الجمهورية اليمنية

وزارة التعليم العالي والبحث العلمي

جامعه صنعاء

كلية الحاسوب وتكنولوجيا المعلومات

بسم الله الرحمن الرحيم

نبذة عن علم إخفاء البيانات (STEGANOGRAPHY)

كيفية إخفاء البيانات في ملفات الصور، الصوت والفيديو

باشراف الدكتور: عبد الماجد الخليدي

اعداد الطالب: ابوبكر محسن الشهاري

٢٠١٨/١٢/١٩

مقدمة

أساس علم إخفاء البيانات(Steganography)

كلمة Steganography في الأساس مشتقة من كلمة يونانية تعني "الكتابة المخفية."

موضوع إرسال رسالة مخفية عن طريق حجب أن هناك شيء مرسل من الأساس هي طريقة (وفكرة) قديمة. ولها قصة تاريخية بدأت أيام الإمبراطورية اليونانية القديمة عندما كانت تكتب الرسائل على رؤوس العبيد آنذاك بعد حلق شعر العبد، حيث يكتب على رأسه رسالة سرية معينة، وعندما يعود شعره للنمو مرة أخرى تختبئ الرسالة السرية تحت شعره الكثيف، وعندها، يتم إرساله للطرف الثاني يقوم بدوره بحلق رأس العبد مرة أخرى حتى يستطيع قرأة الرسالة، وهكذا كانت بدايات استخدام هذة الطريقة لإخفاء رسالة أو معلومة ما تحت غطاء أو شيء ما حتى لا يكون هناك علم أن أي إتصال سري يتم مابين إثنين أو أكثر.
وهذه قد تعد ميزة لإخفاء البيانات، فهي لا تثير الشك بوجود بيانات مخفية، على عكس التشفير، فناتج التشفير يكون مختلفا تماما وغير مقروء أو مفهوم، مما قد يثير الشكوك حوله، توجد أدوات جاهزة تقوم بعملية اخفاء البيانات، منها مثل أداة Steghide، لكن كمبرمجين، سنتعرف على طريقة عمل مثل هذه الأدوات.

هناك عدد من الطرق والخوارزميات لإخفاء البيانات، حيث بإمكانك اخفاء نوع من البيانات (كالنصوص والملفات) داخل نوع آخر من البيانات، فمثلا بإمكانك اخفاء صورة بداخل صورة أخرى، أو صورة بداخل ملف فيديو، أو نص بداخل ملف صوتي، وهكذا، دون أن يلاحظ من يشاهد الصورة بتغيير فيها، فالخوارزميات المختصة بإخفاء البيانات تتلاعب بمحتويات الملف، حيث تعدل الـ bits الخاصة بالملف وتتلاعب به بحيث لا تؤثر على محتويات الملف، وبنفس الوقت تحقن بداخله الـ bits الخاصة بالملف الآخر، فعندما ترى صورة تحتوي بداخلها على نص مثلا، فإنك لن ترى النص، فالنص يُمثل داخل الصورة على شكلbits ، ولن تلاحظ كذلك تغيرا على جودة الصورة (بحسب الطريقة المستخدمة طبعا)

الفرق بين إخفاء البيانات (Steganography) والتشفير(Encryption)

ربما سمعت سابقًا بمصطلح التشفير Encryption) أو (Cryptography وهو – باختصار – تشفير المعلومة لتصبح غير مفهومة وغير قابلة للقرآة إلا من قبل الشخص الذي يمتلك مفتاح التشفير لفك الشفرة وكل من يحصل على هذا المفتاح .التشفير يكون دائمًا لغرض حماية وأمن المعلومات وأسباب تشفير المعلومات كثيرة، منها: تبادل بيانات سرية بين شركات معينة، بين دوائر حكومية معينة، وغيرها .في المقابل، عند الرغبة في إخفاء المعلومات (Steganography) فإننا نقوم بتضمين المعلومات داخل وسيط ما وتحت غطاء معين بحيث لا تظهر هذه الرسالة لمن يستعرض الوسيط الأساسي (سواءً كان صورة أو فيديو) لأنها مخفية تمامًا داخله، وهذا أشبه ما يكون بالتواصل من وراء الكواليس.

