

## A PROOF OF THEOREM 1

We show that if the sampler is distribution preserving, a distinguisher  $\mathcal{D}$  is nonce respecting and the watermarking scheme uses an IND\$-CPA secure cryptographic unit  $C$ , the watermarking scheme is undetectable under IND\$-CPA.

Formally, we get

$$\begin{aligned} \Pr[\text{PrivK}_{\mathcal{D}, \text{WM}}^{\text{IND\$-CPA}}(\lambda) = 1] &= \Pr[b = 1] \Pr[\mathcal{D}(x) = 1] + \Pr[b = 0] \Pr[\mathcal{D}(x) = 0] \\ &= \frac{1}{2} \Pr[\mathcal{D}(\text{WM}_{k, \mathcal{M}}(m, \eta, \pi)) = 1] + \frac{1}{2} \Pr[\mathcal{D}(\mathcal{G}(\mathcal{S}(r))) = 0] . \end{aligned} \quad (1)$$

On a real random input (second term),  $\mathcal{D}$  cannot obtain any information and therefore just guesses with probability  $\frac{1}{2}$ . On a watermarked input (first term),  $\mathcal{D}$  needs to recognize the output of IND\$-CPA secure cryptographic unit for an unknown key, which is hard by assumption. Therefore, the right hand side gets

$$= \frac{1}{2} \left( \frac{1}{2} + (q+1) \text{negl}(\lambda) \right) + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2} + \text{negl}(n) . \quad (3)$$

If we consider a distinguisher  $\mathcal{D}'$  that is **not nonce-respecting**, there is an obvious attack. First,  $\mathcal{D}'$  chooses  $m^*$ ,  $\eta^*$ , and  $\pi^*$  and passes this as  $(m_1, \eta_1, \pi_1)$  and as  $(m, \eta, \pi)$ .  $\mathcal{D}'$  obtains  $x^{(m_1)}$  and  $x$ . Next,  $\mathcal{D}'$  uses inversion and the inverse sampler to recover the ciphertexts  $c_1 = \mathcal{S}^{-1}(\mathcal{I}(x^{(m_1)}))$  and  $c = \mathcal{S}^{-1}(\mathcal{I}(x))$ . Idealized, if they both match,  $\mathcal{D}'$  has found that this image is watermarked and outputs 1, otherwise 0. Usually, they will not be exactly the same due to error in the recovery. However, they are close enough such that recovery is possible, i.e.,  $c \approx c_1$ . We compute the probability for the distinguisher  $\mathcal{D}'$  and find that

$$\Pr[\text{PrivK}_{\mathcal{D}', \text{WM}}^{\text{IND\$-CPA}}(\lambda) = 1] = \Pr[b = 1] \Pr[\mathcal{D}'(x) = 1] + \Pr[b = 0] \Pr[\mathcal{D}'(x) = 0] \quad (4)$$

$$= \frac{1}{2} \Pr[\mathcal{D}'(\text{WM}_{k, \mathcal{M}}(m, \eta, \pi)) = 1] + \frac{1}{2} \Pr[\mathcal{D}'(\mathcal{G}(\mathcal{S}(r))) = 0] \quad (5)$$

$$= \frac{1}{2} \cdot 1 + \frac{1}{2} (1 - \text{negl}(n)) \quad (6)$$

$$= 1 - \frac{1}{2} \text{negl}(n) . \quad (7)$$

Clearly,  $\mathcal{D}'$  has a non-negligible success probability—which is in fact close to 1 even with just one watermarked image—and can therefore easily distinguish between an unwatermarked image and a watermarked one.

## B GENERAL DEFINITION OF DISTRIBUTION-PRESERVING WATERMARKS

We provide a game based definition of WM-IND-CPA, which equivalent to the is the undetectability notion used by Zhao et al. (2025) and Gunn et al. (2025).

