembling an identity matrix, and PDFs significantly deviating from standard normal distributions.

Finally, we conduct three hypothesis tests on a batch of images to examine whether: a) every entry in the output vector follows a normal distribution; b) the mean vector is zero; c) the covariance matrix is identity (see Supp. S6 for details). The results in Tab. 7 (left) show that our SKNs obtain normality in each dimension, and the loss term $\mathcal{L}_{\text{v}}$ is required to achieve this. Furthermore, we conduct similar tests on the output features of DNN0B [36] and SSLWM [10] (Tab. 7 right) and find that they cannot achieve normality since they only use a whitening matrix to transform the feature space of the pre-trained CNN linearly.

Hypo. Test	Ours	Ours w/o $\mathcal{L}_v$	DNN0B [36]	SSLWM [10]
(a) normality	**96.88%**	71.88%	10.16%	3.71%
(b) zero mean	**100%**	100%	84%	0%
(c) identity cov.	**67%**	27%	0%	0%

Table 7: Hypothesis testing on the SKN output distribution. (a) the percentage of entries in the output vector that exhibit normality according to the Kolmogorov-Smirnov test. (b) the acceptance rate of the null hypothesis that the mean vector is zero, and (c) that the covariance matrix is identity.

S8 Discussions about Secure Implementation

We assume that the SKN and SKS are always kept secret, both during watermarking and detection. In practice, our watermark detector could be implemented using a trusted execution environment (TEE), e.g., SGX, or by trusted 3rd party hosts, where the SKN/SKS are not made public.

- For a TEE implementation, the watermarking application would be akin to a biometric authentication application, where the user's data (i.e., SKN and SKS) is stored and compared to a target image (i.e., watermarked image) in the TEE. Thus the SKN and SKS are kept secret, while the verification can be performed by the public within their own TEE. Note that the watermarking application would have to be trusted, e.g., developed and verified by a trusted 3rd party.
- For the implementation with a trusted 3rd party host, both the SKS and SKN would be stored on the 3rd party system. Images would be uploaded to the host's system and the watermark check would be performed by the host, and the results sent back to the uploader.

Even though the SKN and SKS are kept private, an adversary cannot use a fake SKN/SKS to claim ownership of a watermarked image for two reasons. First, it is unlikely that an adversary's newly-generated SKN will match the original SKN (see Case 2 of the §4.6), and thus the adversary cannot pass the ownership verification. Second, our watermark method is robust to attempts to overwrite it with another SKN/SKS. Thus if the adversary adds their own watermark on top of the original watermarked image, then the original watermark will still be present. The original owner can then produce their original watermarked image (without the adversary's watermark) to demonstrate that the chain of processing was from the original watermarked image to the double watermarked image.

Finally, the SKN itself cannot be reverse-engineered from a set of watermarked images (the inputs) without knowing the corresponding SKS (the outputs), which is also kept secret. Also, §4.6 Case 2 shows that for two SKNs trained from different random seeds, one SKN cannot be used to detect the watermark of the other SKN. Thus our framework is secure from reverse engineering by the public.

S9 Watermark Imperceptiblity

Fig. 7 shows a qualitative comparison of the watermarked images by our method and the three comparison models: HiDDeN [25], DNN0B [36], and SSLWM [10]. Both DNN0B and SSLWM, along with our proposed method, are tested under two distinct PSNR values: 32 and 42. The observations from Fig. 7 reveal that at a PSNR of 32, watermarked images generated by HiDDeN exhibit a slight blur yet remain largely identical to their original counterparts. Images processed by DNN0B display minimal noise, whereas SSLWM treated with the ResNet50 approach shows faint line noise. In contrast, our method introduces minor color discrepancies in certain small image areas. Notably, when the PSNR is elevated from 32 to 42, these anomalies are significantly reduced, leading to our method producing images virtually indistinguishable from their originals. Additionally, Fig. 8 shows a broader range of watermarked images generated by our method.