بالتالي، نستطيع القول أن **الفرق الأساسي بين التشفير وإخفاء المعلومات** هو أنه عند تشفير Encryption) أو Cryptography)معلومة ما، يستطيع الطرف الثالث معرفة أن هناك إتصال يتم مابين طرفين (شخصين أو جهتين) لكنه لا يستطيع فهم المعلومات لأنها مشفرة. أما في حالة إخفاء المعلومات (Steganography) ، لا يعلم الطرف الثالث بأن هناك شيء مخفي في الخفاء أو أن هناك إتصال بين إثنين يتم من وراء الكواليس لأنه تم استخدام وسيط ما لإخفاء هذا الإتصال تمامًا.

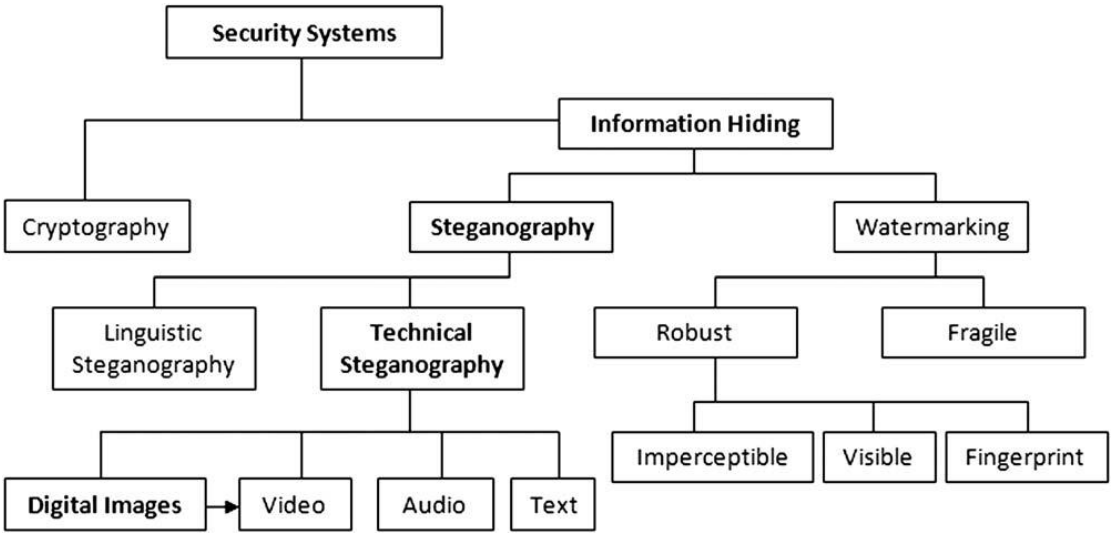
الوسائط المستخدمة في إخفاء البيانات(Steganography)

أبرز الوسائط التي تستخدم في إخفاء المعلومات أربعة وهي:

- الملفات النصية.
- الصور.
- الملفات الصوتية.
- مقاطع الفيديو.

أنواع وطرق حجب البيانات

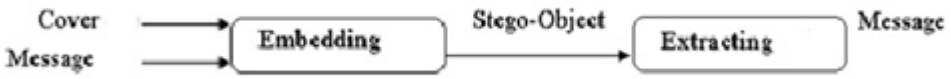
الشكل التالي يوضح عدة أنواع وطرق لحجب البيانات والتي من بينها التشفير (Encryption أو Cryptography) و إخفاء البيانات(Steganography) ، وكما هو موضّح، هنالك أقسام فرعية تحت كل نوع



أنماط إخفاء البيانات(Types of Steganography)

