

به نام خدا



درس امنیت کامپیوتری

تمرین سری اول

مدرس درس:

جناب آقای دکتر دیانت

تهیه شده توسط:

حوریه سبزواری، الناز رضایی

تاریخ ارسال: ۱۴۰۱/۰۱/۲۵

سوال ۱ :

پیاده‌سازی یک روش نهان‌کاوی به عنوان آشکارسازی بر نهان‌نگاری به روش LSB . یعنی فرض کنید که ما با روش LSB عملیات نهان‌نگاری را انجام دادیم، شما باید یک روش نهان‌کاوی به منظور تشخیص آن پیاده‌سازی کنید.

پاسخ ۱ :

کد موردنظر برای این بخش را در فایل Q1.ipynb پیاده‌سازی کردیم. برای اینکه یک تصویر داشته باشیم تا بتوانیم عملیات نهان‌کاوی را روی آن انجام دهیم، در کدمان یک بخش برای encode کردن تصویر، و یک بخش برای decode آن قرار دادیم. سپس تصویر موردنظر را با پیام دلخواه encode کردیم. همانطور که می‌دانیم در این روش برای encode کردن تصویر، پیام در بیت‌های LSB پیکسل‌ها قرار می‌گیرند؛ چرا که این بیت‌ها کم‌ارزش‌ترین بیت‌ها هستند و تغییر در آن‌ها، تفاوت چندانی در تصویر ایجاد نمی‌کند. در تابع Encode، این اتفاق می‌افتد.



شکل ۱: تصویر اصلی قبل از عملیات encoding

حال پس از انتخاب تصویر موردنظر برای عملیات نهان‌کاوی، برنامه را run کرده و گزینه encode را انتخاب می‌کنیم تا تصویر رمزگذاری شده برای مرحله بعد را آماده نماییم. همچنین کلمه Security را به عنوان پیام پنهان به برنامه می‌دهیم. در انتهای نیز آدرس مقصد برای ذخیره عکس

شده را می‌دهیم.

```
Encode the input image

if __name__ == "__main__":
    Stego()

--Welcome to $t3g0--
1: Encode
2: Decode
1
Enter Source Image Path
1.png
Enter Message to Hide
Security
Enter Destination Image Path
2.png
Encoding...
RGBA
Image Encoded Successfully
```

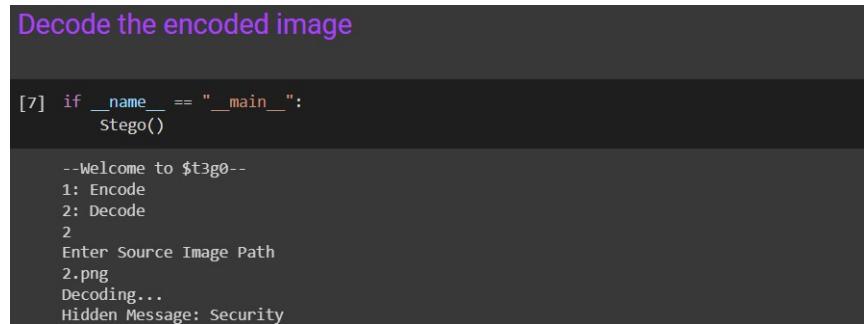
شکل ۲: نحوه encode کردن تصویر

حال تصویر رمزنگاری شده را بررسی می‌کنیم. همانطور که در شکل زیر نیز می‌بینید، تصویر رمزنگاری شده تفاوت چندانی با تصویر اصلی ندارد؛ به گونه‌ای که تغییرات با چشم غیرمسلح قابل مشاهده نیست. یکی از دلایل این پدیده می‌تواند کوتاه بودن متن پیام باشد. عبارت \$t3g0 به کار رفته هنگام ذخیره‌سازی تصویر نیز برای تشخیص انتهای پیام می‌باشد.



