دانشگاه آزاد اسلامی

واحد تبریز

دانشکده فنی و مهندسی ۱

گروه کامپیوتر

پایان‌نامه جهت اخذ مدرک کارشناسی

در رشته فناوری اطلاعات

عنوان:

پنهان‌نگاری در عکس رنگی به روش اصلاح مقادیر پیکسل‌ها بر مبنای باقیمانده تقسیم

استاد راهنما:

دکتر علی اصغر پورحاجی کاظم

نگارش:

هیوا امیری

خرداد ۱۳۹۳

به نام یکتا

دانشگاه آزاد اسلامی
واحد تبریز
دانشکده فنی و مهندسی ۱
گروه کامپیوتر

پایان‌نامه جهت اخذ مدرک کارشناسی
در رشته فناوری اطلاعات

عنوان:

پنهان نگاری در عکس رنگی به روش اصلاح مقادیر پیکسل‌ها بر مبنای باقیمانده تقسیم

استاد راهنما:

دکتر علی اصغر پورحاجی کاظم

نگارش:

هیوا امیری

خرداد ۱۳۹۳

نمره

به عدد	به حروف	امضای استاد پروژه

چکیده

متن، تصویر، صدا و ویدئو را می‌توان به صورت داده‌های دیجیتال بیان کرد. فراگیری فزاینده و رشد سریع استفاده از اینترنت، انسان‌ها را به سوی جهان دیجیتال و ارتباط از طریق داده‌های دیجیتال سوق داده است. هرجا سخن از ارتباط به میان آید، مسأله امنیت کانال ارتباطی نیز مطرح می‌گردد. در واقع، در یک کانال ارتباطی، پنهان‌نگاری روشی از ارسال اطلاعات محرمانه است به نحوی که وجود خود کانال در این ارتباط مخفی بماند. روش پنهان‌نگاری رایانه‌ای روشی از پنهان‌نگاری است که امنیت اطلاعات در رسانه دیجیتال را فراهم می‌سازد و هدف آن درج و ارسال پیام محرمانه از طریق رسانه دیجیتال است بگونه‌ای که هیچ ظنی مبنی بر ارسال اطلاعات برانگیخته نشود. پیام محرمانه می‌تواند به صورت یک تصویر یا متن یا سیگنال کنترل و خلاصه هر چیزی که بتواند بصورت بیتی از صفر و یک قابل بیان باشد، مورد توجه است. امکان دارد پیام محرمانه قبل از پنهان‌نگاری تحت مراحل فشرده‌سازی یا رمزنگاری نیز قرار گیرد. پنهان‌نگاری دارای سه پارامتر اساسی ظرفیت داده قابل تزریق، مقاومت و غیرقابل تشخیص بودن است. بدون شک بیشینه سازی همگی این پارامترها امکان‌پذیر نیست و تنها باتوجه به کاربرد باید مصالحه‌ای بین این پارامترها ایجاد کرد .برای پنهان‌نگاری روش‌ها و الگوریتم‌های متفاوتی ارائه شده است، ساده‌ترین و ابتدایی‌ترین روش انجام این امر، روش کم‌ارزش‌ترین‌بیت نام دارد که با جایگذاری داده مخفی در بیت‌های کم ارزش مقادیر رنگی هر پیکسل کار را انجام می‌دهد. برخی روش‌ها از دو پیکسل رنگی برای مخفی سازی یک رقم استفاده می‌کنند در حالیکه در روش اصلاح مقادیر پیکسل‌ها، برای مخفی سازی یک رقم تنها از پیکسل استفاده می‌کند.

کلید واژه‌ها

پنهان‌نگاری، استگانوگرافی، پنهان‌یابی، پردازش تصویر

فهرست مطالب

فهرست شکل

فهرست جدول

۱) فصل اول : مقدمه

۱-۱) پردازش تصویر دیجیتالی چیست؟

بینایی از حس‌های پیشرفته انسان است، بنابراین شگفت‌آور نیست که تصاویر نقش بسیار مهم و منحصربه‌فردی را در ادراک آدمی بازی می‌کند. بر خلاف انسان‌ها که بینایی‌شان منحصر به بازه مرئی طیف امواج الکترومغناطیسی است، ماشین‌های تصویربرداری تقریبا تمامی طیف الکترومغناطیسی را که از اشعه گاما تا امواج رادیویی گسترده است، می‌پوشاند و می‌تواند روی تصاویری عمل کند که منبع آن‌ها برای انسان نامأنوس است. بنابراین پردازش تصویر دیجیتالی کاربردهای گسترده و متنوعی را دربر می‌گیرد. گاهی پردازش تصویر، عملیاتی تعریف می‌شود که ورودی و خروجی آن تصویر باشد. از طرف دیگر، حوزه‌هایی وجود دارد که مقصد نهایی آن استفاده از رایانه‌ها برای تقلید حس بینایی انسان است. مثلا در بینایی ماشین یادگیری، استنباط، نتیجه‌گیری و انجام عملیات بر اساس ورودی‌های بصری انجام می‌شود. این حوزه‌ها شاخه‌ای از هوش مصنوعی هستند که می‌خواهند از هوش انسان تقلید کنند و در گام‌های اولیه‌ی پیشرفت قرار دارند. حوزه تحلیل تصویر مابین پردازش تصویر و بینایی ماشین واقع شده است.

پردازش تصاویر امروزه بیشتر به موضوع پردازش تصویر دیجیتال گفته می‌شود که شاخه‌ای از دانش رایانه است که با پردازش سیگنال دیجیتال که نماینده تصاویر برداشته شده با دوربین دیجیتال یا پویش شده توسط پویشگر هستند سر و کار دارد.پردازش تصاویر دارای دو شاخه عمده بهبود تصاویر و بینایی ماشین است. بهبود تصاویر دربرگیرنده روش‌هایی چون استفاده از فیلتر محوکننده و افزایش تضاد برای بهتر کردن کیفیت دیداری تصاویر و اطمینان از نمایش درست آنها در محیط مقصد(مانند چاپگر یا نمایشگر رایانه)است، در حالی که بینایی ماشین به روش‌هایی می‌پردازد که به کمک آنها می‌توان معنی و محتوای تصاویر را درک کرد تا از آنها در کارهایی چون رباتیک و محور تصاویر استفاده شود.

در معنای خاص آن پردازش تصویر عبارتست از هر نوع پردازش سیگنال که ورودی یک تصویر است مثل عکس یا صحنه‌ای از یک فیلم. خروجی پردازشگر تصویر می‌تواند یک تصویر یا یک مجموعه از نشان‌های ویژه یا متغیرهای مربوط به تصویر باشد. اغلب تکنیک‌های پردازش تصویر شامل برخورد با تصویر به عنوان یک سیگنال دو بعدی و بکار بستن تکنیک‌های استاندارد پردازش سیگنال روی آنها می‌شود. پردازش تصویر اغلب به پردازش دیجیتالی تصویر اشاره می‌کند ولی پردازش نوری و آنالوگ تصویر هم وجود دارند.

۱-۱-۱) عملیات اصلی در پردازش تصویر

۱. تبدیلات هندسی: همانند تغییر اندازه، چرخش و...
۲. رنگ: همانند تغییر روشنایی، وضوح و یا تغییر فضای رنگ
۳. ترکیب تصاویر: ترکیب دو و یا چند تصویر
۴. فشرده سازی پرونده: کاهش حجم تصویر
۵. ناحیه بندی پرونده: تجزیه تصویر به نواحی با معنی
۶. بهبود کیفیت پرونده: کاهش نویز، افزایش کنتراست، اصلاح گاما و ...
۷. سنجش کیفیت تصویر
۸. ذخیره سازی اطلاعات در تصویر
۹. انطباق تصاویر

۱-۲) تصاویر دیجیتالی

تصاویر سنجش شده که از تعداد زیادی مربعات کوچک(پیکسل) تشکیل شده‌اند. هر پیکسل دارای یک شماره رقمی(Digital Number) می‌باشد که بیانگر مقدار روشنایی آن پیکسل است. به این نوع تصاویر، تصاویر رستری هم می‌گویند. تصاویر رستری دارای سطر و ستون میاشند.

۱-۳) مقادیر پیکسل‌ها

مقدار انرژی مغناطیسی که یک تصویر دیجیتالی به هنگام تصویر برداری کسب می‌کند، رقم‌های دوتایی(Digit binary) یا بیت‌ها(Bits) را تشکیل می‌دهند. حداکثر تعداد حالات برای روشنایی، بستگی به تعداد بیت‌ها دارد. بنابراین ۸ بیت یعنی ۲۵۶ شماره دیجیتالی که دامنه‌ای از ۰ تا ۲۵۵ دارد. به همین دلیل است که وقتی شما تصویر رستری از گیرنده خاصی مانند TM را وارد نرم افزاری می‌کنید تغییرات میزان روشنایی را بین ۰ تا ۲۵۵ نشان می‌دهد.

۱-۴) دقت تصویر

دقت تصویر بستگی به شماره پیکسل‌ها دارد. با یک تصویر ۲ بیتی، حداکثر دامنه روشنایی ۲×۲ یعنی ۴ می‌باشد که دامنه آن از ۰ تا ۳ تغییر می‌کند. در این حالت تصویر دقت (تفکیک پذیری لازم) را ندارد. تصویر ۸ بیتی حداکثر دامنه ۲۵۶ دارد و تغییرات آن بین ۰ تا ۲۵۵ است. که دقت بالاتری دارد.

۱-۵) کاربرد پردازش تصویر در زمینه‌های مختلف

- ماشین بینایی و پردازش تصویر در اتوماسیون صنعتی

کنترل ماشین‌آلات و تجهیزات صنعتی یکی از وظایف مهم در فرآیندهای تولیدی است. بکارگیری کنترل خودکار و اتوماسیون روزبه روز گسترده تر شده و رویکردهای جدید با بهره‌گیری از تکنولوژی‌های نو امکان رقابت در تولید را فراهم می‌سازد. لازمه افزایش کیفیت و کمیت یک محصول، استفاده از ماشین آلات پیشرفته و اتوماتیک می‌باشد. ماشین‌آلاتی که بیشتر مراحل کاری آنها به طور خودکار صورت گرفته و اتکای آن به عوامل انسانی کمتر باشد. امروزه استفاده از تکنولوژی ماشین بینایی و تکنیک‌های پردازش تصویر کاربرد گسترده‌ای در صنعت پیدا کرده‌است و کاربرد آن به‌ویژه در کنترل کیفیت محصولات تولیدی، هدایت روبات و مکانیزم‌های خود هدایت شونده روز به روز گسترده‌تر می‌شود.

عدم اطلاع کافی مهندسین از تکنولوژی ماشین بینایی و عدم آشنایی با توجیه اقتصادی بکارگیری آن موجب شده‌است که در استفاده از این تکنولوژی تردید و در بعضی مواقع واکنش منفی وجود داشته باشد. علی رغم این موضوع، ماشین بینایی روز به روز کاربرد بیشتری پیدا کرده و روند رشد آن چشمگیر بوده‌است. عملیات پردازش تصویر در حقیقت مقایسه دو مجموعه عدد است که اگر تفاوت این دو مجموعه از یک محدوده خاص فراتر رود، از پذیرفتن محصول امتناع شده و در غیر این‌صورت محصول پذیرفته می‌شود. در زیر نمونه پروژه‌هایی که در زمینه پردازش تصاویر پیاده سازی شده‌است، ذکر شده است. این پروژه‌ها با استفاده از پردازش تصویر، شمارش و اندازه‌گیری اشیا، تشخیص عیوب، تشخیص ترک، دسته بندی اشیا و عملیات بیشمار دیگری را انجام می‌دهند:

۱. اندازه گیری و کالیبراسیون
۲. جداسازی پین‌های معیوب

۳. بازرسی لیبل و خواندن بارکد
۴. بازرسی عیوب چوب
۵. بازرسی قرص
۶. بازرسی و دسته بندی زعفران
۷. درجه بندی و دسته‌بندی کاشی
۸. بازرسی میوه
۹. بازرسی شماره چک

- حمل و نقل
  - سرعت سنجی خودرو

رشد استفاده از سیستم های کنترل هوشمند سرعت و ثبت تخلف در سال‌های اخیر مشهود بوده است. این سیستم ها برای تشخیص سرعت خودروهای عبوری، از روش های متفاوتی استفاده می‌کنند. یکی از این روش ها بهره‌گیری از پردازش تصویر است. با استفاده از دو دوربین و کالیبره کردن آن‌ها و پردازش تفاوت دید موجود در تصاویر بدست آمده از دو دوربین امکان تشخیص عمق خودروی عبوری فراهم می‌شود. از مزایای استفاده از روش سرعت سنجی خودروها به کمک پردازش تصویر نسبت به دیگر روش ها مانند رادار و یا لیزر، پسیو بودن این روش است. بدین ترتیب امکان ثبت نشدن تخلف به علت استفاده متخلف از دستگاه های مختل کننده (Jammer) وجود ندارد. همچنین دستگاه‌های هشدار دهنده وجود سیستم های سرعت سنج که با آشکار سازی امواج رادار به متخلف هشدار می‌دهند نیز دیگر کاربری نخواهند داشت. این سیستم های سرعت سنج دارای دونوع هستند.

۱. سرعت سنج ثابت که بر روی پایه‌هایی در کنار بزرگراه‌ها و جاده ها نصب می شوند.
۲. سرعت سنج خودرویی که بر روی خودروی پلیس سوار می شود.

  - پلاک خوانی خودرو

پلاک خوانی خودرو با آموختن کاراکتر هایی که پلاک خودرو از آن تشکیل شده‌است می‌توان در تصویر بدست آمده از دوربین پلاک خوان به دنبال آن کاراکتر ها گشت. سیستم های پلاک خوان خودرو کاربردهای مختلفی دارد که می توان به چند نمونه اشاره کرد.

۱. پلاک خوانی پارکینگ های مجتمع های بزرگ
۲. پلاک خوانی جهت کنترل عبور و مرور در مرزها
۳. پلاک خوانی خودروهای متخلف در سیستم های ثبت تخلف و اعمال جریمه

۲) فصل دوم : معرفی

۲-۱) تاریخچه

تاریخچه پنهان‌نگاری به ۵ قرن قبل از میلاد مسیح و کشور یونان برمی‌گردد، در آن زمان مردی به نام هیستایاکاس می‌خواست پیغامی را به صورت محرمانه برای شخص دیگری بفرستد. وی برای فرستادن پیغام مورد استفاده از این روش استفاده کرد: او برده ای را برای این کار انتخاب کرد و موهای سر برده را تراشید و پیغام محرمانه را بر روی پوست سر برده خالکوبی کرد و سپس مدتی صبر کرد تا موهای فرد رشد کرده و به حالت اول برگشت و بعد او را به سمت مقصد (گیرنده) روانه کرد. در مقصد، گیرنده‌ی پیغام دوباره موهای برده را تراشید و پیغام را بر روی پوست سر او مشاهده کرد. یکی از کاربردهای پنهان‌نگاری رد و بدل کردن پیغام در جنگ‌ها است. نازی ها برای تبادل پیام‌هایشان از انواع روش‌های پنهان‌نگاری استفاده می‌کردند. یکی از این روش ها ارسال پیام‌های رمزی بود. به طور مثال یک جاسوس نازی پیغام زیر را فرستاد:

Apparently neutral's protest is thoroughly discounted and ignored. Isman hard hit. Blockade issue affects pretext for embargo on by-products, ejecting suets and vegetable oils.

حال اگر حرف دوم همه کلمات را کنار یکدیگر قرار دهید جمله زیر بدست می‌آید که منظور پیام اصلی جاسوس نازی بوده است:

Pershing sails from NY June I.

در عصرحاضر پنهان‌نگاری امروزه به کامپیوترها راه یافته‌است. امروزه روش‌های زیادی برای استفاده از پنهان‌نگاری در کامپیوتر ها وجود دارد. یکی از روش های اساسی پنهان‌نگاری با استفاده از کامپیوتر، مخفی کردن یک فایل متنی یا عکس در داخل یک عکس دیگر است.

پنهان‌نگاری در قرن های ۱۵ و ۱۶ توسعه یافت اما در قرن بیستم بود که حقیقتا پنهان‌نگاری شکوفا شد.