S10 Signatures Generation

We utilize three distinct approaches to create signatures. The first approach is to sample a 32-dimensional vector from the SMVN distribution directly, and we call this as the "naïve signature". In the second approach, the generated signature must adhere to a normal distribution and be orthogonal in any direction to the output vector of SKN on the original image, thereby ensuring the initial non-correlation between SKS and the output vector. We refer to signatures produced in this way as "orthogonal signatures". The third approach also demands that the signature follows the normal distribution properties, but it must have a cosine value greater than 0 with the angle formed with the output vector of the SKN on the original image. We denote these as "aligned signatures".

In our experiment, we evaluate the watermark detection performance of different signature generation methods via their p-values for length, angle, and their combined measures. As detailed in Tab. 8, we present the mean and standard deviation, as well as the percentages of p-values falling below 0.05 and 0.01 thresholds. The detection rates when $\alpha = 0.05$ of the three signature generation methods are similar (all 99%). For a slightly stricter detection using $\alpha = 0.01$, the aligned signatures obtain slightly better performance (98% vs 96%) than the naive or orthogonal signatures. This is attributed to the aligned signatures obtaining generally lower p-values overall.

Furthermore, in practice, some owners may prefer to generate only one SKS and apply it to various images instead of generating multiple SKSs (one SKS for each image). This could result in a decrease in accuracy, due to potential misalignment between the fixed SKS and natural SKN output for the given image. The accuracy can be improved by aligning the given SKS for each image (by possibly flipping the sign of the vector), and then using a 2-sided angle hypothesis test (HT) for either a positive or negative direction (see Tab. 9(c)). Therefore, our model can also meet the needs of owners who desire a single SKS for all of their images.