The game is depicted in Figure 4 and works as follows: A distinguisher is either given access to a plain image generation model  $\mathcal{M}$  or to a watermarking scheme  $\text{WM}_{k, \mathcal{M}}$  depending on a coin flip by the game. It can request up to  $q$  arbitrary adaptive messages and prompts. If it has access to  $\mathcal{M}$ , the messages are ignored. WM is called undetectable if there does not exist a distinguisher that can win the game  $\text{PrivK}_{\mathcal{D}, \text{WM}}^{\text{WM-IND-CPA}}(\lambda)$  with more than negligible advantage. Formally, we require that for every polynomial time distinguisher  $\mathcal{D}$ ,  $\Pr[\text{PrivK}_{\mathcal{D}, \text{WM}}^{\text{WM-IND-CPA}}(\lambda)] = \frac{1}{2} + \text{negl}(\lambda)$ .

This definition is an equivalent formulation of Zhao et al. (2025).

While this definition is more generic, it does not explicitly expose the nonce of a watermarking scheme to our analysis, which is why we choose to introduce nonce-based IND\$-CPA.

## C REDUCTION FROM IND\$-CPA TO WM-IND-CPA UNDETECTABILITY

We now consider the relation between WM-IND-CPA (see Appendix B) and IND\$-CPA.

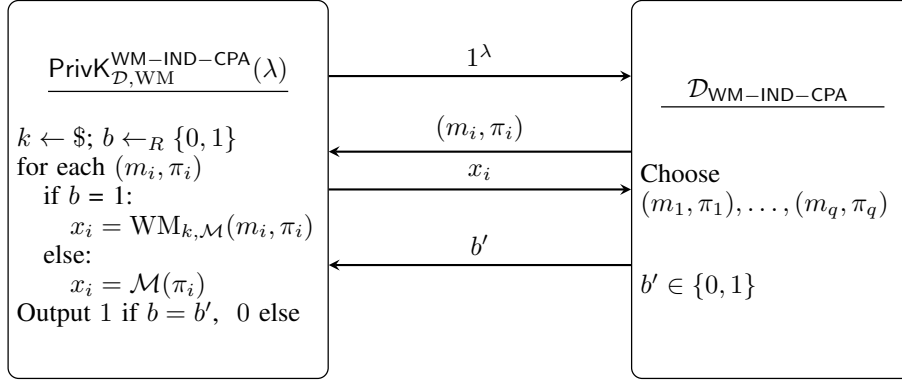


Figure 4: WM-IND-CPA undetectability.

We show that IND\$-CPA is at least as strong as the WM-IND-CPA. We assume that the unwatermarked model actually uses truly random initial latents  $z_T$ . Otherwise all watermarking schemes proposed so far are detectable. Under this assumption, distinguishing outputs from  $\mathcal{M}$  is equivalent to distinguishing images from random seeds (see "else case" in Figure 2). We start with identifying the WM-IND-CPA with  $q$ -IND\$-CPA by making the following observation. It does not matter whether a distinguisher which can request an arbitrary polynomial amount of images requests these upfront in the oracle phase or all of them in the query phase. It is equivalent to a distinguisher that requests up to  $2q$  images.

It is a well known fact in cryptography, that such a distinguisher has an advantage of  $2q$  times over a distinguisher which just has access to one image in the oracle phase Katz & Lindell (2015). We achieved the canonical IND\$-CPA game which does not require nonces (we write c-IND\$-CPA). This undetectability notion does contain an oracle phase to query watermarked images and a challenge phase in which a distinguisher needs to tell for one challenge image whether it was watermarked or not.

The final step reduces our nonce-based undetectability to c-IND\$-CPA. This is straight forward. All the reduction needs to do is for every query simulate a nonce respecting choice of nonces. We assume a strategy that chooses them distinct by some deterministic scheduling. In this case, the reduction perfectly simulates the c-IND\$-CPA such that the advantage transfers.

To summarise, we have shown that nonce-based IND\$-CPA undetectability is at least as strong as WM-IND-CPA undetectability under the assumption that the model actually chooses perfectly random initial seeds.