۲-۲) معرفی

پنهان‌نگاری[۱] علم ارسال و دریافت اطلاعات محرمانه به صورت مخفی بوسیله تزریق آنها در داده‌های چندرسانه‌ای می‌باشد. پنهان‌یابی[۲] نیز به هنر یافتن پیغام های مخفی در داده‌های پنهان‌نگاری شده اطلاق می‌شود. معیارهای کیفیت پنهان‌نگاری شامل غیرقابل تشخیص بودن[۳]، مقاومت[۴] و ظرفیت داده قابل تزریق[۵] می‌باشد. این ویژگی‌ها پنهان‌نگاری را از روش های دیگر همچون نهان‌نگاری و رمزنویسی مجزا می‌سازد. روش پنهان‌نگاری کم ارزش ترین بیت، مرسوم‌ترین روش در پنهان‌نگاری شناخته می‌شود. پوشش و فیلتر کردن، الحاق کردن و دگرگونی تکنیک های دیگری از پنهان‌نگاری به شمار می‌آید[۲]. روش کم ارزش ترین بیت[۶] بدین گونه عمل می‌کند که دو یا چند بیت کم ارزش از پیکسل‌ها با اطلاعات مخفی جایگزین می‌شود. آقای چن و آقای چینگ[۳] برای جایگذاری داده محرمانه از طریق روش کم ارزش ترین بیت با تنظیم

---

۱ Steganography

۲ Steganalysis

۳ Undetectability

۴ Robustness

۵ Capacity

۶ Least Significant Bit (LSB)

بهینه پیکسل‌ها روشی ساده برای پنهان نگاری مطرح کردند. آقای وو و آقای تسای[۴] روش اختلاف ارزش پیکسل‌ها[۱] را مطرح کردند که کیفیتی بالاتر در عکس نهایی و نیز مخفی سازی اطلاعات بیشتری را ارائه می‌کرد [۵],[۶]. به همین دلیل بسیاری روش‌های دیگر بر مبنای روش اختلاف ارزش پیکسل ها ارائه شد.

در این پایان‌نامه روشی برای پنهان‌نگاری عکس‌های رنگی با استفاده از اصلاح مقادیر اجزای رنگی هر پیکسل[۲] معرفی می‌شود. در مقایسه با روش اختلاف ارزش پیکسل‌های آقای وو و تسای، امنیت مخفی سازی اطلاعات بالاتر بوده و نیز کیفیت تصویر نهایی بالاتر خواهد بود. نتایج تجربی از روش ارائه شده کیفیت بالاتری نسبت به روش اختلاف پیکسل‌ها دارد[۱].

۲-۳) انواع استگانوگرافی

عمل پنهان‌نگاری می‌تواند در انواع رسانه‌های دیجیتالی انجام شود، ولی به طور کلی از یک فرمول واحد برای تغییر بیت‌ها استفاده می‌کند، روش کلی به صورت زیر است:

شئ‌ای که قرار است اطلاعات در آن نگهداری شود + اطلاعاتی که باید مخفی شوند + الگوریتم مورد نظر = شئ مورد نظر که اطلاعات در آن مخفی شده‌اند.

- استگانوگرافی در متن

پنهان‌نگاری در متن بوسیله ایجاد فاصله‌های ناچیز بین حروف کلمات یک سند و یا فاصله مابین خطوط برای این امر استفاده می‌کند. شاید به نظر برسد ایجاد فاصله در اجزای یک سند مشخص باشد، ولی فواصلی که برای گنجاندن حروف استفاده می‌شود بسیار ناچیز بوده و توسط چشم غیرقابل تشخیص است.

- استگانوگرافی در تصاویر

وقتی از یک تصویر برای مخفی نمودن یک متن(نوشته) استفاده می‌شود، معمولاً ازروش LSB استفاده می‌شود. ضمناً اگر در درون یک تصویر اطلاعاتی درج شده باشد وسپس این تصویربه فرمت دیگری تبدیل شود، به احتمال بسیار زیاد، بخش اعظمی ازاطلاعات مخفی شده از بین می رود وبخش باقی مانده نیزشاید با سختی فراوان قابل بازیابی باشد.

- استگانوگرافی درصوت

برای این منظور نیز از روشی مشابه روش LSB استفاده می کنند. البته مشکل استفاده از بیت های کم ارزش در یک فایل صوتی، این است که تغییرات در این بیت ها نیز برای گوش انسان قابل تشخیص است .

در حقیقت Spread Spectrum روش دیگری برای مخفی نمودن اطلاعات در یک فایل صوتی است. دراین روش، یک نویز به طور تصادفی در سراسر فایل پخش می شود و اطلاعات در کنار این نویزها قرارداده می شوند. Echo data hiding نیز روش دیگری برای مخفی نمودن اطلاعات در یک فایل صوتی است. این روش از اکو (پژواک) در فایل استفاده می‌کند تا بتواند اطلاعات را مخفی نماید. دراین وضعیت با اضافه کردن صداهای اضافی به بخش های اکو، می توان اطلاعات را در این قسمت ها مخفی نمود.

---

[۱] Pixel Value Differencing (PVD)

[۲] Pixel Value Modification (PVM)

- استگانوگرافی در ویدئو

برای این کار ، معمولاً از روش DCT استفاده می شود. این تکنیک شبیه تکنیک LSB است. یک فایل ویدئویی از تعدادی تصاویر پشت سرهم تشکیل شده است که این تصاویر به نام فریم شناخته می شوند. بنابراین کافی است که اطلاعات خود را درهر فریم یک فایل ویدئویی ، به روش LSB مخفی نماییم.

۲-۴) آثار مرتبط

روش اختلاف ارزش پیکسل‌ها روش مخفی سازی اطلاعات بر پایه اختلاف پیکسل‌های تصویر اولیه می‌باشد که این روش به دو دسته منطقه صاف و منطقه سخت تقسیم می‌شود. در منطقه صاف، یک پیکسل اختلاف ناچیزی با پیکسل‌های همسایه‌اش دارد که برای جایگذاری داده مناسب نمی‌باشد پس پیکسل‌های با اختلاف زیاد نسبت به همسایه‌هایشان برای جایگذاری داده برگزیده می‌شوند. این روش ظرفیت جایگذاری بسیار خوبی را تضمین می‌کند؛ اما تصویر نهایی به راحتی قابل تشخیص است.

نسخه اصلاح شده روش اختلاف ارزش پیکسل‌ها با نام روش اختلاف ارزش سه‌وجهی پیکسل‌ها مطرح شد[۸]. در روش اصلاح شده، سه پیکسل همسایه جهت جایگذاری بیت‌های مخفی در سه جهت مختلف انتخاب می‌شود تا بتواند بار بیشتری از داده را در عکس اولیه ذخیره کند. روش اولیه اختلاف ارزش پیکسل‌ها برخلاف روش اصلاح شده تنها از یک جهت برای جایگذاری داده ها استفاده می‌کرد؛ درحالیکه روش اصلاح شده از سه جهت افقی، عمودی و مورب برای جایگذاری داده مخفی استفاده می‌کند. روش اصلاح شده نیازمند تقسیم پیکسل‌های تصویر اصلی به بلوک‌های ۲ در۲ است.

یک نسخه اصلاح شده‌ی دیگر از روش اختلاف ارزش پیکسل‌ها تحت عنوان روش اختلاف ارزش انطباقی پیکسل‌ها مطرح شد. این روش تنها بر روی تصاویر سیاه‌وسفید قابل پیاده‌سازی است. در تصاویر دیجیتالی سیاه‌وسفید بازه‌ی ارزش پیکسل‌ها از صفر تا ۲۵۵ می‌باشد. چانچه برای پنهان‌نگاری از یک تصویر سیاه‌وسفید و بر مبنای روش اولیه اختلاف ارزش پیکسل ها استفاده شود، تصویر نهایی از بازه ارزش پیکسل های یک عکس سیاه وسفید خارج خواهد شد؛ به همین دلیل از روش اختلاف ارزش انطباقی پیکسل‌ها استفاده می‌شود[۸]. پس روش اختلاف ارزش انطباقی پیکسل‌ها برای اینکه بتواند از مشکل سر ریزی از بازه سیاه‌وسفید جلوگیری کند، از عملکردهای استاندارد و نیز برخی شرایط بهره می‌برد. روش اختلاف ارزش انطباقی پیکسل‌ها نه تنها با روش اختلاف ارزش پیکسل‌های وو و تسای برابری می‌کند بلکه از نظر کیفیت تصویر نهایی نیز بسیار رضایت بخش می‌باشد.

پادما و دکتر ونکاتارامانی[۱] روش مخفی سازی‌ای را با استفاده از طرح حرکت زیگزاگی مطرح کردند. این روش با استفاده از در نظر گرفتن اختلاف ارزش دو یا سه پیکسل همسایه در تصویر اصلی بسیار موفق عمل کرد[۹]. روش بررسی زیگزاگی باعث بهبود امنیت و کیفیت در تصویر نهایی و نیز ظرفیت بالای جایگذاری اطلاعات شد. این روش همچنین از روش اصلاح خطای همینگ برای یافتن و اصلاح خطاها بهره گرفته و برای ارتباطی امن و قابل اطمینان بکار می‌رود[۱۰]. پنهان نگاری تصاویر رنگی بر پایه روش اختلاف ارزش پیکسل ها مشکل سرریزی ارزش پیکسل‌های تصویر را حل می‌کند. این روش امنیت بالاتری را نسبت به روش اختلاف ارزش پیکسل‌های مورد استفاده در تصاویر سیاه‌وسفید تضمین می‌کند و همچنین تصویر نهایی با کیفیت‌تری را نیز ارائه می‌کند[۱۱].

---

[۱] M. Padmaa and Dr. Y. Venkataramani

۲-۴-۱) معرفی روش LSB

دراین قسمت تکنیک LSB را بر روی یک فایل تصویری شرح می‌دهیم. البته قبل از ادامه بحث، بهتراست که درباره چگونگی ذخیره یک تصویر دیجیتالی، توضیحاتی را داشته باشیم.

هر فایل تصویری صرفاً یک فایل دودویی است که حاوی رنگ یا شدت نور هر پیکسل برحسب عددی دودویی است. تصاویر معمولاً از فرمت ۸ بیتی یا ۲۴ بیتی استفاده می‌کنند. در فرمت ۸ بیتی، تنها قادر به استفاده از ۲۵۶ رنگ برای هرپیکسل هستیم (از این ۸ بیت، هر بیت می تواند یکی از مقادیر ۰ یا ۱ را برگزیند که در مجموع $۲^۸$ ، یعنی ۲۵۶ رنگ مختلف داریم). درفرمت ۲۴ بیتی نیز هرپیکسل از $۲^{۲۴}$ بیت رنگ می‌تواند استفاده کند. در این فرمت، هرپیکسل از۳ بایت استفاده می‌کند. هر بایت نشان دهنده شدت روشنایی یکی از سه رنگ اصلی آبی، قرمز و سبز است.

به عنوان نمونه، رنگ‌ها در فرمت html بر اساس فرمت ۲۴ بیتی است، که هر رنگ، کدی بر مبنای ۱۶ دارد که از ۶ کاراکتر تشکیل شده است. دو کاراکتر اول، مربوط به رنگ قرمز، دو کاراکتر دوم مربوط به رنگ آبی و دو کاراکتر سوم، مربوط به رنگ سبز است . برای نمونه برای ساختن رنگ نارنجی، باید مقادیر شدت روشنایی رنگ های قرمز، سبز و آبی ، به ترتیب ۱۰۰٪ و ۵۰٪ و ۰ باشد که در html با #FF7F00 قابل تعریف است.

همچنین اندازه یک تصویر، به تعداد پیکسل‌ها در تصویر بستگی دارد. برای نمونه، برای تصویری با رزولوشن ۴۸۰ ×۶۴۰ که از فرمت ۸ بیتی استفاده می کند، اندازه تصویر باید حدود 640×480×3 =307 KB باشد. به عنوان مثالی دیگر، تصویری با رزولوشن ۱۰۲۴×۷۶۸ که ازفرمت ۲۴ بیتی استفاده می کند، اندازه تصویر باید حدود 1024×768×3 = 2.36MB باشد. البته این اعداد درصورتی صادق هستند که هیچ فشردگی بر روی فایل اعمال نشده باشد. لازم به ذکر است، فرمت‌های تصویری GIF و BMP، 8 بیتی بوده و از روش Lossless[۱] استفاده می کنند. در مقابل، فرمت JPEG از روش Lossy[۲] استفاده می‌کند. در استگانوگرافی از فرمت های GIF وBMP و PNG به دلیل ویژگی‌هایی که دارند، استفاده می‌شوند.

ساده‌ترین راه برای پیاده سازی استگانوگرافی استفاده ازبیت‌های کم ارزش هرپیکسل یا همان روش LSB است. برای این منظور اطلاعات را به دو صورت دودویی درآورده و در بیت‌های کم ارزش پیکسل‌های تصویر قرار می‌دهیم . البته ما خواهان این هستیم که تصویر مورد نظر نیز زیاد تغییری نداشته باشد. بنابراین اگر از فرمت ۲۴ بیتی برای این کار استفاده کنیم، چشم انسان قادر به شناسایی این تغییر در تصویر نیست.

فرض کنید که سه پیکسل مجاور هم داریم که به صورت زیر کد شده‌اند:

	قرمز	آبی	سبز
پیکسل ۱	۱۰۰۱۰۱۰۱	۰۰۰۰۱۱۰۱	۱۱۰۰۱۰۰۱
پیکسل ۲	۱۰۰۱۰۱۱۰	۰۰۰۰۱۱۱۱	۱۱۰۰۱۰۱۰
پیکسل ۳	۱۰۰۱۱۱۱۱	۰۰۰۱۰۰۰۰	۱۱۰۰۱۰۱۱

[۱] روشی در گرافیک برای فشرده سازی تصاویراست که درآن تمام اطلاعات تصویرحفظ می شود و فقط از تعداد محدودی ازاطلاعات استفاده می شود و در برنامه‌های خاصی، اطلاعات حفظ شده قابل بازیابی است بنابراین از کیفیت تصویر نیز کاسته نمی شود

[۲] دراین روش بخشی ازاطلاعات تصویر برای همیشه ازبین می رود.

حال فرض کنید که می خواهیم ۹ بیت اطلاعات ۱۰۱۱۰۱۱۰۱را در این پیکسل‌ها مخفی نماییم (فرض میشود که این ۹ بیت اطلاعات رمزنگاری شده، یک پیام باشند). حال اگر از روش LSB استفاده شود و این ۹ بیت در بیت‌های کم ارزش بایت‌های این سه پیکسل قرار داده‌شوند، اعداد زیر را خواهیم داشت.

	قرمز	آبی	سبز
پیکسل ۱	۱۰۰۱۰۱۰۱	۰۰۰۰۱۱۰۰	۱۱۰۰۱۰۰۱
پیکسل ۲	۱۰۰۱۰۱۱**۱**	۰۰۰۰۱۱۱۰	۱۱۰۰۱۰۱**۱**
پیکسل ۳	۱۰۰۱۱۱۱۱	۰۰۰۱۰۰۰۰	۱۱۰۰۱۰۱۱

ملاحظه می‌شود که فقط ۴ بیت تغییر داده شده‌اند و این لطمه زیادی به تصویر وارد نمی‌کند، به طوری که چشم اصلاً قادر به تشخیص این تغییرات نیست. به عنوان مثال، تغییر بیت رنگ آبی از ۱۱۱۱۱۱۱۱ به ۱۱۱۱۱۱۱۰ اصلاً برای چشم قابل تشخیص نیست. ناگفته نماند تصاویر سیاه وسفید نیز برای پنهان‌نگاری بسیار مناسب هستند.

حال شاید خواهان مخفی کردن یک متن در یک تصویر باشیم. در این وضعیت هر کاراکتر، یک بایت (۸ بیت) فضا اشغال می‌کند. از آنجا که این بیت‌ها را باید درون پیکسل‌های تصویری قرار دهیم، می‌بایست این هشت بیت را به بسته‌های یک بیتی تقسیم نماییم و هر بیت را در بیت های سطح پایین یکی ازسه رنگ اصلی پیکسل ها، قرار دهیم با این شیوه، کلمات تمامی زبان هایی را که با ساختار ASCII یا UTF-8 سازگارند، می توان درون تصاویر جاسازی نمود.

۲-۵) تفاوت پنهان‌نگاری و رمزنگاری

تفاوت اصلی رمزنگاری[۱] و پنهان‌نگاری آن است که در رمزنگاری هدف اختفاء محتویات پیام است و نه به طور کلی وجود پیام! اما در پنهان‌نگاری هدف مخفی کردن هر گونه نشانه‌ای از وجود پیام است. در مواردی که تبادل اطلاعات رمز شده مشکل آفرین است باید وجود ارتباط پنهان گردد. به عنوان مثال اگر شخصی به متن رمزنگاری شده‌ای دسترسی پیدا کند، به هر حال متوجه می‌شود که این متن حاوی پیام رمزی می‌باشد. اما در پنهان نگاری شخص سوم ابدا از وجود پیام مخفی در متن اطلاعی حاصل نمی‌کند. در موارد حساس ابتدا متن را رمزنگاری کرده، آنگاه آن را در متن دیگری پنهان‌نگاری می‌کنند. اما با وجود بهتر بودن پنهان‌نگاری در مقابل رمزنگاری همچنان بسیاری از مردم می‌گویند رمزنگاری بهتر از پنهان‌نگاری عمل می‌کند .