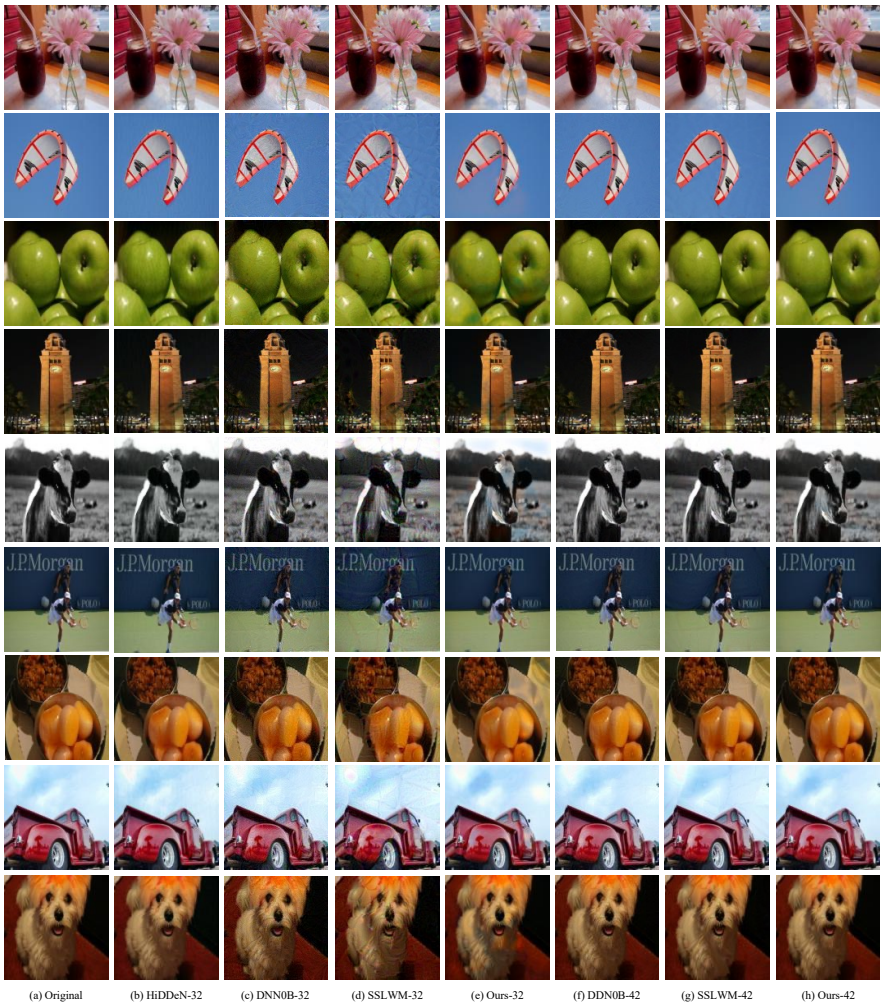

Fig. 7: Qualitative comparison of watermark imperceptibility at two PSNR levels, 32 and 42, with HiDDeN, DNN0B and SSLWM

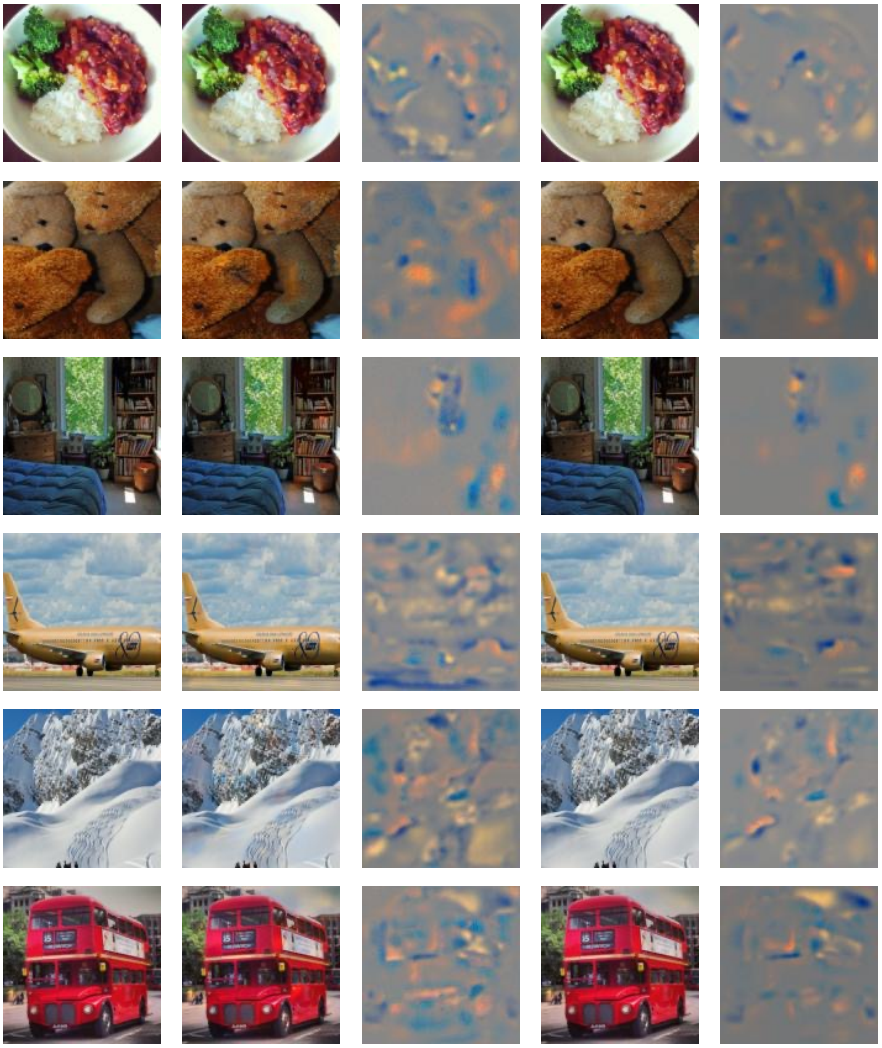

Fig. 8: Examples of watermarking using our proposed method. Watermarking is performed with two specified PSNR target values: PSNR = 32 and PSNR = 42, respectively.