- Pure Steganography

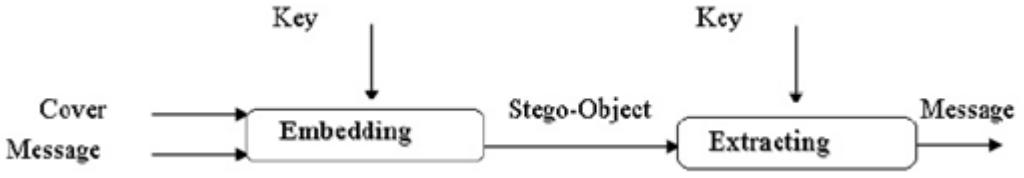
وهو النوع أو النمط العادي والخام من الأنماط المستخدمة لإخفاء المعلومات، هنا يتم تضمين المعلومات أو الرسالة الخفية داخل الوسيط بشكل مباشر وبدون كلمة سرية (شكل ١).



شكل ١: الآلية المتبعة في النوع العادي أو الخام من إخفاء المعلومات

- Secret Key Steganography

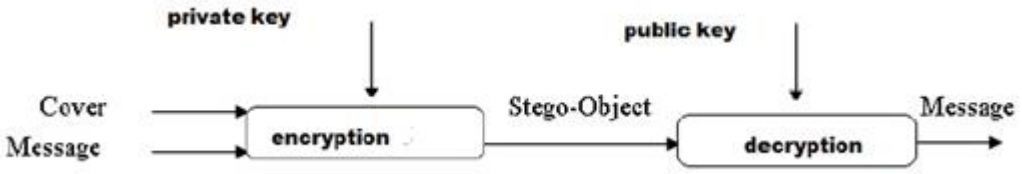
يعني إخفاء المعلومات بإستخدام مفتاح أو كلمة سرية تضاف للرسالة المخفية عند إخفائها داخل الوسيط المستهدف. وهكذا لا يمكن إسترجاع أو قراءة الرسالة المخفية من قبل الطرف الثاني إلا بمعرفة الكلمة السرية، وبإضافة الكلمة السرية لعملية الإخفاء تكون العملية آمنة ومعقدة أكثر (شكل ٢).



شكل ٢ :الآلية المتبعة في إخفاء المعلومات مع إستخدام كلمة سرية

- Public Key Steganography

ويعني إخفاء المعلومات بإستخدام مفتاح عام، والعملية هنا تشبة العملية المتبعة في التشفير عن طريق إستخدام مفتاحين، الأول مفتاح "عام" ويستخدمه الشخص الأول عند عملية إخفاء المعلومة، ويتم إستخدام المفتاح الثاني "الخاص" من قبل الشخص المستقبل عند إسترجاعه للمعلومة المخفية، مع العلم أن المفتاح الخاص له علاقة مباشرة مع المفتاح العام (شكل ٣).



شكل ٣ :الآلية المتبعة في إخفاء المعلومات مع إستخدام كلمة سرية

ما هي التطبيقات اخفاء المعلومات؟

وهي تستخدم أساسا إخفاء المعلومات لحجب معلومات سرية / البيانات أثناء التخزين أو النقل. على سبيل المثال، يمكن للمرء أن يخفي رسالة سرية في ملف صوتي وإرسال هذا إلى طرف آخر عبر البريد الإلكتروني بدلا من إرسال الرسالة في تنسيق نصية.سيكون المتلقي على الطرف الآخر يفك رموز شفرة الرسالة المخفية باستخدام مفتاح التشفير الخاص. في سيناريو أسوأ الحالات، حتى لو كان هناك طرف ثالث تمكن من الوصول إلى البريد الإلكتروني، كل ما يمكن أن يجد هو الملف الصوتي وليس البيانات المخفية في داخله.أغراض أخرى من إخفاء المعلومات تتضمن البحث الرقمية من الصور لأسباب مثل حماية حقوق التأليف والنشر.