۲-۶) کاربردها

می‌توان برای جلوگیری از پیگرد انتشار غیرقانونی محتوا و فایل‌های تولیدی از این روش استفاده کرد. با فرض اینکه یک تصویر یا فایل PDF تولید شده‌باشد؛ با استفاده از این روش شخص تولید کننده می‌تواند حق کپی[۲] اثر خود را در

---

[۱] Cryptography

[۲] Copyright

فایل مربوطه پنهان کند تا در صورت لزوم بعداً بتواند از حق خود دفاع کند. به این روش حفاظت از اثر، نهان‌نگاری[1] گفته می‌شود.

به عنوان مثالی دیگر شخصی می‌خواهد رمزعبور یا یک فایل مهم اداری را توسط ایمیل برای همکارش ارسال کند، آیا استفاده از پنهان‌نگاری راه خوبی برای این کار نیست؟ اگر ایمیل شخص یا همکارش هم توسط شخص دیگری (مثلاً سرویس دهنده اینترنت) شنود و کنترل شود، کمتر کسی می‌داند که در یک تصویر معمولی چه اطلاعاتی ممکن است وجود داشته باشد! و در پایان اگرشخصی بخواهد در یک تالار گفتگوی عمومی فایل یا اطلاعاتی خاصی را در اختیار برخی افراد قرار دهد، استگانوگرافی بهترین راه حل است.

۲-۷) پنهان‌یابی

درحالی که هدف پنهان‌نگاری مخفی کردن اطلاعات و جلوگیری از پیدا شدن و جلب توجه آنهاست، پنهان‌یابی علمی است که برای پیدا کردن چنین مطالب مخفی شده‌ای به کار می‌رود. نهان‌یابی شبیه یک کاراگاه است و پنهان‌نگاری شبیه یک مجرم. یکی سعی می‌کند دیگری را بیابد. (البته این بدین مفهوم نیست که پنهان‌نگاری بد است بلکه این مثال برای درک بهتر مطلب آورده شده‌است) پنهان‌یابی سعی می کند تا اطلاعات پنهان شده را پیدا کند اما اغلب متون مخفی که با استفاده از نرم افزارهای استگانوگرافی مخفی شده‌اند علامت خاصی از خود نشان نمی‌دهند؛ یعنی مثلا اگر به شما چندین عکس داده‌شود تا یک متن مخفی را ازدرون آنها پیدا کنید باید ابتدا تشخیص دهید که کدام عکس شامل این متن مخفی است چرا که هیچ علامت خاصی وجود ندارد تا شما آن را تشخیص دهید. حتی اگر عکس اولیه و اصلی نیز وجود داشته باشد به راحتی قابل تشخیص نیست زیرا نه از لحاظ ظاهری و نه از لحاظ حجم این دو عکس تفاوت چندانی با یکدیگر ندارند.

نسل‌های مختلفی از نرم افزار های استگانوگرافی وجود دارد که پنهان‌یابی یکی از انواع ان است. به طور کلی روش‌های پنهان‌نگاری در صورتی امن هستند که تصویر میزبان یا گنجانه دارای نشانه‌های قابل کشف نباشد. به بیان دیگر، خواص آماری تصویر میزبان یا گنجانه[2] باید همانند خواص آماری پوشانه[3] باشد. توانایی کشف پیام در تصویر به طول پیام پنهان بستگی دارد. واضح است که هرچه مقدار اطلاعاتی که در یک تصویر قرار می‌دهیم کمتر باشد امکان کمتری هست که نشانه‌های قابل کشف به وجود آید. انتخاب فرمت تصویر نیز تأثیر زیادی بر سیستم پنهان‌نگاری دارد. فرمت‌های فشرده نشده‌ای مثل BMP، فضای زیادی برای پنهان‌نگاری فراهم می‌کنند ولی استفاده از آنها به دلیل حجم بالای اطلاعات زائد آنها شک برانگیز است.

---

[1] Watermarking

[2] Cover image

[3] Stego image

۳) فصل سوم : معرفی روش PVM

روش اصلاح مقادیر پیکسل‌ها که از این به بعد آن را با نام PVM خواهیم شناخت، تصویر اصلی را به سه پالت رنگی قرمز و سبز و آبی تقسیم می‌کند. هر پیکسل شامل ۲۴بیت است که به سه دسته هشت تایی تقسیم می‌شود. هشت بیت اول برای رنگ قرمز، هشت بیت دوم برای رنگ سبز و هشت بیت سوم برای رنگ آبی در نظر گرفته می‌شود. در روش ارائه شده تمامی سه جزء برای جایگذاری داده استفاده می‌شود. ابتدا هر جزء رنگی یک پیکسل جدا می شود و سه ماتریس M×N تشکیل می‌شود به طوری که درایه های ماتریس اول نشان دهنده ارزش رنگ قرمز در هر پیکسل،درایه‌های ماتریس دوم نشان دهنده ارزش رنگ سبز و همچنین درایه‌های ماتریس سوم برای نشان دادن ارزش رنگ آبی است.

روش اصلاح مقادیر پیکسل‌ها برای مخفی سازی داده‌ها از یک روند ترتیبی برای هر جزء رنگی بهره می‌جوید بدین صورت که ابتدا رقم اول داده محرمانه را به درایه اول از ماتریس رنگ قرمز اختصاص می‌دهد به همین منوال رقم دوم به درایه اول ماتریس سبز و رقم سوم به درایه اول ماتریس آبی اختصاص داده می‌شود و این روند تا تمام شدن کل داده محرمانه ادامه می‌یابد. استفاده از این روش برای جایگذاری داده محرمانه باعث افزایش امنیت، ظرفیت و همچنین بهبود کیفیت تصویر نهایی می‌شود.

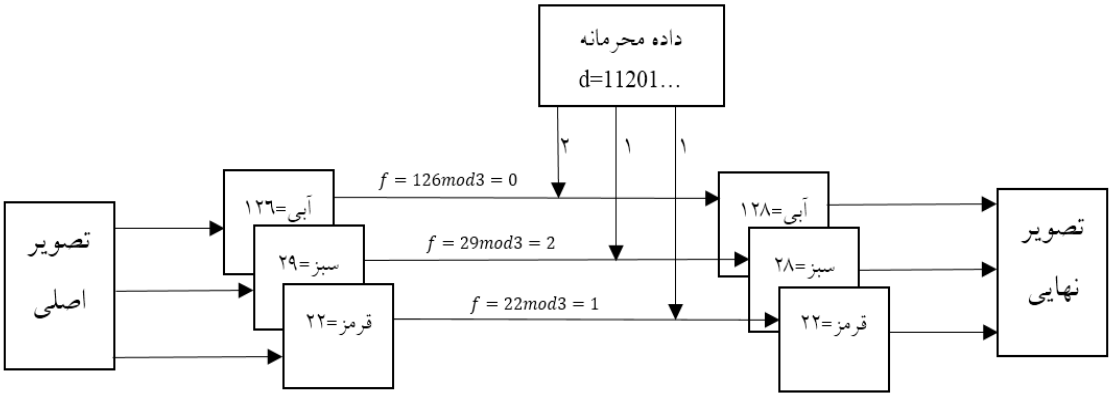
۳-۱) روند تزریق داده محرمانه

گام اول: مقادیر درایه‌های سه ماتریس رنگی بر اساس تصویر اولیه استخراج می‌شود. درایه های اول تا nام به صورت g_1، g_2 تا g_n نامگذاری می‌شود؛ به طوری که ماتریس قرمز به صورت g_{ri} ها، ماتریس سبز با g_{gi} ها و ماتریس آبی با g_{bi} نشان داده می‌شود.

گام دوم: متن داده محرمانه به صورت یک رشته از حروف در نظر گرفته می‌شود. هر حرف از رشته به کد اسکی نظیر آن تبدیل می‌شود سپس این کد اسکی به مبنای سه برده می‌شود. مقادیر به دست آمده برای تزریق به عکس آماده است. هر رقم از دنباله بدست آمده به ترتیب به g_{r1} ، g_{g1} ، g_{b1} ، g_{r2} ، g_{g2} ، g_{b2} و g_{r3} و... تعلق دارد.

گام سوم: برای هر درایه ماتریس فرمول محاسبه باقیمانده تقسیم بر سه را به صورت زیر اعمال می‌کنیم.

$$f_{(g_i)} = g_i \; mod \; 3 \qquad (۱)$$



شکل(۳-۱) مثالی از نحوه تزریق داده محرمانه به تصویر با روش اصلاح ارزش پیکسل‌ها

گام چهارم: در روش اصلاح مقادیر ارزش پیکسل‌ها، g_iها باید در بازه بین ۰ تا ۲۵۴ قرار بگیرند تا مقادیر مناسبی تلقی شوند. (چنانچه مقدار یکی از رنگ های پیکسل عدد ۲۵۵ شود روشی برای اصلاح وجود دارد که در قسمت های بعدی به آن اشاره خواهد شد). مقدار تابع f دارای سه حالت زیر است:

- حالت اول : اگر مقدار تابع f برابر با عدد اختصاص داده شده باشد تغییری در مقدار درایه داده نمی‌شود.
- حالت دوم: اگر مقدار تابع f بزرگتر از عدد اختصاص داده شده باشد چنانچه قدر مطلق اخلاف این‌دو برابر یک باشد از مقدار درایه متناظر یک واحد کم می‌شود و چنانچه قدرمطلق این اختلاف برابر دو باشد، دو واحد از درایه متناظر کم می‌شود.
- حالت سوم: اگر مقدار تابع f، کوچکتر از عدد اختصاص داده شده باشد، چنانچه قدر مطلق اختلاف این‌دو برابر یک باشد به مقدار درایه متناظر یک واحد اضافه می‌شود و چنانچه قدرمطلق این اختلاف برابر دو باشد به مقدار درایه متناظر دو واحد اضافه می‌شود.

گام پنجم: ترکیب ماتریس‌های جدید تولید شده در گام چهارم، تصویر نهایی را به دست می‌دهد.

برای حالتی که مقدار رنگ ۲۵۵ باشد مشکل سر ریز مقدار وجود دارد. برای حل این مشکل، به هنگام اعمال الگوریتم بررسی می‌کنیم که اگر مقدار رنگی ۲۵۵ باشد یک واحد از آن کم می‌کنیم، در این صورت مشکل سرریزی مقدار از بین می‌رود و از طرف دیگر تغییر چندانی در عکس نهایی ایجاد نمی‌کند. مقدار ۲۵۴ به هیچ عنوان سر ریز نخواهدشد، زیرا در نظر بگیرید باقیمانده تقسیم ۲۵۴ بر سه برابر دو است، حال چنانچه داده محرمانه نیز دو باشد، این دو باهم برابر هستند و عدد ۲۵۴ هیچ تغییری نمی‌کند و در حالتی‌که داده محرمانه ۰ و یا ۱ باشد، فقط از مقدار رنگ کم می‌شود، به همین دلیل در هیچ حالتی به مقدار ۲۵۴ اضافه نخواهد شد و مشکل سر ریز به کلی برطرف می‌شود.

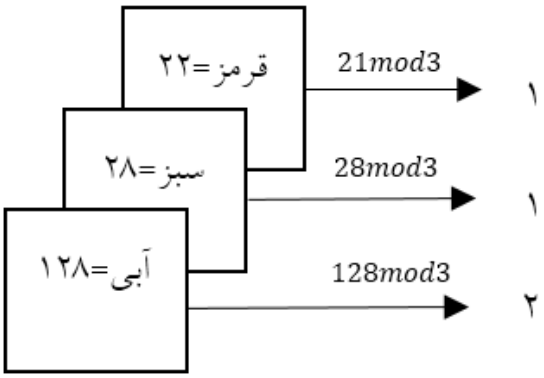
۳-۲) روند استخراج داده محرمانه از تصویر پنهان‌نگاری شده

گام اول: تصویر پنهان‌نگاری شده را به سه پالت رنگی قرمز، سبز و آبی تقسیم می‌کنیم. مقادیر درایه‌های بدست آمده در هریک از سه ماتریس باید در بازه صفر تا ۲۵۴ باشد. چنانچه درایه‌ای با مقدار ۲۵۵ موجود باشد می‌توان نتیجه گرفت که تصویر حاضر پنهان‌نگاری نشده‌است.

گام دوم: با اعمال فرمول۲ با ترتیب مشخص برای هریک از درایه‌های سه ماتریس می‌توان دنباله متن محرمانه را در مبنای سه مشاهده کرد.

$$d = g_{ri} \bmod 3 \qquad (۲)$$

گام سوم: دنباله تولید شده در گام دوم با استاندارد از پیش تعیین شده به مبنای ده بازگردانده می‌شود. این مقادیر معرف کد اسکی هر کاراکتر است، پس مقادیر را به حروف متناظر با کدهای اسکی تبدیل می‌کنیم تا رشته داده محرمانه ظاهر شود.



شکل(۳-۲) مثالی از بازیابی داده محرمانه از تصویر پنهان نگاری شده به روش اصلاح ارزش پیکسل‌ها

۳-۳) محاسبه ظرفیت تزریق

همانطور که در فصل دوم گفته شد، روش PVM هر رقم را در یک پیکسل پنهان می‌کند؛ حال لازم به ذکر است که منظور از رقم، ارقام ۰ الی ۹ است، در واقع این روش می‌تواند یک رقم از بازه بسته ۰ تا ۹ را در یک پیکسل جای دهد. دلیل اینکه، هر رقمی که بخواهیم تزریق کنیم ابتدا باید به مبنای سه برده شود. حداکثر تعداد رقم برای نمایش بزرگترین مقدار این بازه یعنی ۹، سه رقم است و باتوجه به فرمول (۱)، هر رقم در مبنای سه، در یک رنگ از پیکسل جای می‌گیرد، بنابراین برای نمایش سه رقم به سه رنگ یعنی یک پیکسل نیاز داریم. برای اعداد بزرگتر از ۹ یا باید خود عدد را به مبنای سه ببریم (در این صورت بیش از یک پیکسل برای نمایش نیاز است) و یا باید تک تک رقم ها را از عدد خود جدا کرده و تزریق کنیم.

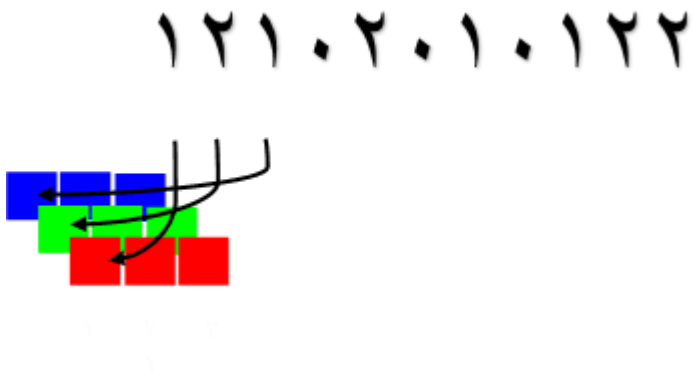
چنانچه بخواهیم حرف a را در تصویر پنهان کنیم لازم است معادل اسکی حرف a یعنی ۹۷ را در نظر گرفته و به مبنای سه ببریم. عدد ۹۷ در مبنای سه برابر ۱۰۱۲۱ است. حال عدد ۱۰۱۲۱ را بر اساس فرمول (۱) به تصویر تزریق می‌کنیم. حال چنانچه بخواهیم علامت ! را نیز در تصویر تزریق کنیم معادل اسکی "!" یعنی ۳۳ را به مبنای سه می‌بریم. عدد ۳۳ در مبنای سه برابر ۱۰۲۰ است و طبق روال قبل، ۱۰۲۰ را طبق فرمول به تصویر تزریق می‌کنیم.

برای رشته‌ای مانند !a!aa و یا هرمثال دیگر می‌بینیم که طول رشته محرمانه در مبنای سه متفاوت است؛ برای a، ۵رقم و برای !، ۴رقم. این امر در هنگام تزریق مشکلی ایجاد نمی‌کند ولی برای استخراج به مشکل بر می‌خوریم زیرا نمی‌دانیم چند رقم مربوط به کدام کاراکتر است. برای حل این مشکل به یک استاندارد معین برای تزریق نیاز داریم تا در هنگام استخراج نیز بر اساس همین استاندارد داده‌ها را بازیابی کنیم.