	naïve signatures			
	mean	std	< 0.05	< 0.01
Length	0.0139	0.0277	0.9468	0.6778
Angle	0.0305	0.0953	0.8724	0.7634
Combine	0.0020	0.0119	0.9918	0.9660
	orthogonal signatures			
	mean	std	< 0.05	< 0.01
Length	0.0074	0.0184	0.9788	0.8454
Angle	0.0284	0.0709	0.8536	0.7034
Combine	0.0024	0.0172	0.9914	0.9632
	aligned signatures			
	mean	std	< 0.05	< 0.01
Length	0.0124	0.0291	0.9568	0.7308
Angle	0.0109	0.0376	0.9358	0.8514
Combine	**0.0011**	**0.0091**	**0.9948**	**0.9828**

Table 8: Watermark detection performance for different signature generation methods. The mean and standard deviation (std) of the detector's p-values are shown, as well as the percentage of p-values below significance level $\alpha = 0.05$ and $\alpha = 0.01$.

	alignment	per image SKS storage	< 0.05	< 0.01
(a) one SKS for each image	Yes	1 SKS per image	99.5%	98.3%
(b) single SKS for all images	Yes	1 bit per image	98.3%	95.2%
(c) w/ two-sided HT	No	None	98.6%	96.4%

Table 9: Comparison between one-side and two-sides tail hypothesis testing of angle metric when only a single SKS is used for multiple images. The table shows the percentage of p-values below significance level α= 0.05 and α = 0.01.

S11 Generalization Ability

To claim that our model has a good generalization ability, we train it on MSCOCO [22], and then use it to embed watermarks into unseen real images (ImageNet [7]) and AI-generated images (InstructPix2Pix [3]). Tab. 10 shows that our method can achieve a high-level detection accuracy, meaning that it has good generalization.

S12 Runtime Comparison

Tab. 11 shows the average runtime for watermarking and detection over 100 images. Our method is faster than DNN0B [35] and SSLWM [10] because we use a smaller backbone to map images into vectors and watermarking is performed completely in GPU, while DNN0B requires a CPU-based iteration for its proposed "RMAC" module.

Dataset	< 0.05	< 0.01
ImageNet	99.1%	97.0%
InstructPix2Pix	99.4%	99.6%

Table 10: The detection rate of our method for watermarking images from 2 unseen datasets: ImageNet and InstructPix2Pix. The table shows the percentage of p-values below significance level α= 0.05 and α = 0.01.

method	backbone	watermarking (s)	detection (s)
DNN0B [35]	VGG19	1.18	0.013
SSLWM [10]	ResNet50	7.13	0.013
Ours	ResNet18	**0.67**	**0.005**

Table 11: Runtime comparisons (average over 100 images).

S13 Target Length Analysis

In this experiment, we discuss how different target lengths t_l in Eq. (4) would affect the detection of watermarks. In Tab. 12, decreasing t_l will result in failed Hypothesis testing for length (HT4L), while increasing t_l will make the watermark generation biased towards smaller p-values for the HT4L. Therefore, we choose and fix $t_l = 63$ in other all experiments to generate watermarks. Importantly, the length value 63 is corresponding to the significance of p=0.01 from the HT4L.

t_l	$p < 0.05$			$p < 0.01$		
	Length	Angle	Combined	Length	Angle	Combined
20	0.0%	98.8%	97.0%	0.0%	97.0%	94.2%
63	95.7%	93.6%	99.5%	73.0%	85.1%	98.3%
100	99.7%	49.6%	99.9%	98.7%	22.9%	98.7%

Table 12: Detection accuracy for different target lengths t_l in Eq. (4).