شکل ۳: تصویر encode شده

حال از تابع Decode برای نهان‌کاوی تصویر encode شده استفاده می‌نماییم. همانطور که در شکل زیر مشاهده می‌نمایید، برنامه به درستی کلمه Security که به عنوان پیام پنهان به تابع Encode داده بودیم را برای ما چاپ می‌کند.



```
Decode the encoded image

[7] if __name__ == "__main__":
    stego()

--Welcome to $t3g0--
1: Encode
2: Decode
2
Enter Source Image Path
2.png
Decoding...
Hidden Message: Security
```

شکل ۴: نحوه decode کردن تصویر

Reference

سوال ۲:

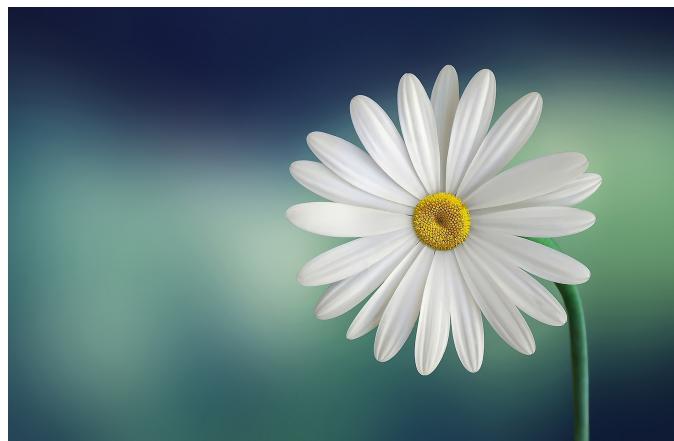
پیاده‌سازی یک روش نشان‌گذاری از نوع Visible و غیرشکننده . البته بدیهی است که ابتدا باید تحقیق کنید و یک روش نشان‌گذاری از نوع Visible و غیرشکننده را پیدا کنید و سپس آن را پیاده‌سازی کنید.
پیاده‌سازی باید به زبان‌های C++ یا Python باشد.

پاسخ ۲:

واترمارکینگ فرآیند جاسازی یک سیگنال دیجیتال منحصر به فرد و قابل شناسایی است که به عنوان واترمارک شناخته می‌شود، در یک شی چندرسانه‌ای مانند تصویر، ویدئو یا فایل صوتی. واترمارک می‌تواند برای تأیید صحت یا مالکیت شی یا برای ردیابی توزیع و استفاده از آن استفاده شود. واترمارک‌ها اغلب به گونه‌ای طراحی می‌شوند که برای چشم یا گوش انسان نامحسوس باشند تا در کیفیت یا محتوای رسانه تداخلی ایجاد نکنند. تکنیک‌های مختلفی برای واترمارک وجود دارد، از جمله واترمارک‌های قابل مشاهده و غیرقابل مشاهده و واترمارک‌های غیرشکننده و شکننده.

واترمارک قابل مشاهده: نمونه ای از واترمارک قابل مشاهده زمانی است که یک عکاس آرم یا اطلاعات حق چاپ خود را به یک تصویر اضافه می کند. این باعث می شود که تصویر متعلق به آنها باشد و بدون اجازه نمی توان از آن استفاده کرد.

واترمارک غیرشکننده: واترمارک غیرشکننده نوعی از واترمارک است که در آن واترمارک همچنان قابل تشخیص است حتی اگر رسانه به نحوی تغییر کرده باشد. به عنوان مثال، اگر هنرمندی بخواهد از موسیقی خود در برابر دزدی دریابی محافظت کند، می تواند یک واترمارک قوی در فایل صوتی جاسازی کند. حتی اگر کسی سعی کند فایل را با تغییر میزان بیت، فرمت یا طول تغییر دهد، واترمارک همچنان قابل تشخیص است. این باعث می شود که برای ردیابی توزیع رسانه های دارای حق چاپ مفید باشد. هدف از واترمارک غیرشکننده این است که اطمینان حاصل شود که حتی اگر رسانه به نحوی تغییر کرده باشد، مانند فشرده سازی، برش، تغییر اندازه، چرخش، اضافه کردن نویز هنوز واترمارک قابل شناسایی است.