برای نمایش کلیه حروف و اعداد لاتین، به یک بایت حافظه نیاز داریم، به بیان دیگر با یک بایت می‌توان ۲۵۶حالت مختلف را نمایش داد که این تعداد حالت برای نمایش کلیه کاراکترهای حروف لاتین کافی است. برای ساخت یک استاندارد، ما در نظر گرفتیم که حداکثر مقداری که با یک بایت نمایش داده می‌شود را به مبنای سه ببریم؛ پس عدد ۲۵۵ را در مبنای سه محاسبه کردیم و به مقدار ۱۰۰۱۱۰ رسیدیم. بنابراین حداکثر به شش رقم نیاز است. استاندارد را بر اساس بلوک‌های ۶تایی قرار دادیم. برای مثال به هنگام تزریق ! کافی است به جای تزریق ۱۰۲۰ که از استاندارد بلوک شش تایی پیروی نمی‌کند، دو رقم صفر بی ارزش به سمت چپ ۱۰۲۰ اضافه کنیم تا آنرا برای تزریق استاندارد نماییم. درواقع به جای تزریق ۱۰۲۰، ۰۰۱۰۲۰ را به تصویر تزریق می‌کنیم. هرچند ساخت بلوک‌های شش تایی از ظرفیت داده قابل تزریق می‌کاهد ولی برای پیاده‌سازی عملی، این امر اجتناب ناپذیر است.

به این نتیجه رسیدیم که هر کاراکتر به شش رقم تبدیل می‌شود و هرسه رقم در یک پیکسل جای می‌گیرد؛ پس می‌توان گفت برای تزریق هر کاراکتر لاتین به دو پیکسل نیاز داریم. بر این اساس حداکثر تعداد کاراکتر قابل تزریق به یک تصویر x×y برابر است با:

$$Max\ Secret\ Data = \frac{x \times y}{2} - 6 \qquad (۳)$$



شکل(۳-۳). نحوه تزریق هر دیجیت به هر رنگ

۳-۳-۱) ظرفیت تزریق عکس در عکس

به دنبال پنهان‌نگاری متن در تصویر به روش PVM، می‌توان با اندکی تغیر، تصویری را در تصویری دیگر پنهان کرد. روش کار بر اساس الگوریتم PVM است ولی به هنگام پیاده‌سازی اندکی تغییرات در کد برنامه لازم است. در این قسمت به محاسبه ظرفیت تصویر برای پنهان‌نگاری تصویر در تصویر می‌پردازیم. برای فهم آسان‌تر، تصویری که قصد مخفی کردنش را داریم با نام عکس مخفی و تصویری که عکس مخفی در آن مخفی شده‌است با نام عکس پوششی خطاب می‌کنیم.

به منظور پنهان‌نگاری تصویر در تصویر، ابتدا اطلاعات عکس مخفی را به رشته‌ای از ارقام در مبنای سه تبدیل می‌کنیم، سپس رشته ارقام را همانند پنهان‌نگاری متن در تصویر، به عکس پوششی تزریق می‌کنیم. اطلاعات یک تصویر شامل مختصات طول و عرض هر پیکسل و رنگ آن است. چنانچه به همراه رنگ هر پیکسل مختصات آن‌را نیز یادداشت کنیم، حجم زیادی برای اطلاعات هر پیکسل هدر می‌رود، بنابراین تصمیم شد که بر اساس یک فرمت معین، فقط رنگ پیکسل‌ها را به ترتیب از اولین پیکسل تا آخرین پیکسل یادداشت کنیم.

رنگ هر پیکسل مقداری مابین بازه‌ی بسته صفر تا ۲۵۵ است، پس شرایطی مانند کدهای اسکی در پنهان‌نگاری متن در عکس دارد. بر این اساس برای ذخیره سازی هر رنگ به دو پیکسل نیاز است. به طور کلی به ازای هر پیکسل از عکس مخفی، شش پیکسل از عکس پوششی نیاز است.

به هنگام استخراج، نیاز است ابعاد عکس مخفی و طول رشته آماده تزریق را بدانیم، به همین دلیل در روند پنهان‌نگاری، ابعاد عکس مخفی را به همراه طول داده مخفی به عکس پوششی تزریق می‌کنیم. برای این منظور، شش پیکسل اول رشته آماده تزریق به طول این رشته و شش پیکسل بعدی برای یادداشت ابعاد عکس مخفی اختصاص دارد.

طبق گفته‌های فوق، ظرفیت یک تصویر با ابعاد x×y برای پوشش دادن یک تصویر با ابعاد m×n برابر است با

$$m \times n \ \leq \frac{x \times y}{6} - 12 \qquad (۳)$$

۴) فصل چهارم : پیاده سازی

در این فصل به معرفی شبه‌کدهای استفاده شده در نرم‌افزار که ضمیمه پایان‌نامه است پرداخته خواهدشد. شبه-کدها بر اساس زبان C# بوده و متن کامل کدهای برنامه در قسمت ضمایم نیز نوشته شده‌است. به طور کلی برنامه از سه قسمت کلی تشکیل شده‌است؛ فرم اصلی، کلاس توابع و کلاس متغیرها. در ابتدا به معرفی توابع و کتابخانه‌های مورد نیاز برای پردازش تصویر و توابع تبدیل متغیر پرداخته می‌شود.

۴-۱) معرفی توابع و کتابخانه‌ها

در ابتدا برای خواندن یک تصویر از حافظه جانبی و همچنین ذخیره کردن عکس خروجی از کتابخانه I/O استفاده می‌شود که این کتابخانه توابع مورد نیاز برای خواندن و نوشتن فایل‌ها را به ما می‌دهد.

```
using System.IO;
```

برای پردازش تصویر و دسترسی به ساختار فایل‌های تصویری از کتابخانه Drawing استفاده می‌شود.

```
using System.Drawing;
```

سایر کتابخانه‌ها به صورت پیش‌فرض توسط C# فراخوانی شده‌اند.

۴-۱-۱) کار با تصاویر

به منظور خواندن یک فایل تصویری از کلاس Bitmap استفاده می‌شود. این کلاس اختصاصا برای دسترسی به اطلاعات تصاویر به کار گرفته می‌شود. نحوه تعریف یه شئ از این کلاس به صورت زیر است:

```
Bitmap MyBitmap = new Bitmap(پارامترها);
```

پارامترهای شئ ایجاد شده حالات مختلفی دارد، می‌توان مسیر یک فایل تصویری را در آن مشخص کرد و یا اندازه خاصی را به آن نسبت داد.

به منظور دسترسی به ابعاد فایل تصویری انتخاب شده از خاصیت width و height به صورت زیر استفاده می‌شود:

```
int x = MyBitmap.Width;
int y = MyBitmap.Height;
```

برای نشان دادن رنگ‌های یک پیکسل از ساختمان Color استفاده می‌شود. نحوه تعریف یک ساختمان از این نوع به صورت زیر است:

```
Color pixelColor;
```

از تابع ()Getpixel که یکی از خواص کلاس Bitmap است برای خواندن مقادیر یک پیکسل خاص استفاده می‌شود؛ ترکیب این تابع با کلاس Bitmap و ساختمان Color برای خواندن و نمایش اطلاعات پیکسلی به مختصات x و y به صورت زیر است:

```
Color pixelcolor = MyBitmap.GetPixel(x, y);
```

با اجرای این دستور مقادیر پیکسل x و y خوانده شده و در ساختمان pixelcolor ذخیره می‌شود.

برای ذخیره تصویر از خاصیت ()save از کلاس Bitmap استفاده می‌کنیم. ورودی‌های این متد، محل ذخیره و فرمت ذخیره است. ذخیره تصویری در درایو C و با فرمت PNG به صورت زیر است:

```
MyBitmap.Save("C:\", System.Drawing.Imaging.ImageFormat.Png);
```

خاصیت ()Setpixel از کلاس Bitmap برای نوشتن مقادیر جدید یک پیکسل استفاده می‌شود. پارامترهای آن طول پیکسل، عرض پیکسل و رنگ پیکسل است. شبه‌کد زیر نحوه استفاده را بیان می‌کند:

```
MyBitmap.SetPixel(x, y, Color.Black);
```

برای انتساب رنگ به پیکسل از خواص ساختمان Color استفاده می‌شود. یکی از این خواص ()FromArgb که به ما امکان استفاده از رنگ‌های ۳۲بیتی را می‌دهد. برای انتصاب رنگ a و b و c به پیکسل در مختصات x و y به صورت زیر عمل می‌شود:

```
MyBitmap.SetPixel(x, y, Color.FromArgb(a, b, c));
```

در متد ()FromArgb همانطور که از نام متد پیداست، پارامتر اول برای نشان دادن رنگ قرمز، پارامتر دوم برای رنگ سبز و پارامتر سوم برای انتصاب رنگ آبی استفاده می‌شود. چنانچه بخواهیم خاصیت شفافیت را به تصویر اضافه کنیم، پارامتر اول را، α [1] یا همان میزان شفافیت در نظر می‌گیریم. همگی این پارامترها مقداری مابین ۰ تا ۲۵۵ را قبول می‌کنند.

۴-۱-۲) کار با رشته‌ها

از آنجایی که قسمت اصلی استگانوگرافی به داده مخفی و اطلاعات متنی محرمانه مربوط می‌شود لذا برای ساخت نرم‌افزار آن باید توانایی کار با رشته‌ها را داشت. مباحث مورد نیاز برای کار با رشته‌ها در ادامه مطرح شده‌است.

داده محرمانه به صورت متن توسط کاربر وارد می‌شود. این متن به عنوان یک رشته در متغیری از نوع string ذخیره می‌شود. شبه‌کد زیر متنی را از Textbox می خواند:

```
string secretKey = Textbox1.Text;
```

در برخی قسمت‌ها لازم است طول متن وارد شده را بدانیم. شبه‌کد زیر طول متن وارد شده در Textbox را در متغیری از نوع integer قرار می‌دهد:

```
int secretKeyLength = Textbox1. Length;
```

همانطور که در فصل سوم گفته شد، برای اینکه متن داده محرمانه برای تزریق در عکس آماده شود باید ابتدا تک تک کاراکترها به کد اسکی متناظر با آن تبدیل شوند. شبه‌کد زیر برای تبدیل یک رشته خاص به کدهای اسکی بکار می‌رود، کلاس Encoding دارای خواصی برای این تبدیلات است، Encoding.ASCII عمل تبدیل را انجام می‌دهد:

```
ascii=
```

---

[1] آلفا یا میزان شفافیت تصویر، همانند رنگ‌ها مقداری مابین ۰ تا ۲۵۵ را می‌پذیرد، چنانچه مقدار آلفا وارد نشود، مقدار آن به صورت پیشفرض صفر یا وضوح کامل در نظر گرفته می‌شود.

```
(new string(Encoding.ASCII.GetBytes("same_text")).SelectMany(b =>
b.ToString()).ToArray()));
```

پس از بدست آوردن مقدار اسکی هر کاراکتر باید این مقدار از مبنای ده‌دهی به مبنای سه برده شود. پس از تبدیل مبنا نیاز است عدد بدست آمده را بر طبق استاندارد طرح شده در فصل سوم قالب بندی کرد. شبه‌کد زیر رشته‌ای را گرفته و به اندازه نیاز در سمت چپ آن صفر بی ارزش قرار می‌دهد تا بلوک‌های شش تایی آماده شود:

```
string padding = base3.PadLeft(6, '0');
```

نیاز است برای تک تک کاراکترها عمل تبدیل به کد اسکی و تبدیل مبنا انجام شود؛ کاربر متنی طولانی وارد می‌کند که در متغیری از نوع string ذخیره می‌شود. اگر بخواهیم کاراکترهای این رشته را تک تک بیرون بکشیم از تابع ()Substring به صورت زیر استفاده می‌شود:

```
int OneChar= base3.Substring(x, 1);
```

کد فوق از کاراکتر xام به اندازه یک کاراکتر شمارش کرده و پس از برگرداندن کاراکتر شمارش شده آن را در متغیر OneChar ذخیره می‌کند.

برای نمایش اطلاعاتی مانند مقادیر خروجی توابع و یا مقادیر یک متغیر نیاز است که خروجی به صورت رشته باشد این در حالیست که برخی متغیرها و یا خروجی توابع از نوع Integer هستند. به منظور تبدیلات انواع متغیرها از سری دستورات زیر استفاده می‌شود[1]:

```
int i;
string s;
```

```
s = i.ToString();          تبدیل عدد به رشته
i = int.Pars(s);           تبدیل رشته به عدد
```

۳-۱-۴) کلاس توابع

کلاس Functions توسط برنامه‌نویس ساخته‌شده و شامل برخی الگوریتم‌های مورد نیاز است که در طول کار از آن‌ها استفاده می‌شود[2]. برای نمونه در تبدیلات مبناها، C# فقط برای تبدیلات مبناهای ۲و ۸و ۱۰و ۱۶ دارای تابع پیش‌فرض است؛ پس نیاز است تابعی برای تبدیل مبنا از ده به سه و سه به ده نوشته‌شود. توابع این کلاس به شرح زیر است:

- تابع ConvertToBase

از این تابع برای تبدیل هر مقداری در مبنای ده به هر مبنای دلخواه استفاده می‌شود. این تابع دارای دو پارامتر بوده و خروجی آن به صورت رشته است. دستورات آن را در ادامه مشاهده می‌کنید:

```
public static string ConvertToBase3(int value, int toBase)
```

---

[1] فقط دو مورد از تبدیلات مورد نیاز ذکر شده‌است.

[2] برنامه نویسی ماژولار

```
        {

            if (toBase < 2 || toBase > 36) throw new
ArgumentException("toBase");
            if (value < 0) throw new ArgumentException("value");
            if (value == 0) return "0";
            string AlphaCodes = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";
            string retVal = "";
            while (value > 0)
            {
                retVal = AlphaCodes[value % toBase] + retVal;
                value /= toBase;
            }
            return retVal;

        }
```

پارامتر اول شامل مقداری است که می‌خواهیم به مبنای مورد نظر برود و پارامتر دوم مشخص کننده مبنای مورد نظر برای تبدیل است. در انتها نیز خروجی به صورت یک رشته بازگردانده می‌شود. نحوه فراخوانی این تابع به صورت زیر است:

```
String Tobase3 = Functions.ConvertToBase(ascii, 3);
```

- تابع ConvertToBase10

در خواندن داده محرمانه از عکس استگانوگرافی شده، توالی‌ای از اعداد در مبنای سه بدست خواهیم آورد که لازم است به مبنای ده بازگردانده شوند. این تابع صرفا به منظور تبدیل اعداد از مبنای سه به مبنای ده نوشته شده‌است:

```
public static int ConvertToBase10(string value)
        {
            int valueLenth, retval = 0;
            valueLenth = value.Length;
            for (int x = 1; x <= valueLenth; x++)
                retval += (int)(int.Parse(value.Substring(x - 1, 1)) *
Math.Pow(3, valueLenth - x));
            return retval;
        }
```

همانطور که می‌بینیم این تابع دارای یک پارامتر از نوع رشته بوده و خروجی آن عدد صحیح است. برای فراخوانی به صورت زیر عمل می‌کنیم:

```
int ascii = Functions.ConvertToBase10(base3);
```

- تابع setPixel

مفهوم اصلی روش PVM[1] در قالب تابع setPixel() بیان شده است. این تابع، داده محرمانه را به همراه مقدار هر رنگ دریافت کرده و طبق الگوریتم PVM مقدار جدیدی را برای رنگ پیکسل محاسبه کرده و برمی‌گرداند. این تابع دارای دو پارامتر است. پارامتر اول مربوط به داده محرمانه و دیگری مقدار رنگ پیکسل می‌باشد. پس از اعمال الگوریتم، خروجی به صورت عدد بازگردانده می‌شود.

```
1 public static int setPixel(int secretDigit, int colorPixel)
2         {
3             if (colorPixel == 255)
4                 colorPixel--;
5             string sw = (secretDigit - (colorPixel % 3)).ToString();
6             switch (sw)
7             {
8                 case "1":
9                 case "-1":
10                case "0":
11                    if (secretDigit < colorPixel % 3)
12                        colorPixel--;
13                    else if (secretDigit > colorPixel % 3)
14                        colorPixel++;
15                    break;
16
17                case "2":
18                case "-2":
19                    if (secretDigit < colorPixel % 3)
20                        colorPixel -= 2;
21                    else if (secretDigit > colorPixel % 3)
22                        colorPixel += 2;
23                    break;
24            }
25            return colorPixel;
26        }
```

در خط سوم بررسی می‌شود که مقدار هر رنگ در بازه ۰ تا ۲۵۴ باشد، در صورتی که مقداری ۲۵۵ باشد یک واحد از آن کم می‌کند. در خط پنجم فرمول (۱) اعمال شده و خروجی تابع با داده محرمانه مقایسه می‌شود. حالت‌های مختلف مقایسه بوسیله حلقه switch/case پیاده شده‌است.