على الرغم من إخفاء المعلومات لها تطبيقات كثيرة مفيدة، قد تكون بعض استخدام هذه التقنية لأغراض غير مشروعة مثل اخفاء محتوى إباحي في غيرها من الملفات الكبيرة. كما تم الإبلاغ عن الشائعات حول استخدام الإرهابيين لإخفاء المعلومات والاتصال لإخفاء معلوماتهم السرية والتعليمات. وذكر مقال يدعي أن، شركة كيو دا استخدمت إخفاء المعلومات لترميز الرسائل في الصور ونقلهم عن طريق رسائل البريد الإلكتروني، من قبل صحيفة نيويورك تايمز، في أكتوبر ٢٠٠١.

كيف اخفاء المعلومات أدوات العمل؟

أدوات Stegnography تطبيق خوارزميات ذكية لترسيخ بعناية رسائل نصية أو بيانات مشفرة داخل غيرها من الملفات الكبيرة مثل الصور، الفيديو، الصوت أو ملف قابل للتنفيذ. وبعض الأدوات تضمين البيانات المشفرة في نهاية ملف آخر بحيث تكون هناك مساحة كافية لتخزين بيانات أكبر.

هناك العديد من أدوات إخفاء المعلومات المتاحة على الانترنت ولكن فقط عدد قليل قادرون على عمل لا تشوبه شائبة. لم أجد أي أداة تعمل تماما على البيانات الصغيرة والكبيرة على حد سواء. ومع ذلك فقد تمكنت من تطوير أداة التي يمكن أن تعمل تماما على جميع أنواع الملفات وجميع حجم البيانات. وتسمى أداة **"Stego السحرية".** يمكنك تحميل البرنامج من الرابط التالي.

تحميل StegoMagic

تحميل StegoMagic.zip

ملف مضغوط يحتوي على نسختين من StegoMagic: واحد لتشفير الرسائل النصية والآخر لتشفير الملفات الثنائية. ويمكن استخدام StegoMagic_TXT لاخفاء رسائل نصية في ملفات أخرى مثل صورة أو ملف صوت. ويمكن استخدام StegoMagic_BIN لإخفاء ملف واحد ثنائي في آخر مثل ملف قابل للتنفيذ داخل صورة أو صورة داخل ملف فيديو.

مع StegoMagic، ليس هناك أية قيود على حجم ونوع الملف الذي كنت تنوي إخفاء. على سبيل المثال، يمكنك إخفاء شريط فيديو من ١ غيغابايت في حجم صورة من حجم ١ ميغابايت أو إخفاء ملف قابل للتنفيذ داخل مستند Word. الأداة هي جميلة واضحة لاستخدام ولا يتطلب أي فهم خاص لهذا المفهوم.
في نهاية عملية التشفير، سيتم إنشاء مفتاح فك التشفير السري، وهو مطلوب خلال عملية فك التشفير.

كيفية استخدام StegoMagic؟

افترض أنك تريد اخفاء رسالة نصية داخل ملف JPG:
١. وضع JPG وملف نصي (.txt) في المجلد نفسه الذي من *StegoMagic_TXT.exe*
٢. تشغيل *StegoMagic_TXT.exe* واتبع تعليمات الشاشة لتضمين الرسالة النص داخل صورة JPG.
٣.اضغط باستمرار على *مفتاح فك التشفير السري.*
الآن يمكنك إرسال هذه الصورة لصديقك عبر البريد الإلكتروني. إلى فك تشفير الرسالة الخفية، وينبغي أن صديقك تحميل هذا الملف JPG على الأداة*السحرية Stego* واستخدام *مفتاح فك التشفير السري.*

تطبيق خوارزمية LSB باستخدام PHP لإخفاء نص بداخل صورة.

مفاهيم نظرية مهمة قبل البدء

ان كل البيانات المخزنة داخل الحاسوب هي في الأساس تمثل بنظام العد الثنائي binary، وقد سمي نظام العد الثنائي بهذا الاسم، لاحتوائه فقط على رقمين) رمزين) لتمثيل البيانات، وهما ٠ و ١، ولعرض بيانات يفهمها المستخدم) كالنصوص والصور والأرقام) يتم تحويل نظام Binary إلى أنظمة عد أخرى لتسهيل تمثيل البيانات، وسنتطرق في هذه المقدمة بشكل غير مفصل لطريقة تحويل نظام العد Binary إلى نظام العد) Decimal الارقام التي نستخدمها في الحياة اليومية)، من ثم سنقوم بتوضيح طريقة تمثيل الصور في الحاسوب باستخدام نظام العد Binary وذلك لأننا سنستخدم الصور كمثال لاخفاء البيانات بداخلها.