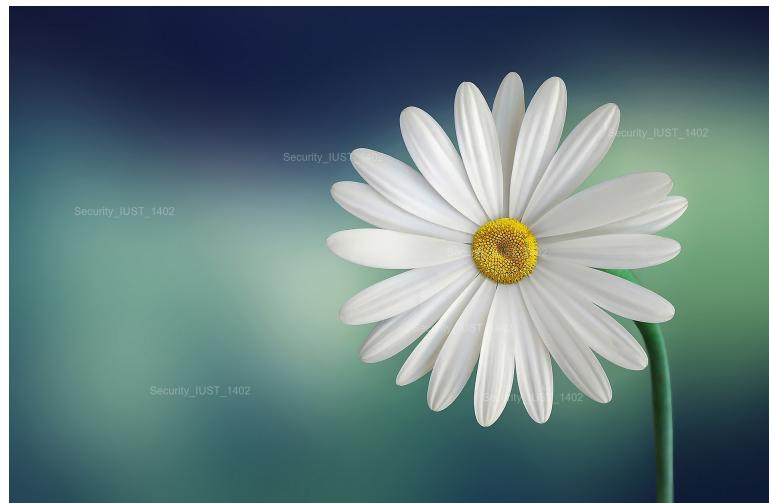
شکل ۵: تصویر اصلی

در برخی موارد، واترمارک های غیرشکننده نیز قابل مشاهده هستند. به عنوان مثال، یک شبکه تلویزیونی ممکن است یک لوگوی قابل مشاهده به پخش خود اضافه کند و در عین حال یک واترمارک غیرشکننده را نیز تعییه کند که برای بینندگان قابل مشاهده نیست. در این سوال ما با استفاده از کتابخانه های PIL و OpenCV یک جمله را به عنوان واترمارک به صورت تصادفی در ۷ نقطه‌ی تصویر اصلی با شفافیت کم قرار دادیم.

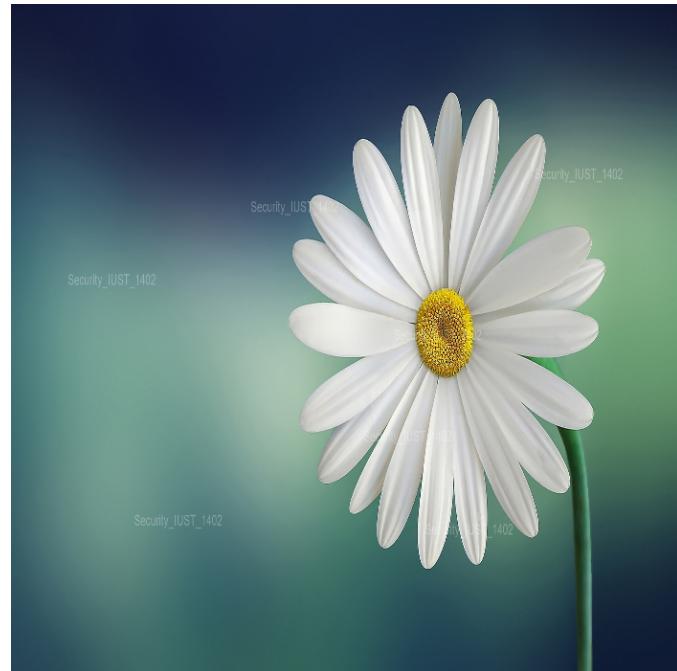


شکل ۶: تصویر watermark شده

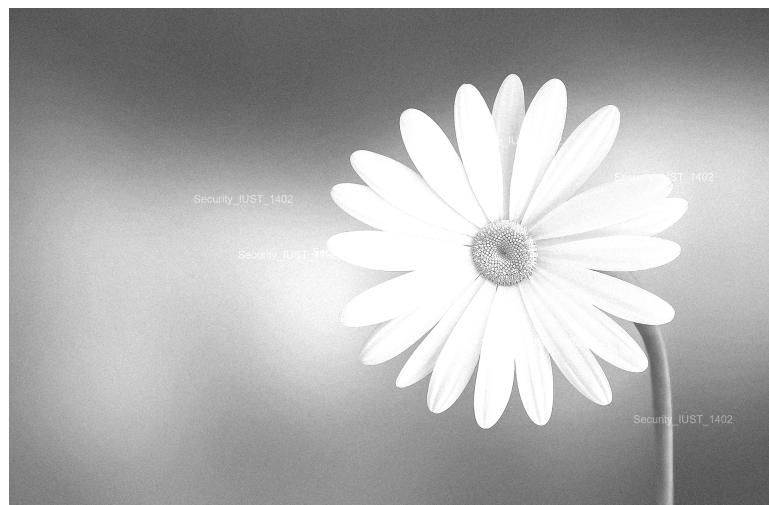
سپس تصویر واترمارک شده را تحت چهار عملیات چرخش، برش، تغییر اندازه و افزودن نویز بررسی کردیم. همانطور که مشاهده می‌کنید، واترمارک‌ها از بین نرفته‌اند و تشانده‌های غیرشکننده و قابل مشاهده بودن آن است.



شکل ۷: اعمال چرخش بر روی تصویر watermark شده



شکل ۸: اعمال تغییر اندازه بر روی تصویر watermark شده



شکل ۹: اعمال نویز بر روی تصویر watermark شده



شکل ۱۰: اعمال برش بر روی تصویر watermark شده

Reference