۴-۲) کلاس متغیرها

به هنگام ساخت یک نرم افزار نیاز است برخی متغیرها به صورت global[2] تعریف شوند تا در همه فرم‌ها و توابع قابل دسترسی باشند. برای این منظور کلاسی با نام variables ساخته‌شد تا متغیرهای مورد نیاز در آن تعریف شده و در هرجای نرم‌افزار قابل دسترسی باشند. لیست و کاربرد هریک از متغیرها به شرح زیر است:

`public static string filePath;`	مسیر فایل را در خود نگه می‌دارد.
`public static string filePathSecretPhoto;`	مسیر فایل را در خود نگه می‌دارد.

---

[1] اصلاح ارزش پیکسل‌ها

[2] مفهوم متغیرهای محلی و عمومی مورد نظر است.

`public static string filePathWrite;`	مسیر فایل را در خود نگه می‌دارد.
`public static bool test = false;`	شرطی را بررسی می‌کند.
`public static bool P2PMain = false;`	شرط انتخاب تصویر اولیه را بررسی می‌کند.
`public static bool P2PSecret = false;`	شرط انتخاب تصویر نهایی‌را بررسی می‌کند.
`public static Bitmap MyBitmap;`	یک کلاس تصویر می‌سازد.

۴-۳) فرم اصلی

فرم اصلی شامل کلیدهایی برای مخفی کردن متن درعکس، خواندن عکس از متن، اختفای عکس در عکس و واکشی عکس از عکس است.

۴-۳-۱) کلید مخفی کردن متن در عکس

عکس مورد نظر توسط کلاس Bitmap به نرم‌افزار وارد می‌شود:

```
variables.MyBitmap = new Bitmap(variables.filePathWrite);
```

پس از لود عکس حداکثر داده مخفی که می‌توان به عکس تزریق کرد محاسبه می‌شود:

```
calculateMaxData = ((variables.MyBitmap.Width *
variables.MyBitmap.Height) / 2) - 3;
                if (SecretData.TextLength < calculateMaxData)
```

پس از محاسبه، طول متن وارد شده با حداکثر داده قابل تزریق مقایسه می‌شود. چنانچه عکس ظرفیت لازم را نداشته باشد برنامه از کاربر تقاضای کاهش متن می‌کند

```
if (SecretData.TextLength < calculateMaxData)
      Continue…
else
      MessageBox.Show("SecretData is to large!\nPlease select a larger
photo or reduce SecretData length");
```

از شش پیکسل اول به عنوان سرآیند استفاده می‌شود و شامل طول داده تزریق شده است. بعد از محاسبه مقدار داده تزریق شده، این مقدار به مبنای سه تبدیل شده و در ابتدای رشته آماده برای تزریق قرار می‌گیرد:

```
Base3 += Functions.ConvertToBase3(secretKey.Length * 2, 3).PadLeft(18,
'0');
```

با استفاده از حلقه for تک تک کاراکترهای داده محرمانه به کد اسکی و مبنای سه تبدیل شده و سپس به رشته آماده تزریق اضافه می‌شود. این حلقه به تعداد کاراکترها تکرار می‌شود، در مرحله اول کد اسکی متناظر بدست می‌آید سپس کد اسکی به مبنای سه تبدیل شده و در مرحله آخر استاندارد بلوک شش تایی روی آن اعمال می‌شود:

```
for (int t = 0; t < secretKey.Length; t++)
{
      int ascii = int.Parse(new
string(Encoding.ASCII.GetBytes(secretKey.Substring(t, 1)).SelectMany(b
=> b.ToString()).ToArray()));
      base3 = Functions.ConvertToBase3(ascii, 3);
      var padding = base3.PadLeft(6, '0');
      base3T += padding;
}
```

ابعاد عکس وارد شده بوسیله حلقه‌های for تودرتو پیمایش می‌شود. در هر چرخش حلقه یک پیکسل در نظر گرفته شده و برای هریک از رنگ‌های سه‌گانه آن تابع setPixel() فراخوانی می‌شود. پس از فراخوانی رنگ‌ها مقادیر جدید هر رنگ با متد Setpixel() از کلاس Bitmap بر روی عکس اعمال می‌شود. به دنبال واکشی هر رنگ، متغیر temp یک واحد افزایش پیدا می‌کند. وظیفه این متغیر شمارش کاراکتر به کاراکتر رشته آماده تزریق است. همچنین در انتهای هر چرخش حلقه بررسی می‌کند که آیا رشته محرمانه با پایان رسیده است یا نه؛ چنانچه به پایان رسیده باشد از حلقه خارج می‌شود.

```
1 for (x = 0; x < variables.MyBitmap.Width; x++)
2     for (y = 0; y < variables.MyBitmap.Height; y++)
3      {
4       pixelcolor = variables.MyBitmap.GetPixel(x, y);
5       red = Functions.setPixel(int.Parse(base3T.Substring(temp, 1)),
pixelcolor.R);
6       temp++;
7       green = Functions.setPixel(int.Parse(base3T.Substring(temp,
1)), pixelcolor.G);
8       temp++;
9       blue = Functions.setPixel(int.Parse(base3T.Substring(temp, 1)),
pixelcolor.B);
10      temp++;
11      Variables.MyBitmap.SetPixel(x, y, Color.FromArgb(red, green,
blue));
12      if (temp == base3T.Length)
13          goto done;
14     }
```

در انتها نوبت به ذخیره کردن عکس نهایی می‌رسد. دستور زیر دیالوگی[1] را برای ذخیره عکس با نام و مسیر دلخواه در اختیار کاربر قرار می‌دهد:

```
1  saveTo.Filter = "txt files (*.png)|*.png";
2  saveTo.FilterIndex = 2;
3  saveTo.RestoreDirectory = true;
4  if (saveTo.ShowDialog() == DialogResult.OK)
```

[1] دیالوگ یا openFrom.ShowDialog() به پنجره‌ای برای انتخاب فایل یا انتخاب مسیری برای ذخیره فایل اطلاق می‌شود.

```
5   {
6    variables.MyBitmap.Save(saveTo.FileName,
System.Drawing.Imaging.ImageFormat.Png);
7    labelHidedImagePath.Text = saveTo.FileName;
8    labelStatus.Text = "Succeed !";
9   }
10 else
11  MessageBox.Show("Save fail");
```

۴-۳-۲) کلید استخراج متن از عکس

استخراج متن از عکس همانند مخفی کردن متن است با این تفاوت که روالی معکوس را طی می‌کند. به وسیله یک دیالوگ عکس به نرم‌افزار فراخوانی می‌شود، طول داده تزریق شده استخراج شده و در متغیری ذخیره می‌شود، سپس رشته آماده تزریق[۱]، استخراج شده و در متغیری ذخیره می‌شود. هر شش کاراکتر این رشته به عنوان یک عدد در نظر گرفته شده و به تابع، برای تبدیل به مبنای ده، ارسال می‌شود. خروجی تابع کد اسکی‌ای است که به کاراکتری خاص تعلق دارد، کد اسکی به کاراکتر متناظر خود تبدیل شده و به رشته داده استخراجی اضافه می‌شود. در انتها محتوای رشته داده استخراجی نمایش داده می‌شود.

در این میان، متغیری تعداد چرخش‌های حلقه for را شمارش کرده و با طول داده تزریق شده مقایسه می‌شود، در صورتی که داده تزریق شده به پایان برسد از حلقه خارج می‌شود[۲].

کد زیر طول داده محرمانه را استخراج می‌کند:

```
for (y = 0; y < 6; y++)
 {
  pixelcolor = myBitmap.GetPixel(0, y);
  codeLength += (pixelcolor.R % 3).ToString();
  codeLength += (pixelcolor.G % 3).ToString();
  codeLength += (pixelcolor.B % 3).ToString();
 }
codeL = Functions.ConvertToBase10(codeLength) + 6;
```

برای استخراج داده نیز از کد زیر استفاده شده‌است:

```
for (x = 0; x < myBitmap.Width; x++)
 for (y = 0; y < myBitmap.Height; y++)
  {
   pixelcolor = myBitmap.GetPixel(x, y);
```

---

[۱] همان رشته‌ای مورد نظر است که در قسمت "کلید مخفی کردن متن در عکس" به آن اشاره شد.

[۲] برای خروج از حلقه، از دستور goto استفاده شده‌است.

```
    red = pixelcolor.R;
    base3T += (red % 3).ToString();
    green = pixelcolor.G;
    base3T += (green % 3).ToString();
    blue = pixelcolor.B;
    base3T += (blue % 3).ToString();
    counter++;
    if (counter == codeL)
     goto done;
   }
done:
string[] base3t = new string[(base3T.Length) / 6];
z = 0;
string key = null;
for (x = 18; x < base3T.Length; x += 6)
 {
  base3t[z] = Functions.ConvertToBase10(base3T.Substring(x,
6)).ToString();
  key +=
char.ConvertFromUtf32(Functions.ConvertToBase10(base3T.Substring(x,
6))).ToString();
  z++;
 }
```

۴-۳-۳) کلید مخفی سازی تصویر در تصویر

مخفی کردن تصویر در تصویر همانند مخفی کردن متن در عکس است. با این تفاوت که داده‌های مخفی، همان داده‌های تصویری است که می‌خواهیم مخفی کنیم. هر پیکسل را با پنج مولفه می‌توان نمایش داد؛ طول، عرض، مقدار رنگ قرمز، مقدار رنگ سبز و مقدار رنگ آبی. می‌توان با اعمال استانداردی خاص، از طول و عرض پیکسل‌ها صرف نظر کرده و فقط رنگ هر پیکسل را برای مخفی سازی یادداشت کرد. برای این امر لازم است طول و عرض تصویر را نیز به همراه اطلاعات رنگ‌ها به تصویر اصلی تزریق کنیم.

در این روش از مخفی سازی، شش پیکسل اول مربوط به طول داده محرمانه و شش پیکسل بعدی ابعاد تصویر محرمانه را در خود دارد. برای این منظور ابتدا طول داده مورد نیاز برای نمایش تصویر محرمانه را در رشته آماده تزریق جای می‌دهیم، در ادامه نیز ابعاد تصویر محرمانه را به ترتیب طول و عرض یادداشت می‌کنیم و درپایان بوسیله‌ی حلقه for تو در تو، مقادیر رنگی تصویر محرمانه را به رشته آماده تزریق اضافه می‌کنیم. به بیان دیگر، در این روش، داده محرمانه توالی‌ای از اعداد بازه‌ی بسته صفر تا ۲۵۵ است که مقدار رنگ‌های تصویر محرمانه را مشخص می‌کند.

```
string ImageValuesInbase3 =
Functions.ConvertToBase3(variables.MyBitmap.Width *
variables.MyBitmap.Height * 6, 3).ToString().PadLeft(18, '0');
```

کد فوق تعداد پیکسل مورد نیاز برای مخفی کردن تصویر محرمانه را محاسبه کرده و پس از تبدیل به مبنای سه و قالب بندی، به رشته آماده تزریق اضافه می‌کند.

```
ImageValuesInbase3 +=
Functions.ConvertToBase3(variables.MyBitmap.Width, 3).PadLeft(9, '0');
ImageValuesInbase3 +=
Functions.ConvertToBase3(variables.MyBitmap.Height, 3).PadLeft(9, '0');
```

کد فوق نیز ابعداد تصویر محرمانه را به ترتیب طول و عرض استخراج کرده و بعد از تبدیل به مبنای سه و قالب بندی، به رشته آماده تزریق اضافه می‌کند.

```
for (int i = 0; i < variables.MyBitmap.Width; i++)
 for (int j = 0; j < variables.MyBitmap.Height; j++)
  {
  ColorPixelIn = variables.MyBitmap.GetPixel(i, j);
  ImageValuesInbase3 += Functions.ConvertToBase3(ColorPixelIn.R,
3).PadLeft(6, '0');
  ImageValuesInbase3 += Functions.ConvertToBase3(ColorPixelIn.G,
3).PadLeft(6, '0');
  ImageValuesInbase3 += Functions.ConvertToBase3(ColorPixelIn.B,
3).PadLeft(6, '0');
                    }
```

کد فوق مقادیر هر رنگی پیکسل‌های تصویر محرمانه را استخراج کرده و پس از تبدیل به مبنای سه و قالب بندی، به رشته آماده تزریق اضافه می‌کند.

پس از آماده شدن رشته آماده تزریق، مابقی کارها همانند روال مطرح شده در قسمت ۴,۳,۱ انجام می‌شود که در اینجا از بیان مجدد آن صرف نظر می‌کنیم.

۴-۳-۴) کلید استخراج عکس از عکس

تمامی روال استخراج عکس از عکس مشابه استخراج متن از عکس است. با این تفاوت که بعد از استخراج داده آماده تزریق، آن را به متن تبدیل نمی‌کنیم! بلکه به جای تبدیل به متن، در صدد ساخت تصویری برمی‌آییم که داده‌های آن در رشته آماده تزریق گنجانده شده‌است.

```
Bitmap BitmapOut = new Bitmap(x, y);
```

همانطور که در قسمت ۴,۱,۱ بدان اشاره شد، کلاس Bitmap می‌تواند دارای پارامتر طول و عرض باشد. با این توضیح، پس از استخراج ابعاد عکس، یک کلاس Bitmap طبق کد فوق ایجاد می‌کنیم که پارامترهای آن طول و عرضی است که از تصویر پنهان‌نگاری شده استخراج شده‌است.

در ادامه بوسیله حلقه for توردرتو، ابعاد کلاس Bitmap ایجاد شده پیمایش شده و در هر گردش حلقه، مقادیر رنگی یک پیکسل از داده مخفی خوانده شده و بوسیله تابع متد ()Setpixel، رنگ به پیکسل اعمال می‌شود.

```
for (x = 0; x < BitmapOut.Width; x++)
 for (y = 0; y < BitmapOut.Height; y++)
  {
   red = Functions.ConvertToBase10(base3T.Substring(counter, 6));
   counter += 6;
```

```
 green = Functions.ConvertToBase10(base3T.Substring(counter, 6));
 counter += 6;
 blue = Functions.ConvertToBase10(base3T.Substring(counter, 6));
 counter += 6;
 BitmapOut.SetPixel(x, y, Color.FromArgb(red, green, blue));
}
```

کد فوق ابعاد کلاس Bitmap را پیمایش کرده و در هر گذر یک پیکسل رنگ‌دهی می‌شود. در پایان نیز کلاس Bitmap مورد نظر در آدرس تعین شده بوسیله کاربر، ذخیره می‌شود.

۴-۳-۵) سایر کلیدهای فرم

سایر کلیدهای موجود به منظور انتخاب یا ذخیره داده از روی دیسک است. همگی آن‌ها یک پنجره دیالوگ را فراخوانی می‌کنند تا کاربر بتواند به آسانی فایل‌های مورد نظر خود را انتخاب کرده و یا مسیری را برای ذخیره داده برگزیند. کلید CleanForm همگی اطلاعات فرم را پاک می‌کند و فضای اشغال شده توسط متغیرهای مورد استفاده را به سیستم بازمی‌گرداند.

در قسمت Summary از فرم، خلاصه‌ای از اطلاعات و گنجایش تصویر انتخاب شده ذکر می‌شود، همچنین در این قسمت تعداد کاراکترهای درج شده در Textbox را نیز می‌توان مشاهده کرد.

۵) فصل پنجم: نتایج تجربی

پس از ارائه و معرفی هر الگوریتمی این سوال پیش می‌آید که آیا الگوریتم جدید بهتر از روش‌های قبلی است یا خیر؟ برای تحقیق این امر، برخی ارزیابی‌ها و فرمول‌های استاندارد بر روی الگوریتم پیاده می‌شود. نتایج بدست آمده با نتایج بدست آمده روش‌های قبلی مقایسه شده و بدین ترتیب کارایی الگوریتم جدید سنجیده می‌شود.

در این فصل قصد داریم با اعمال ورودی‌های استاندارد به نرم‌افزار و اجرای الگوریتم PVM، خروجی آن‌را مشاهده کرده و با سایر روش‌های موجود، که در فصل ۲ معرفی شدند، مقایسه کنیم. در این ارزیابی مقدار PSNR ورودی و خروجی و مقدار داده تزریق شده را مبنای سنجش قرار خواهیم داد.

ورودی های استاندارد برای این منظور، تعدادی از عکس‌های استاندارد است که برای امور پردازش تصویر مورد استفاده قرار می‌گیرد. در این ارزیابی از تصاویر Lenna، Baboon، Airplane و Pepper استفاده شده‌است[۱].