نظام العد العشري Decimal هو نظام العد الذي يمثل الأرقام العادية التي نستخدمها في الحياة اليومية، وسمي بالعشري لامكانية تمثيله بعشرة أرقام، وهي الأرقام من ٠ إلى ٩.

لنفترض أن لدينا بايت واحد من البيانات (والذي يساوي (8 bit، لو تخيلنا شكله، فسيكون كالآتي:

لتمثيل أي رقم Decimal بنظامBinary ، نقوم أولا بترقيم كل bit من اليمين إلى اليسار بـ(2^x) حيث x هو رقم الـ bit ابتداء من ٠ ووصولا إلى) n حيث n هو عدد ال bits الاجمالي ناقص واحد، وهو في هذه الحالة ٧)، فمثلا، البت الاول من اليمين يمثل بالرقم (١ = ٢^٠) والثاني (٤ = ٢^٢) وهكذا، فيصبح الترقيم كالآتي:

والآن يتم تمثيل أي رقم بجعل قيم الـ bits التي مجموع ترقيمها يساوي الرقم الذي نريده، فالرقم ١٦ مثلا يُمثل بهذه الطريقة:

وهذا مثال على الرقم ١٣٧:(128+8+1)

ومن الطريقة السابقة، نلاحظ أن أقصى قيمة عددية يمكن أن يمثلها البايت الواحد هي ٢٥٥ (وهي ناتج جمع كل الأرقام السابقة ١٢٨+٦٤+٣٢+١٦+٨+٤+٢+١)، ولتمثيل أعداد أكبر تحتاج إلى المزيد من البايتات.

يفترض أن أغلب المبرمجين يعرفون المعلومات السابقة، لكن كيف يفيدنا هذا في موضوعنا ؟

كما ذكرنا في المقدمة، فإن خوارزميات إخفاء البيانات، تعتمد بشكل أساسي على التلاعب بالـbits ، لذلك لابد من فهم هذه المقدمة قبل شرح خوارزميات اخفاء البيانات، وبما أن مثالنا سيكون عن إخفاء نص بداخل صورة، فإن علينا أن نعرف كيف يتم تمثيل الصورة باستخدام نظام العد الثنائي.

مم تتكون الصور؟

تتكون الصور من مجموعة كبيرة من النقاط يسمى الواحد منهاPixel ، وكل بكسل يحتوي لون واحد فقط، بالتالي فإننا عندما نتكلم عن طريقة "تمثيل الصور" نتكلم فعليا عن طريقة تمثيل كل بكسل داخل الصورة، لذلك بامكانك تشبيه الصورة بـ Matrix(مصفوفة ثنائية البعد) ويحتوي كل عنصر في هذه المصفوفة على بكسل واحد فقط، تختلف طريقة تمثيل البكسل بحسب عمق الألوان ووجود شفافية في الصورة من عدمه، لكن بشكل عام، نستطيع أن نقول أن اللون يتكون من تداخل قيم ثلاثة ألوان، وهي الأحمر

والأخضر والأزرق(RGB) ، وقيمة كل لون من هذه الألوان تُمثل باستخدام رقم من ٠ إلى ٢٥٥ وهو الرقم الذي يحدد كثافة اللون، ومما سبق معرفته عن طريقة تمثيل أرقام Decimal باستخدامBinary ، نعلم أن الرقم ٢٥٥ هو (١١١١١١١١) في نظام Binary، أي أننا سنكون بحاجة لبايت واحد فقط لتمثيل اللون، وعلى افتراض ان البكسل فيه ٣ قنوات فقط دون قناة الشفافية الموجودة في بعض صيغ الصور كـ PNG مثلا، فإننا سنمثل البكسل باستخدام 3 Bytesحيث يُمثل كل لون في بايت واحد.