## D PRCW IS NOT IND\$-CPA UNDETECTABLE

Next, we are going to show that PRCW is not undetectable under nonce-based IND\$-CPA. More precisely we show that a nonce-respecting distinguisher  $\mathcal{D}$  can break the pseudorandomness if it chooses nonces distinct but non-random.

In order to do so, we first provide a more detailed definition of a PRCW code word. Then, we provide an attack on PRCW as proof. Finally, we outline its consequences on both PRCW and GS++.

**PRCW** A PRC extends a normal LDPC code first by encoding a concatenation of a message  $m$ , some check bits  $\beta$  and a nonce  $\eta$ . The check bits are used to ensure a desired FPR on detecting watermarks in general. They are always the same and matched upon watermark verification. The nonce ensures that if the same  $m$  is encoded twice, its values differ. Furthermore, it contains a sparse binary error vector  $e \sim \text{Bin}(|c|, \delta)$  for a small  $\delta$ , which ensures that recovering  $G$  from some  $c$  is computationally hard. Finally, it contains a random but fixed bit string  $\text{otp}$  of length  $|c|$  that

ensures a technical independence property. As an encoding, we get the following equation over bits.  
 $c = G(\beta||m||\eta)^T \oplus e \oplus \text{otp}$ .

**Proof of Theorem 3** The attack works as follows: First,  $\mathcal{D}$  requests code words for a fixed message  $m$  (e.g.  $m = 0$ ) and for four nonces  $\eta_1, \dots, \eta_4$  with the property that  $\bigoplus_{i=1}^4 \eta_i = 0$ . It requests the first three in the first phase to obtain the valid PRC code words for these and the last one in phase two. Afterwards,  $\mathcal{D}$  XORs all the received codewords  $c_i$ . If all four of them are code words, the resulting bitstring should only contain a small amount of ones. Otherwise, it should be fairly large.

In more detail, we assume that  $b = 1$  and therefore all four code words are valid. We end up with the XOR of the noise vectors  $e_i$ . Not that for any constant, the XOR of an even number of it is zero, e.g.  $\bigoplus_{i=1}^4 \text{otp} = 0$ .

$$\begin{aligned} \bigoplus_{i=1}^4 c_i &= \bigoplus_{i=1}^4 G(\beta|\eta_i|m)^T \oplus e_i \oplus \text{otp} \\ &= G\left(\bigoplus_{i=1}^4 (\beta|\eta_i|m)^T\right) \oplus \bigoplus_{i=1}^4 e_i \\ &= G \cdot 0 \oplus \bigoplus_{i=1}^4 e_i = \bigoplus_{i=1}^4 e_i \end{aligned}$$

If code word is random, we expect about half of the bits to be one. If not, it should be the XOR of four independent noise vectors from  $\text{Bin}(|c|, \delta)$ , which contain less than  $4|c|\delta$  ones on expectation, which far from  $\frac{n}{2}$ . How far depends on the specific choice of parameters for  $\delta$  and  $|c|$ , however a possible configuration for our experimental section (see section 4.2 is  $|c| = 2^{14} = 16,384$  and  $\delta = 0.0081$ . It is obvious to see that the probability of  $\mathcal{D}$  being able to distinguish is far from negligible and more close to one.  $\square$

**Consequences for PRCW and GS++** Our attack strategy is not valid if  $\mathcal{D}$  chooses nonces uniformly at random, which is how PRCW was proposed by Gunn et al. (2025). In this case, the best strategy is a meet in the middle attack, but the nonce length of a PRC is chosen such that it is infeasible in accordance with the security parameter  $\lambda$ . As a consequence, PRCW is undetectable if nonces  $\eta$  are chosen uniformly at random every time, but does not fulfil the stronger notion of IND\$-CPA security.

This extends to GS++. There, a PRC is used but without any form of check bits  $\beta$ , additional nonce  $\eta$  or otp. Instead, its encoded message is the nonce of the entire scheme. However, this nonce is shorter. It is used as a salt in a hash function together with the secret key  $k$  to expand it to a full pair of stream cipher key and nonce. If at this point a pseudorandom expansion function like SHAKE-128 (Bertoni et al., 2009) is used, the nonce is actually pseudorandom at each step. This gives IND\$-CPA.