برای محاسبه PSNR از نرم افزار MATLAB R2012b بهره خواهیم گرفت؛ همچنین از این نرم افزار برای رسم هیستوگرام پراکندگی رنگ نیز استفاده خواهیم کرد. مقایسه هیستوگرام‌های تصویر اولیه و تصویر نهایی، شمایی کلی از تغییرات تصویر بدست می‌دهد، به همین دلیل مقایسه هیستوگرام‌ها معیار درستی برای ارزیابی الگوریتم نخواهد بود. به همین منظور از معیار سیگنال به نویز یا PSNR بهره خواهیم برد.

ارزیابی الگوریتم را برای دوحالت مختلف انجام خواهدشد، یکی از حالات پنهان‌نگاری متن در عکس، و دیگری پنهان‌نگاری عکس استگانوگرافی شده در عکس پوششی است. در واقع در استگانوگرافی تصویر در تصویر دو مرحله استگانوگرافی متوالی انجام شده و نتایج در ادامه آورده شده است. ابتدا به پنهان‌نگاری استاندارد متن در تصویر پرداخته خواهد شد و در ادامه پنهان‌نگاری تصویر در تصویر مورد ارزیابی قرار خواهد گرفت.

۵-۱) محاسبه PSNR

نسبت سیگنال پیک به سیگنال نویز[۲] یک اصطلاح مهندسی است که میزان قدرت سیگنال اصلی را به سیگنال دارای نویز محاسبه می‌کند. از آنجایی که بیشتر سیگنال‌ها محدوده گسترده ای دارند، این نسبت را بوسیله لگاریتم نشان می‌دهند.

معمولا از PSNR برای محاسبه خطا در فشرده سازی استفاده می‌کنند. در این فصل منظور از سیگنال اصلی همان تصویر اصلی و منظور از سیگنال نویز، تصویر نهایی پس از اعمال پنهان‌نگاری است. مولفه اساسی محاسبه این معیار مربع میانگین خطاها[۳] یا MSE است، برای محاسبه MSE داریم:

$$MSE = \frac{1}{m\,n}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2 \qquad (۴)$$

در این فرمول، I(i,j) مقدار یک پیکسل خاص از تصویر اصلی و K(i,j) مقدار یک پیکسل خاص از تصویر نهایی است. مقادیر کل اختلاف پیکسل‌ها با هم جمع شده و با تقصیم بر مقدار mn، میزان میانگین بدست می‌آید.

---

[۱] تصاویر استاندارد از وبسایت http://sipi.usc.edu برگرفته شده‌است.

[۲] Peak signal-to-noise ratio

[۳] Mean Squared Error

در ادامه برای محاسبه PSNR داریم:

$$PSNR = 10.\log_{10}(\frac{MAX_I^2}{MSE}) \quad (۵)$$

در اینجا، متغیر MAX حداکثر مقدار ممکن برای هر رنگ تصویر است. تصاویر مورد استفاده ۲۴بیتی هستند پس برای هر رنگ، هشت بیت اختصاص می‌یابد؛ پس حداکثر مقدار برابر ۲۵۵ می‌باشد. به طور کلی در مسایل PCM خطی که از B بایت داده استفاده می‌کنند، حداکثر مقدار ممکن برابر 2^B-1 است.

خروجی تابع لگاریتمی فوق را با واحد dB بیان می‌کنند. دسی بل یک واحد لگاریتمی برای بیان نسبت یک کمیت فیزیکی (معمولاً توان یا شدت) به یک مقدار مرجع مشخص است. که همانطور که اشاره شد واحد مرجع در تصاویر، حداکثر مقدار رنگ هر پیکسل است.

بر این اساس، طبق دستورات نرم افزار MATLAB، توالی کدی را برای محاسبه PSNR دو تصویر پیاده سازی کردیم. متغیر in ماتریسی از مقادیر رنگی تصویر اولیه و متغیر out ماتریسی از مقادیر رنگی تصویر نهایی را در خود دارند. به منظور دقت بیشتر، برای هر رنگ MSE مخصوص آن رنگ را محاسبه کرده‌ایم و در پایان MSE کل را برابر میانگین MSEها قرار داده‌ایم.

```
In = imread('lenna.png');
Out = imread('lennaOut.jpg');
[rows columns ~] = size(In);
mseRImage = (double(In(:,:,1)) - double(Out (:,:,1))) .^ 2;
mseGImage = (double(In(:,:,2)) - double(Out (:,:,2))) .^ 2;
mseBImage = (double(In(:,:,3)) - double(Out (:,:,3))) .^ 2;
mseR = sum(sum(mseRImage)) / (rows * columns);
mseG = sum(sum(mseGImage)) / (rows * columns);
mseB = sum(sum(mseBImage)) / (rows * columns);
mse = (mseR + mseG + mseB)/3;
PSNR_Value = 10 * log10( 255^2 / mse);
```

دستورات فوق دو تصویر با نام‌های lenna.png و lennaOut.png را از ورودی خوانده و در دوماتریس ذخیره می‌کند، در ادامه MSE دو تصویر محاسبه شده و در پایان مقدار PSNR را در متغیر PSNR_value ذخیره می‌کند.

برای انجام آزمایش، دو فایل txt استانداردی شامل تمامی کاراکترهای اسکی از اسکی ۳۲ تا اسکی ۱۲۶ تولید شد. یکی از فایل‌ها با نام text100.txt، در کل ۱۰۳۱۱۱ کاراکتر را شامل می‌شد. این در حالیست که هر کدام از تصاویر استاندارد می‌توانستند ۱۳۱۰۶۹ کاراکتر را در خود جای دهند. فایل دیگر با نام text127.txt، شامل ۱۳۱۰۴۹ کاراکتر بود که تقریبا کل ظرفیت تصویر را پر می‌کرد.

عمل پنهان‌نگاری برای چهار تصویر معرفی شده انجام شد. نتایج را در جدول صفحه بعد مشاهده می‌کنید.

جدول(۱). نتایج آزمایش پنهان‌نگاری متن در تصویر

نام تصویر	حجم داده تزریق شده KB	PSNR(dB)	حجم تصویر قبل از تزریق KB	**حجم تصویر بعد از تزریق KB**
Lenna	۱۰۰	۴۷/۹۰۲۷	۷۶۶	۷۵۲
Airplane	۱۰۰	۴۷/۹۱۴۲	۶۲۵	۶۰۷
Baboon	۱۲۷	۴۶/۸۶۵۸	۸۳۱	۸۲۱
Pepper	۱۲۷	۴۶/۸۵۶۷	۷۷۸	۷۵۹

با نگاهی به ظرفیت تصاویر قبل و بعد از پنهان‌نگاری نتیجه می‌شود این عمل تاثیری در افزایش حجم عکس اولیه ندارد، زیرا با توجه به ذات روش PVM، داده‌ای به تصویر اضافه نمی‌شود بلکه داده‌های موجود را با ترتیب خاصی مرتب می‌کند، بنابراین تاثیری در حجم تصویر نهایی ندارد.

مقدار PSNR، بطور میانگین ۴۷dB برای روش پیشنهادی بدست آمد. حال به مقایسه ارزیابی‌های بدست آمده با روش‌های پیشین می‌پردازیم

جدول(۲). مقایسه با سایر روش‌ها

تصویر	روش وو و تسای (PVD)		روش اختلاف TPVD		**روش PVM**	
	داده تزریقی(B)	PSNR(dB)	داده تزریقی(B)	PSNR(dB)	داده تزریقی(B)	PSNR(dB)
Lenna	۵۱/۲۱۹	۴۱/۷۹	۷۵/۸۳۶	۳۸/۸۹	۱۰۳/۱۱۱	۴۷/۹۰۲۷
Airplane	۵۷,۱۴۶	۳۷/۹۰	۸۲,۴۰۷	۳۳/۹۳	۱۰۳/۱۱۱	۴۷/۹۱۴۲
Baboon	۵۱,۲۲۴	۴۰/۶۰	۷۶,۳۵۲	۳۸/۷۰	۱۳۱/۰۴۹	۴۶/۸۶۵۸
Pepper	۵۰,۹۰۷	۴۰/۹۷	۷۵,۵۷۹	۳۸/۵۰	۱۳۱/۰۴۹	۴۶/۸۵۶۷

همانطور که مشـاهده می‌شود مقایسه روش PVM با روش PVD و روش TPVD، نتیجه داد که روش پیشنهادی بیشترین داده تزریق شده را به همراه کمترین افت کیفیت تصویر دارد. نمودار هیستوگرام آنالیز تصویر اولیه و تصویر نهایی نیز برای نتایج، در ادامه رسم شده است.

مقایسه این نمودار ها نشان می‌دهد که تغییر میان دو هیستوگرام تصویر اصلی و هیستوگرام تصویر نهایی بسیار ناچیز است. و این باعث می شود پنهان‌یاب نتواند به راحتی پیغام نهفته در عکس نهایی را بیابد. روش جاری تحریف بسیار ناچیزی بر روی پیکسل‌های عکس نهایی نسبت به عکس اولیه ایجاد می کند و نیز امنیت بالایی را فراهم میکند زیرا مقادیر مختلفی از بیت‌ها در پالت های مختلف رنگی نهفته شده‌اند پس تشخیص اینکه چه تعداد بیت در داخل یک پیکسل نهفته شده است بسیار دشوار است. در نتیجه تحریف ارزش پیکسل‌ها در تصویر نهایی بسیار کم بوده و نتیجه این تحریف کم باعث کیفیت بسیار بالای تصویر نهایی می شود.

۵-۲) رسم هیستوگرام

پس از انجام استگانوگرافی بر روی یک تصویر نمی‌توان تفاوتی میان تصویر اولیه و تصویر نهایی تشخیص داد، به همین دلیل ابتدا هیستوگرام تصاویر را رسم کرده و هیستوگرام‌ها را مقایسه می‌کنیم. یکی از امکانات نرم افزار متلب، پردازش

تصویر است. در اینجا بوسیله نرم افزار متلب برای هریک از تصایر فوق هیستوگرام را رسم کرده و به مقایسه آن‌ها می پردازیم، ولی در ابتدا برخی دستورات لازم برای کار با تصاویر را ذکر می‌کنیم.

به منظور رسم هیستوگرام، ابتدا باید تصاویر رنگی موجود را به تصاویر سیاه و سفید تبدیل کنیم، سپس برای تصویر سیاه وسفید هیستوگرام مربوطه را رسم می‌کنیم.

```
in = imread('lenna.png');
```

دستور فوق اطلاعات تصویر lenna.png را در ماتریسی با نام in ذخیره می‌کند.

```
in_gary = rgb2gray(in);
```

دستور فوق تصویر رنگی ذخیره شده در متغیر in را پس از سیاه وسفید کردن در متغیر in_gray ذخیره می‌کند.

```
figure, imhist(in_gray);
```

در پایان با اجرای دستور فوق پنجره‌ای نمایش داده می‌شود که محتوای آن نمودار هیستوگرام مربوط به تصویر lenna.png است. همچنین به منظور درک بیشتر، می‌توان با اجرای کد زیر تصویر سیاه وسفید شده را مشاهده کرد.

```
figure, imshow(in_gray);
```

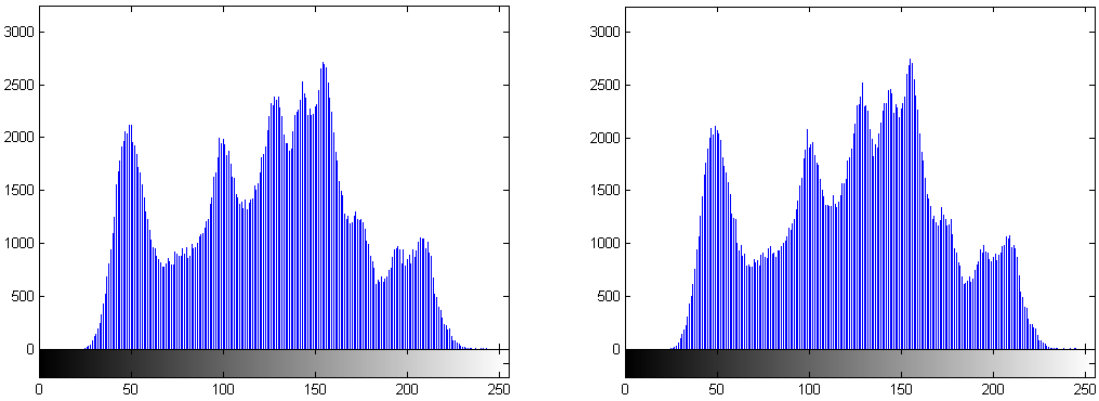
پس از اعمال کدهای فوق بر روی هشت تصویر مورد نظر، نتایج زیر بدست آمد:

شکل(۵-۱). تصویر Lenna

الف) قبل از استگانوگرافی

ب) بعد از استگانوگرافی

شکل(۵-۲). هیستوگرام **Lenna**

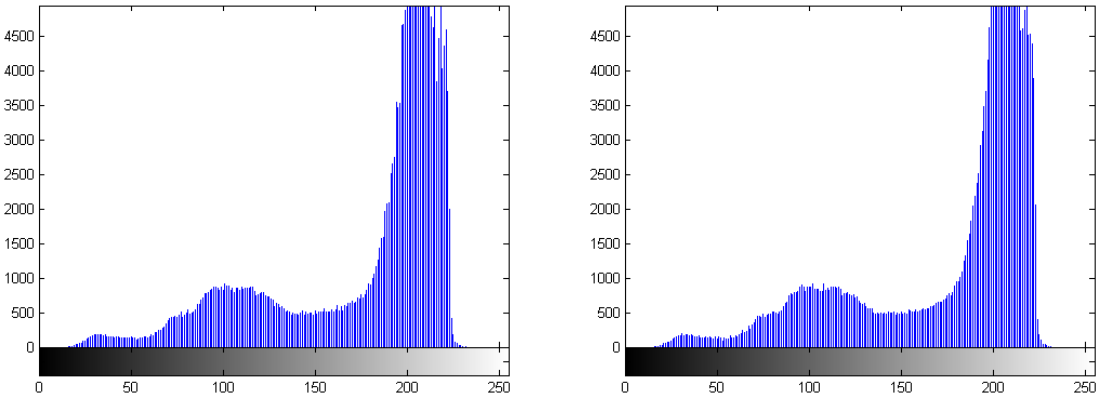


الف) قبل از استگانوگرافی ب) بعد از استگانوگرافی

شکل(۵-۳). تصویر **Airplane**

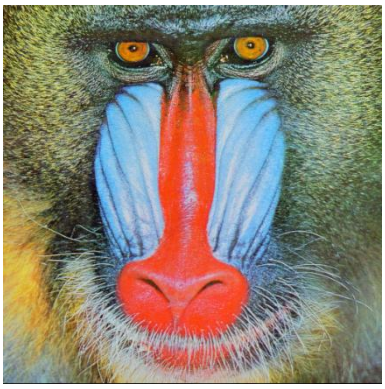
الف) قبل از استگانوگرافی ب) بعد از استگانوگرافی