## E IMPACT OF WATERMARK CONFIGURATIONS ON IMAGE DIVERSITY

To show the impact of different watermarking schemes and their configurations on image diversity, we evaluate two metrics empirically. First, we calculate the FID (Heusel et al., 2017) between five non-overlapping sets of 2.000 watermarked and unwatermarked images taken from a total of 10.000 images each and report mean and standard deviation. Generation prompts are taken randomly from the Stable Diffusion Prompts dataset<sup>7</sup> and are aligned in each set, meaning images in both sets are generated from the same list of prompts. Second, we calculate the mean pairwise LPIPS (Zhang et al., 2018) scores between pairs of images within sets of 100 images 10 times, generated from 10 different prompts and report mean and standard deviation from the 10 runs. Like in the main experiments, we use Stable Diffusion V2.1, the DPM sampler<sup>8</sup>, as well as default guidance scale (7.5) and inference steps (50).

<sup>7</sup>Stable Diffusion Prompts

<sup>8</sup>DPM Solver Multistep Scheduler

**Results** All watermarking schemes show FID and LPIPS scores comparable to unwatermarked generation with the exception of Gaussian Shading setup with same keys and same nonces. For this setup, the FID scores are higher and LPIPS scores are lower, indicating more similar outputs which is due to similar initial latents

	FID ↓	LPIPS ↑
No Watermark	25.1281 $\pm$ 0.3018	0.5921 $\pm$ 0.0333
GS Reference	25.0959 $\pm$ 0.2396	0.5946 $\pm$ 0.0315
GS - New key, new nonce	25.0658 $\pm$ 0.2101	0.5945 $\pm$ 0.0318
GS - Same key, same nonce	25.4393 $\pm$ 0.3335	0.5861 $\pm$ 0.0321
GS++	25.0975 $\pm$ 0.1872	0.5932 $\pm$ 0.0313
PRCW	25.1470 $\pm$ 0.1485	0.5950 $\pm$ 0.0319

Table 2: Experiments on image diversity under different watermarking schemes and settings.

## F RANDOMNESS CHECK

We start with the following observation: by inversion, we recover a long bitstring  $\hat{c}$ . If it gets decrypted with a wrong nonce, we will obtain a pseudorandom bitstring. This is the case because  $\text{PRNG}(k, \eta')$  is a new pseudorandom sequence, so  $\tilde{s} = \hat{c} \oplus \text{PRNG}(k, \eta')$  is a new encryption of  $\hat{c}$  instead of a decryption. Since our stream cipher is secure, this is pseudorandom. Hence, the amount of ones in each strip of length  $\rho$  is binomially distributed according to  $\text{Bin}(\rho, \frac{1}{2})$ . If we chose the correct nonce, we will get a repetition code where each bit is repeated  $\rho$  times (see Section 2.2) up to some error.

We therefore observe the amount of ones in each block of  $\rho$  consecutive bits and determine the probability, that this was generated by a binomial distribution with  $\text{Bin}(\rho, \frac{1}{2})$  (random bits) or by a biased one (potentially an encoded bit). We choose a standard test power of 80%. There are  $|m|$  blocks in  $\tilde{s}$ . If it was indeed random, we expect each individual test to fail with a probability of 20%. We examine whether more tests failed than one would expect from  $|m|$  samples from  $\text{Bin}(\rho, \frac{1}{2})$  by performing an additional one-sided test over all. We set our FPR to 0.01, since false positives are costly in runtime, but false negatives would imply a missed detection. If we consider the  $\tilde{s}$  non-random, we assume that we actually found the correct nonce  $\eta$  and continue with the verification. This gives the runtime of  $\mathcal{O}(M + N)$ .