شکل(۵-۴). هیستوگرام **Airplane**

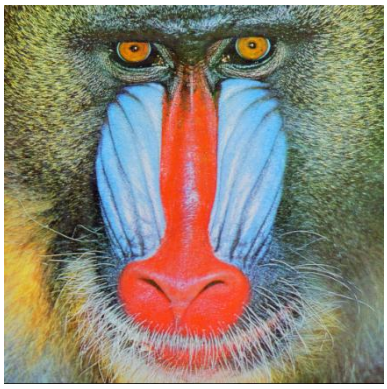


الف) قبل از استگانوگرافی ب) بعد از استگانوگرافی

شکل(۵-۵). تصویر **Baboon**

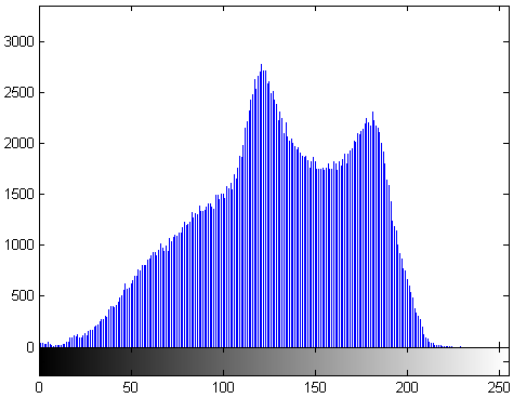


ب) بعد از استگانوگرافی

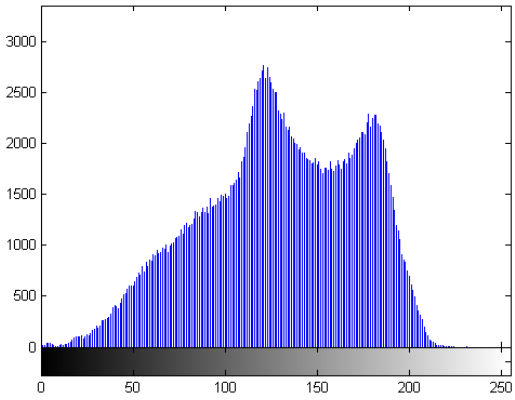


الف) قبل از استگانوگرافی

شکل(۵-۶).هیستوگرام **Baboon**



ب) بعد از استگانوگرافی



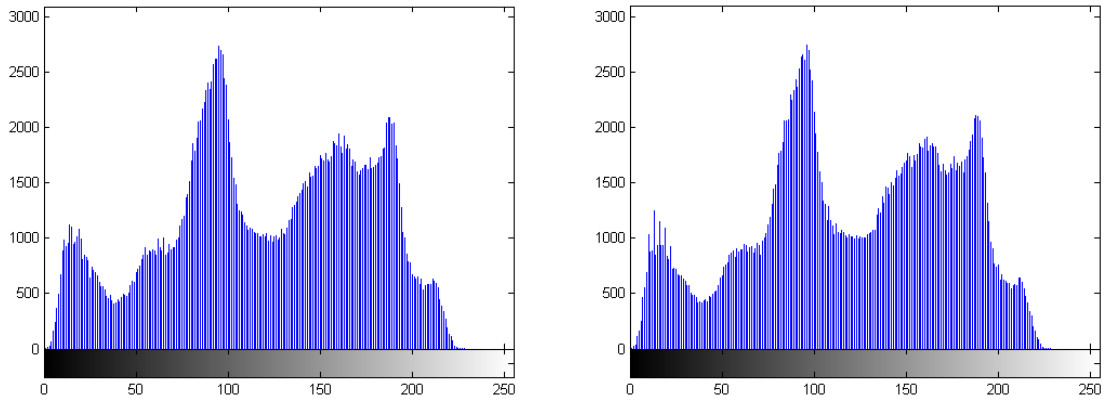
الف) قبل از استگانوگرافی

شکل(۵-۷). تصویر **Pepper**

ب) بعد از استگانوگرافی

الف) قبل از استگانوگرافی

شکل(۵-۸). هیستوگرام Pepper



الف) قبل از استگانوگرافی **ب) بعد از استگانوگرافی**

همانطور که از مقایسه تصاویر نتیجه می‌شود، مشاهده تغیر ناچیز در رنگ آن‌ها با چشم غیرممکن است، ولی با مقایسه هیستوگرام‌ها می‌توان به تغیرات اندک میان دو تصویر اولیه و نهایی پی برد. همین امر باعث می‌شود تشخیص تصویر پنهان‌نگاری شده بسیار دشوار شده و یا عملا غیر ممکن شود.

در نهایت پس از انجام ارزیابی و مقایسه میان روش‌های موجود، با صراحت می‌توان گفت روش پیشنهادی علاوه بر اینکه دارای امنیت بالایی است، حجم داده بیشتری را بدون کاهش محسوس کیفیت تصویر ارائه می‌کند.

در ادامه به ارزیابی پنهان‌نگاری تصویر در تصویر خواهیم پرداخت. از آنجایی که نمی‌توان تصاویر استاندرد فوق را در هم مخفی کرد، ابتدا یکی از تصاویر را به اندازه مورد نیاز کوچک کرده، سپس در دیگری مخفی می‌کنیم. برای این منظور تصویر Lenna با ابعاد استاندارد ۵۱۲×۵۱۲ به تصویری با ابعاد ۱۲۸×۱۲۸ تقلیل یافت، سپس تا حداکثر ظرفیت آن با متن پنهان‌نگاری شده و خروجی را به عنوان عکس مخفی در تصویر Airplane به عنوان عکس پوششی مخفی خواهیم‌کرد.

جدول زیر نتایج آزمایش را به همراه دارد.

جدول(۳). نتایج آزمایش پنهان‌نگاری تصویر در تصویر

نام تصویر	حالت	حجم داده تزریق شده KB	PSNR(dB)	حجم تصویر قبل از تزریق KB	**حجم تصویر بعد از تزریق KB**
Lenna(128×128)	متن در تصویر	۸/۱۴۸	۴۶/۸۱۰۶	۴۰	۴۹
Airplane	تصویر در تصویر	۴۹	۵۱/۰۴۳۶	۶۲۵	۶۲۵

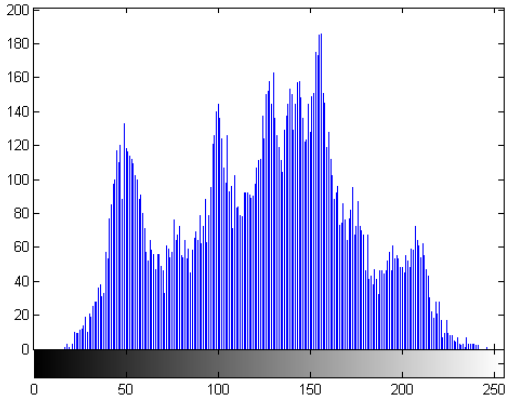
تصاویر صفحه بعد خروجی نرم افزار را برای پنهان‌نگاری متوالی نشان می‌دهد

شکل(۵-۹). تصویر کوچک شده **Lenna**

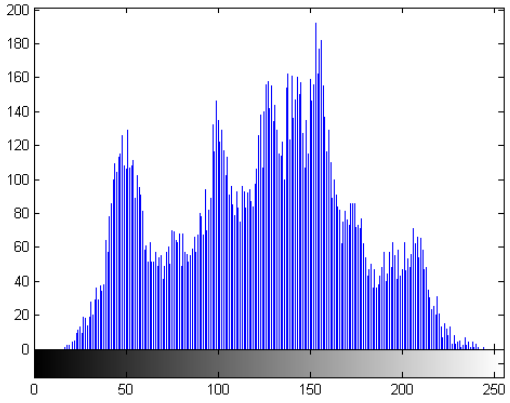
الف) قبل از پنهان‌نگاری

ب) بعد از پنهان‌نگاری

شکل(۵-۱۰). هیستوگرام تصویر کوچک شده **Lenna**



الف)قبل از پنهان‌نگاری



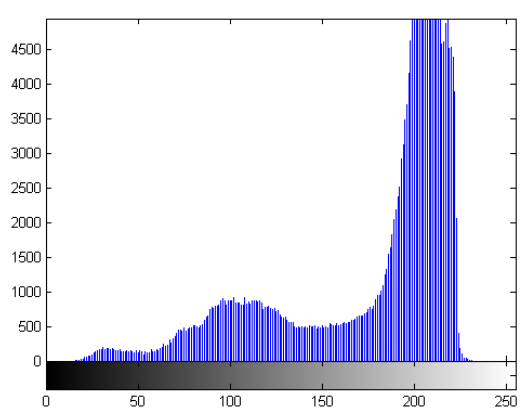
ب)بعد از پنهان‌نگاری

شکل(۵-۱۱). تصویر **Airplane** به عنوان عکس پوششی **Lenna**

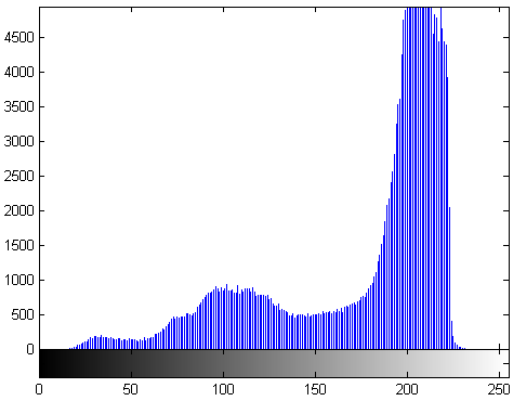
الف) قبل از از پنهان‌نگاری

ب) بعد از پنهان‌نگاری و حاوی تصویر پنهان‌نگاری شده **Lenna**

شکل(۵-۱۲). هیستوگرام تصویر Airplane به عنوان عکس پوششی Lenna



الف) قبل از پنهان‌نگاری



ب) بعد از از پنهان‌نگاری و حاوی تصویر پنهان‌نگاری شده Lenna

مشاهده نتایج PSNR و آنالیزهای هیستوگرام نشان می‌دهد پنهان‌نگاری متوالی، همچنان کارایی خود را داشته و تغییری در کیفیت تصویر خروجی ندارد؛ پس به عنوان روشی امن تر می‌توان از پنهان‌نگاری متوالی استفاده کرد. استفاده از این روش شخص پنهان‌یاب را نیز دچار سردرگمی خواهدکرد. زیرا با استخراج داده های عکس پوششی با توالی‌ای از اعداد مواجه خواهدشد که معنی خاصی نمی‌دهند.

۶) فصل ششم: نتیجه گیری

در این پایان‌نامه روش اصلاح ارزش پیکسل‌ها برای مخفی سازی داده محرمانه با استفاده از فرمول محاسبه باقیمانده تقسیم مقدار پیکسل بر رقم متناظر آن در داده محرمانه که در مبنای سه می‌باشد انجام شد. همچنین این روش تضمین می‌کند مقدار پیکسل‌ها در عکس نهایی از بازه صفر تا ۲۵۴ تجاوز نمی کند. در روش اختلاف ارزش پیکسل‌ها برای جایگذاری یک رقم از داده محرمانه نیاز به دو پیکسل متوالی بود اما روش اصلاح ارزش پیکسل‌ها برای جایگذاری یک رقم از داده محرمانه تنها به یک پیکسل نیاز دارد.

روش پیشنهادی ظرفیت و امنیت بالاتری را برای عکس های رنگی نسبت به روش‌های پیشین ارائه می‌کند و همچنین کیفیت ظاهری بهتری را برای تصویر نهایی به دست می‌دهد و نهایتا روش ارائه شده پیغام محرمانه مخفی شده را به راحتی و بدون جدول بازه‌های رنگ استخراج می‌کند.

از ایرادات وارد بر برنامه پیاده سازی شده می‌توان به مدت زمان اجرای الگوریتم اشاره کرد. این امر برای حالاتی که تصویر بزرگ باشد و کل عکس پنهان‌نگاری شود، بسیار به طول خواهد انجامید، ولی چنانچه تصویر کوچک باشد و یا متن محرمانه کوتاه باشد بسیار خوب عمل می‌کند.

هر چند الگوریتم پیشنهادی به صورت ذاتی از امنیت بالایی برخوردار است ولی می‌توان با رمز نگاری داده محرمانه امنیت مضاعفی برای آن فراهم آورد. با این اوصاف چنانچه پنهان‌یاب بتواند عکس پوششی را پنهان‌یابی کرده و اطلاعات آن را استخراج کند، با رشته‌ای از متن رمز نگاری شده مواجه می‌شود که بدون داشتن کلید بلا استفاده خواهد بود.

چنانچه نخواهیم از رمزنگاری داده محرمانه بهره ببریم، می‌توان پس از پنهان‌نگاری متن در تصویر، تصویر نهایی را به عنوان عکس مخفی در عکس پوششی دیگری مخفی کنیم؛ در این حالت شخص پنهان‌یاب پس از استخراج داده مخفی، با توالی‌ای از اعداد مواجه می‌شود که تقریبا نامفهوم است. بنابراین انجام دو پنهان‌نگاری متوالی بسیار امن‌تر خواهدبود.

مراجع

[1] V.Nagaraj, Dr. V. Vijayalakshmi and Dr. G. Zayaraz "On Color Image Steganography based on Pixel Value Modification Method Using Modulus Function". IERI Procedia.Volume 4, 2013, Pages 17–24 (EECS 2013)

[2] 2013 International Conference on Electronic Engineering and Computer Science (EECS 2013)

[3] R. J. Anderson and F. A. P. Petitcolas, "On the limits of steganography". IEEE Journal on Selected Areas in Communications, pages 474-481, 1998.

[4] C.K. Chan, L.M. Cheng, "Hiding data in images by simple LSB substitution", pattern recognition vol. 37, Issue 3,pp. 469-474, 2004.

[5] D.C. Wu, and W.H. Tsai, "A Steganographic method for images by pixel-value differencing", Pattern Recognition Letters, Vol. 24, pp. 1613-1626, 2003.

[6] H.C. Wu, N.I. Wu, C.S. Tsai, and M.S. Hwang, "Image steganographic scheme based on pixel-value differencing and LSB replacement methods," IEE Proceedings on Vision, Image and Signal Processing, Vol. 152, No. 5, pp. 611-615, 2005.

[7] C.M. Wang, N.I. Wu, C.S. Tsai, M.S. Hwang, "A high quality steganography method with pixel-value differencing and modulus function", Journal of System Software, Vol.81, No.1, pp.150-158, 2008.

[8] K.C. Chang, C.P. Chang, P.S. Huang, and T.M. Tu, "A Novel Image Steganographic Method Using Tri- way Pixel-Value Differencing," Journal of Multimedia, Vol. 3, No. 2, pp.37-44, June 2008.

[9] J.K. Mandal, Debashis Das "Steganography Using Adaptive Pixel Value Differencing (APVD) for Gray Images through Exclusion of Underflow/Overflow", Computer Science & Information Series, ISBN: 978-1-921987-03-8, pp. 93-102, 2012.

[10] M. Padmaa and Dr. Y. Venkataramani, "ZIG-ZAG PVD – A Nontraditional Approach," International Journal of Computer Applications, Vol. 5, No. 7, pp. 5-10, August 2010.

[11] Ki-Hyun Jung and Kee-Young Yoo, "Improved Exploiting Modification Direction Method by Modulus Operation," International Journal of Signal Processing, Image Processing and Pattern, Vol. 2, No.1, March, 2009.

ضمائم

متن کامل کد

Form1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;
namespace Steganography
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        OpenFileDialog openFrom = new OpenFileDialog();
        SaveFileDialog saveTo = new SaveFileDialog();

//########################################################################
#####
        private void BrowseInputPic_Click(object sender, EventArgs e)
        {
            openFrom.Filter = "Text Files (.png)|*.png|All Files
(*.*)|*.*";
            openFrom.FilterIndex = 1;
            if (openFrom.ShowDialog() == DialogResult.OK)
            {
                variables.filePathWrite = openFrom.FileName;
                InputPathLabel.Text = variables.filePathWrite;
                variables.test = true;
                Bitmap temp = new Bitmap(variables.filePathWrite);
```

```
                labelPhotoSize.Text = Convert.ToString(temp.Width) + "
x " + Convert.ToString(temp.Height);
                labelTotalPixel.Text = Convert.ToString(temp.Height *
temp.Width);
                labelMaxCharToSave.Text = Convert.ToString(((temp.Width
* temp.Height) / 2) - 3);
                pictureBox3.Visible = true;
            }
            else
                InputPathLabel.Text = "No File Selected";
        }

//##################################################################
#####
        private void StartReading_Click(object sender, EventArgs e)
        {
            openFrom.Filter = "Text Files (.png)|*.png|All Files
(*.*)|*.*";
            openFrom.FilterIndex = 1;
            if (openFrom.ShowDialog() == DialogResult.OK)
            {
                labelReadDataPhotoPath.Text = openFrom.FileName;
                variables.filePath = openFrom.FileName;
                InputPathLabel.Text = variables.filePath;
                string base3T = null;
                string codeLength = null;
                int codeL = 0;
                int red, green, blue, x, y, z;
                Bitmap myBitmap = new Bitmap(variables.filePath);
                Color pixelcolor;
                int counter = 0;
                for (y = 0; y < 6; y++)
                {
                    pixelcolor = myBitmap.GetPixel(0, y);
                    codeLength += (pixelcolor.R % 3).ToString();
                    codeLength += (pixelcolor.G % 3).ToString();
                    codeLength += (pixelcolor.B % 3).ToString();
                }
                codeL = Functions.ConvertToBase10(codeLength) + 6;
                for (x = 0; x < myBitmap.Width; x++)
                    for (y = 0; y < myBitmap.Height; y++)
                    {
                        pixelcolor = myBitmap.GetPixel(x, y);
                        red = pixelcolor.R;
                        base3T += (red % 3).ToString();
```

```
                        green = pixelcolor.G;
                        base3T += (green % 3).ToString();
                        blue = pixelcolor.B;
                        base3T += (blue % 3).ToString();
                        counter++;
                        if (counter == codeL)
                            goto done;
                    }
            done:
                string[] base3t = new string[(base3T.Length) / 6];
                z = 0;
                string key = null;
                for (x = 18; x < base3T.Length; x += 6)
                {
                    base3t[z] =
Functions.ConvertToBase10(base3T.Substring(x, 6)).ToString();
                    key +=
char.ConvertFromUtf32(Functions.ConvertToBase10(base3T.Substring(x,
6))).ToString();
                    z++;
                }
                SecretData.Text = key;
                labelTotalHidden.Text = key.Length.ToString();
            }// end ResultDialog IF
            else
                SecretData.Text = "ERROR, No File Selected";
        }

//#####################################################################
#####
        private void StartHidding_Click(object sender, EventArgs e)
        {
            if (variables.test == true)
            {
                variables.MyBitmap = new
Bitmap(variables.filePathWrite);
                Color pixelcolor;
                string secretKey = SecretData.Text;
                string base3T = null;
                string base3;
                string format;
                int calculateMaxData;
                calculateMaxData = ((variables.MyBitmap.Width *
variables.MyBitmap.Height) / 2) - 3;
                if (SecretData.TextLength < calculateMaxData)
```

```
                {
                    format = Functions.ConvertToBase3(secretKey.Length
* 2, 3).PadLeft(18, '0');
                    base3T += format;
                    for (int t = 0; t < secretKey.Length; t++)
                    {
                        int ascii = int.Parse(new
string(Encoding.ASCII.GetBytes(secretKey.Substring(t, 1)).SelectMany(b
=> b.ToString()).ToArray()));
                        base3 = Functions.ConvertToBase3(ascii, 3);
                        var padding = base3.PadLeft(6, '0');
                        base3T += padding;
                    }
                    int x, y, red, green, blue;
                    int temp = 0;
                    for (x = 0; x < variables.MyBitmap.Width; x++)
                        for (y = 0; y < variables.MyBitmap.Height; y++)
                        {
                            pixelcolor = variables.MyBitmap.GetPixel(x,
y);
                            red =
Functions.setPixel(int.Parse(base3T.Substring(temp, 1)), pixelcolor.R);
                            temp++;
                            green =
Functions.setPixel(int.Parse(base3T.Substring(temp, 1)), pixelcolor.G);
                            temp++;
                            blue =
Functions.setPixel(int.Parse(base3T.Substring(temp, 1)), pixelcolor.B);
                            temp++;
                            variables.MyBitmap.SetPixel(x, y,
Color.FromArgb(red, green, blue));
                            if (temp == base3T.Length)
                                goto done;
                        }
                done:
                    saveTo.Filter = "txt files (*.png)|*.png";
                    saveTo.FilterIndex = 2;
                    saveTo.RestoreDirectory = true;
                    if (saveTo.ShowDialog() == DialogResult.OK)
                    {
                        variables.MyBitmap.Save(saveTo.FileName,
System.Drawing.Imaging.ImageFormat.Png);
                        labelHidedImagePath.Text = saveTo.FileName;
                        labelStatus.Text = "Succeed !";
                    }
                    else
```

```
                    variables.MyBitmap = null;
                }//if (SecretData.TextLength < calculateMaxData)
                else
                    MessageBox.Show("SecretData is to large!\nPlease
select a larger photo or reduce SecretData length");
            }//if (variables.test == true)
            else
                MessageBox.Show("ERROR, Choose a Photo");
        }

//#####################################################################
#####
        private void TextfileBrowse_Click(object sender, EventArgs e)
        {
            openFrom.Filter = "Text Files (.txt)|*.txt|All Files
(*.*)|*.*";
            openFrom.FilterIndex = 1;
            if (openFrom.ShowDialog() == DialogResult.OK)
            {
                labelReadDataFromTXT.Text = openFrom.FileName + "
Selected";
                variables.filePath = openFrom.FileName;
                string text =
System.IO.File.ReadAllText(variables.filePath);
                SecretData.Text = text;
                pictureBox4.Visible = true;
            }
        }

//#####################################################################
#####
        private void saveDataBut_Click(object sender, EventArgs e)
        {
            saveTo.Filter = "txt files (*.txt)|*.txt";
            saveTo.FilterIndex = 2;
            saveTo.RestoreDirectory = true;
            if (saveTo.ShowDialog() == DialogResult.OK)
            {
                labelSaveDataTXTafterReadPath.Text = saveTo.FileName;
                System.IO.File.WriteAllText(saveTo.FileName,
SecretData.Text);
            }
        }

//#####################################################################
#####
```

```
        private void ClearForm_Click(object sender, EventArgs e)
        {
            SecretData.Text = null;
            variables.filePath = null;
            variables.filePathWrite = null;
            variables.test = false;
            variables.P2PSecret = false;
            variables.P2PMain = false;
            InputPathLabel.Text = null;
            labelStatus.Text = null;
            labelTotalPixel.Text = null;
            labelTotalHidden.Text = null;
            labelSaveDataTXTafterReadPath.Text = null;
            labelReadDataPhotoPath.Text = null;
            labelReadDataFromTXT.Text = null;
            labelPhotoSize.Text = null;
            labelMaxCharToSave.Text = null;
            labelHidedImagePath.Text = null;
            labelCharInTextbox.Text = null;
            pictureBox1.Visible = false;
            pictureBox2.Visible = false;
            pictureBox3.Visible = false;
            pictureBox4.Visible = false;
            checkBox1.Checked = false;
        }
        private void SecretData_TextChanged(object sender, EventArgs e)
        {
            labelCharInTextbox.Text = SecretData.TextLength.ToString();
        }
        private void labelHidedImagePath_Click(object sender, EventArgs
e)
        {
        }
        private void labelReadDataFromTXT_Click(object sender,
EventArgs e)
        {
        }

//#####################################################################
#####
        private void button2_Click(object sender, EventArgs e)
        {
            if (variables.P2PMain && variables.P2PSecret)
            {
```

```
                Bitmap BitmapOut = new
Bitmap(variables.filePathSecretPhoto);
                variables.MyBitmap = new Bitmap(variables.filePath);
                Color ColorPixelIn;
                Color ColorPixelOut;
                string ImageValuesInbase3 =
Functions.ConvertToBase3(variables.MyBitmap.Width *
variables.MyBitmap.Height * 6, 3).ToString().PadLeft(18, '0');
                MessageBox.Show(ImageValuesInbase3);
                ImageValuesInbase3 +=
Functions.ConvertToBase3(variables.MyBitmap.Width, 3).PadLeft(9, '0');
                MessageBox.Show(ImageValuesInbase3);
                ImageValuesInbase3 +=
Functions.ConvertToBase3(variables.MyBitmap.Height, 3).PadLeft(9, '0');
                MessageBox.Show(ImageValuesInbase3);
                for (int i = 0; i < variables.MyBitmap.Width; i++)
                    for (int j = 0; j < variables.MyBitmap.Height; j++)
                    {
                        ColorPixelIn = variables.MyBitmap.GetPixel(i,
j);
                        ImageValuesInbase3 +=
Functions.ConvertToBase3(ColorPixelIn.R, 3).PadLeft(6, '0');
                        ImageValuesInbase3 +=
Functions.ConvertToBase3(ColorPixelIn.G, 3).PadLeft(6, '0');
                        ImageValuesInbase3 +=
Functions.ConvertToBase3(ColorPixelIn.B, 3).PadLeft(6, '0');
                    }
                int x, y, red, green, blue, temp = 0;
                for (x = 0; x < BitmapOut.Width; x++)
                    for (y = 0; y < BitmapOut.Height; y++)
                    {
                        ColorPixelOut = BitmapOut.GetPixel(x, y);
                        red =
Functions.setPixel(int.Parse(ImageValuesInbase3.Substring(temp, 1)),
ColorPixelOut.R);
                        temp++;
                        green =
Functions.setPixel(int.Parse(ImageValuesInbase3.Substring(temp, 1)),
ColorPixelOut.G);
                        temp++;
                        blue =
Functions.setPixel(int.Parse(ImageValuesInbase3.Substring(temp, 1)),
ColorPixelOut.B);
                        temp++;
                        BitmapOut.SetPixel(x, y, Color.FromArgb(red,
green, blue));
                        if (temp == ImageValuesInbase3.Length)
                            goto done;
```

```
                }
            done:
                saveTo.Filter = "txt files (*.png)|*.png";
                saveTo.FilterIndex = 2;
                saveTo.RestoreDirectory = true;
                if (saveTo.ShowDialog() == DialogResult.OK)
                {
                    BitmapOut.Save(saveTo.FileName,
System.Drawing.Imaging.ImageFormat.Png);
                    labelHidedImagePath.Text = saveTo.FileName;
                    labelStatus.Text = "Succeed !";
                }
            }
            else
                MessageBox.Show("Select
Photos.","ERROR",MessageBoxButtons.OK,MessageBoxIcon.Error);
        }

//#########################################################################
#####
        private void checkBox1_CheckedChanged(object sender, EventArgs
e)
        {
            if (checkBox1.Checked == true)
            {
                buttonBrowseSecretPhoto.Enabled = true;
                buttonHidePhotoInPhoto.Enabled = true;
                buttonBrowseMainPhoto.Enabled = true;
            }
            else
            {
                buttonBrowseSecretPhoto.Enabled = false;
                buttonHidePhotoInPhoto.Enabled = false;
                buttonBrowseMainPhoto.Enabled = false;
            }
        }

//#########################################################################
#####
        private void button1_Click(object sender, EventArgs e)
        {
            openFrom.Filter = "Text Files (.png)|*.png|All Files
(*.*)|*.*";
            openFrom.FilterIndex = 1;
            openFrom.Title = "Select Photo ...";
```

```
            if (openFrom.ShowDialog() == DialogResult.OK)
            {
                variables.filePath = openFrom.FileName;
                string base3T = null;
                string codeLength = null;
                string PicSizeW = null;
                string PicSizeH = null;
                int codeL = 0;
                int x, y, red, green, blue;
                int[] c = new int[4];
                Bitmap myBitmap = new Bitmap(variables.filePath);
                Color pixelcolor;
                int counter = 0;


                for (x = 0; x < myBitmap.Width; x++)
                    for (y = 0; y < myBitmap.Height; y++)
                    {
                        if (counter < 6)
                        {
                            pixelcolor = myBitmap.GetPixel(x, y);
                            codeLength += (pixelcolor.R %
3).ToString();
                            codeLength += (pixelcolor.G %
3).ToString();
                            codeLength += (pixelcolor.B %
3).ToString();
                        }
                        if ((counter >= 6) && (counter < 9))
                        {
                            pixelcolor = myBitmap.GetPixel(x, y);
                            PicSizeW += (pixelcolor.R % 3).ToString();
                            PicSizeW += (pixelcolor.G % 3).ToString();
                            PicSizeW += (pixelcolor.B % 3).ToString();
                        }
                        if ((counter >= 9) && (counter < 12))
                        {
                            pixelcolor = myBitmap.GetPixel(x, y);
                            PicSizeH += (pixelcolor.R % 3).ToString();
                            PicSizeH += (pixelcolor.G % 3).ToString();
                            PicSizeH += (pixelcolor.B % 3).ToString();
                        }
                        if (counter == 12)
                            goto endFor;
```

```
                    counter++;
                }
            endFor: ;
                x = Functions.ConvertToBase10(PicSizeW);
                y = Functions.ConvertToBase10(PicSizeH);
                Bitmap BitmapOut = new Bitmap(x, y);
                int check = 0;
                codeL = Functions.ConvertToBase10(codeLength) + 12;
                for (x = 0; x < myBitmap.Width; x++)
                    for (y = 0; y < myBitmap.Height; y++)
                    {
                        if (check >= 12)
                        {
                            pixelcolor = myBitmap.GetPixel(x, y);
                            base3T += (pixelcolor.R % 3).ToString();
                            base3T += (pixelcolor.G % 3).ToString();
                            base3T += (pixelcolor.B % 3).ToString();
                            if (check + 1 == codeL)
                                goto done;
                        }
                        check++;
                    }
            done:
                counter = 0;
                check = 0;
                for (x = 0; x < BitmapOut.Width; x++)
                    for (y = 0; y < BitmapOut.Height; y++)
                    {
                        red =
Functions.ConvertToBase10(base3T.Substring(counter, 6));
                        counter += 6;
                        green =
Functions.ConvertToBase10(base3T.Substring(counter, 6));
                        counter += 6;
                        blue =
Functions.ConvertToBase10(base3T.Substring(counter, 6));
                        counter += 6;
                        BitmapOut.SetPixel(x, y, Color.FromArgb(red,
green, blue));
                    }
                saveTo.Filter = "PNG files (*.png)|*.png";
                saveTo.FilterIndex = 2;
                saveTo.RestoreDirectory = true;
                if (saveTo.ShowDialog() == DialogResult.OK)
```

```
                {
                    BitmapOut.Save(saveTo.FileName,
System.Drawing.Imaging.ImageFormat.Png);
                    labelHidedImagePath.Text = saveTo.FileName;
                    labelStatus.Text = "Succeed !";
                }
                else
                  MessageBox.Show("Hidden Photo is not
Saved","ERROR",MessageBoxButtons.OK,MessageBoxIcon.Error);
            }
            else
                MessageBox.Show("Photo Not Selected", "ERROR",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        private void buttonBrowseSecretPhoto_Click(object sender,
EventArgs e)
        {
            openFrom.Filter = "Text Files (.png)|*.png|All Files
(*.*)|*.*";
            openFrom.FilterIndex = 1;
            openFrom.Title = "Select Secret Photo ...";
            if (openFrom.ShowDialog() == DialogResult.OK)
            {
                variables.filePath = openFrom.FileName;
                pictureBox1.Visible = true;
                variables.P2PSecret = true;
            }
        }
        private void buttonBrowseMainPhoto_Click(object sender,
EventArgs e)
        {
            openFrom.Filter = "Text Files (.png)|*.png|All Files
(*.*)|*.*";
            openFrom.FilterIndex = 1;
            openFrom.Title = "Select Main Photo ...";
            if (openFrom.ShowDialog() == DialogResult.OK)
            {
                variables.filePathSecretPhoto = openFrom.FileName;
                pictureBox2.Visible = true;
                variables.P2PMain = true;
            }
        }
    }
}
```

6-1-1) Functions.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;


namespace Steganography
{
    class Functions
    {
        public static string ConvertToBase3(int value, int toBase)
        {
            if (toBase < 2 || toBase > 36) throw new
ArgumentException("toBase");
            if (value < 0) throw new ArgumentException("value");


            if (value == 0) return "0"; //0 would skip while loop


            string AlphaCodes = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";


            string retVal = "";


            while (value > 0)
            {
                retVal = AlphaCodes[value % toBase] + retVal;
                value /= toBase;
            }
            return retVal;
        }


        public static int ConvertToBase10(string value)
        {
            int valueLenth, retval = 0;
            valueLenth = value.Length;
            for (int x = 1; x <= valueLenth; x++)
                retval += (int)(int.Parse(value.Substring(x - 1, 1)) *
Math.Pow(3, valueLenth - x));
            return retval;
```

```
        }

        public static int setPixel(int secretDigit, int colorPixel)
        {
            if (colorPixel == 255)
                colorPixel--;
            string sw = (secretDigit - (colorPixel % 3)).ToString();
            switch (sw)
            {
                case "1":
                case "-1":
                case "0":
                    if (secretDigit < colorPixel % 3)
                        colorPixel--;
                    else if (secretDigit > colorPixel % 3)
                        colorPixel++;
                    break;

                case "2":
                case "-2":
                    if (secretDigit < colorPixel % 3)
                        colorPixel -= 2;
                    else if (secretDigit > colorPixel % 3)
                        colorPixel += 2;
                    break;
            }
            return colorPixel;
        }
    }
}
```

6-1-2) Variables.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;
using System.Drawing;
```

```
namespace Steganography
{
    class variables
    {
        public static string filePath;
        public static string filePathSecretPhoto;
        public static string filePathWrite;
        public static bool test = false;
        public static bool P2PMain = false;
        public static bool P2PSecret = false;
        public static Bitmap MyBitmap;

    }
}
```

Abstract

Steganography is the only answer for secure and secret communication. Existing methods in image steganography focus on increasing embedding capacity of secret data. According to existing methods, the experimental results indicate that two pixels are required for one secret digit embedding. In direction of improve the embedding size of secret data, a novel method of Pixel Value Modification (PVM) by modulus function is proposed. The proposed PVM method can embed one secret digit on one pixel of cover image. Thus, the proposed PVM method gives good quality of stego image. The experimental outputs validate that good visual perception of stego image with more secret data embedding capacity of stego image can be achieved by the proposed method.

Islamic Azad University
Tabriz branch
Faculty of Engineering
Department of Computer

B. Sc. Thesis

Color Image Steganography based on Pixel Value Modification Method Using Modulus Function

Supervisor:

Dr. Ali Asghar Pourhaji Kazem

By:

Hiwa Amiri

2014, June

Islamic Azad University
Tabriz branch
Faculty of Engineering
Department of Computer

B. Sc. Thesis

Color Image Steganography based on Pixel Value Modification Method Using Modulus Function

Supervisor:

Dr. Ali Asghar Pourhaji Kazem

By:

Hiwa Amiri

2014, June