Our method involves a trade-off: If we choose FPR too small, we essentially lower the error correction capabilities of the underlying repetition code and make GS less robust. However, if we choose FPR too large, we have higher runtime. We empirically found 0.01 a valid choice.

## G FULL EXPERIMENTAL DETAILS

All experiments were conducted on a server with 4 Nvidia A40 GPUs, 32 AMD EPYC 7282 16-Core Processors, and 500 GB RAM, running Ubuntu 20.04.6 LTS. The exact Python version and package requirements can be found in our project repository (<https://anonymous.4open.science/r/submission-F475/>).

Across all experiments, the image sizes and the latent sizes are set to  $512 \times 512$  and  $4 \times 64 \times 64$ , respectively.

For all watermarking schemes, we set a message length of 256 and false positive rate of  $10^{-6}$ .

The exact parameters used for individual schemes are as follows:

- **PRCW**
  - sparsity  $t$ : 3
  - error probability  $\delta$ : 0.0081
  - length of nonce  $\eta$ : 39 bits

- length of check bits  $\beta$ : 30 bits
- Decoding iterations: 8
- **GS++**
  - sparsity  $t$ : 7
  - error probability  $\delta$ : 0.0176
  - length of nonce  $\eta$  (ChaCha): 96 bits
  - length of nonce in PRC (PRC): 256 bits
  - Hash function: SHAKE-128
- **GS**
  - Key length: 256 bits
  - Nonce length: 96 bits

We furthermore set both denoising steps and inversion steps to 50, both for regular DDIM inversion as well as exact inversion.

All prompts used for experiments are taken from the Stable Diffusion Prompts dataset<sup>9</sup>.

## H EXAMPLE IMAGES

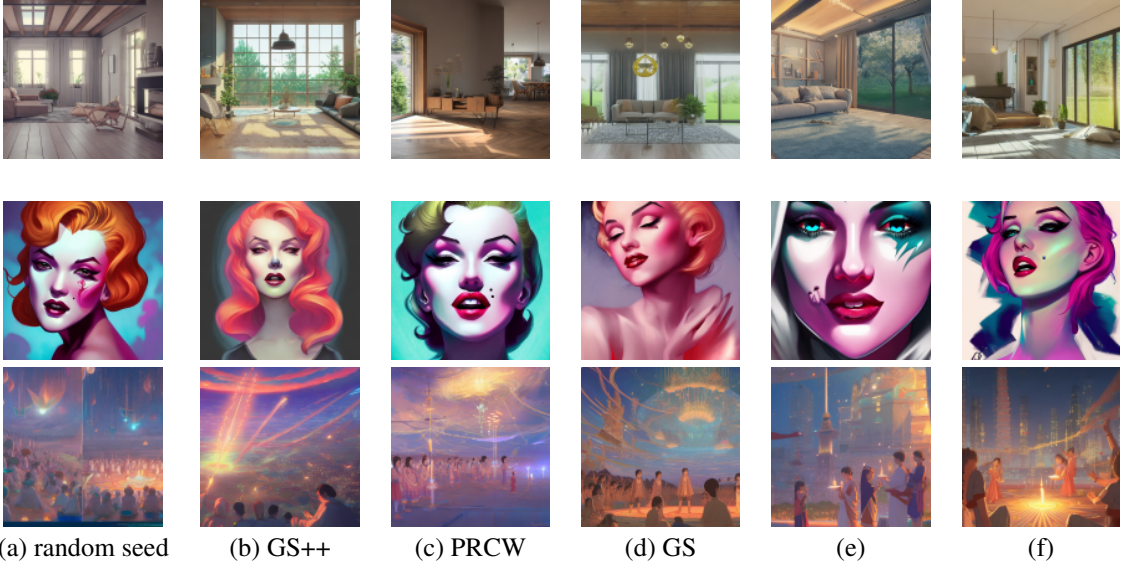


Figure 5: Comparison of different watermarking schemes. (e) is GS in the same key, same nonce setting. (f) is GS in the new key, new nonce setting.

<sup>9</sup>Stable Diffusion Prompts