هذا مثال على أحد الألوان وتمثيله باستخدام:BInary

ما هي خوارزمية LSB وكيف تعمل

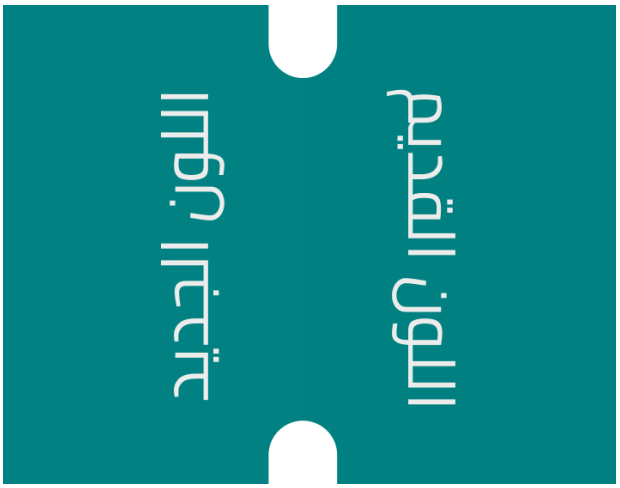
Least Significant **B**it (**LSB**)تعني بالعربية الـ Bit الأقل تأثيرا، لكن ما هو الـ Bit الأقل تأثيرا ؟

ربما لاحظت في الأمثلة السابقة أنك لو قرأت الـ Bits من اليمين إلى **اليسار**، فالقيمة العددية التي يمثلها الـ Bit تزداد، فالـ Bit الأول من اليمين يمثل لنا الرقم ١، بينما يمثل الأخير الرقم ١٢٨، لذلك فإننا كلما تحركنا إلى **اليمين** تقل القيمة العددية التي يمثلها الـ Bit) ويقل تأثيره على البيانات)، فمثلا لو كان لدينا صورة وعكسنا قيمة الـ Bit الأول من اليمين لأحد البكسلات (أو حتى لجميعها) ففعليا لن تلاحظ فرقا كبيرا، فلو كانت كثافة اللون الأحمر في ذلك البكسل ١٢٧، فستصبح القيمة ١٢٦ بعد تغيير الـ Bit الأول من ١ إلى ٠، والعين البشرية لن تلاحظ التغيير، فهو تغيير طفيف جدا، لذلك يسمى الـ Bit الأقل تأثيرا، فكلما تحركنا إلى اليمين قلت

أهمية الـBit ، في الصورة السابقة شاهدنا لونا ومثلناه باستخدام Binary، الصورة التالية ستعرض نفس اللون بعد تغيير أول Bit من كل لون، إضافة إلى عرض اللون القديم لتقارن بينهما:

من الواضح جدا عدم وجود فرق وعدم إمكانية التمييز بين اللونين، ماذا لو قمنا بتقريب الصورة أكثر ؟

هذه هي الصورة بعد التقريب بنسبة ١٢٨٠٠% تقريبا (الخطان الأبيضان يمثلان الحد بين اللون القديم والجديد):



تلاحظ أنه من غير الممكن أبدا التمييز بين اللونين بالعين المجردة ولا يمكن ملاحظة أي فرق، لكن كيف يفيدنا هذا في اخفاء البيانات؟

بما أن تغيير قيمة الـ Bit الأقل تأثيرا لا تُعد مشكلة بالنسبة للملف (سواء كانت صورة أو مقطع صوت أو فيديو أو غيرها) فإن بإمكاننا استخدام هذه الـ Bitsفي إخفاء بيانات أخرى، على سبيل المثال، هذا الـ Binary الخاص بهذا النص: "code"

```
01100011 01101111 01100100 01100101
```

لنفترض أن لدينا صورة نريد إخفاء هذا النص بداخلها، سنقوم بقراءة كل Pixel في الصورة، ونستخرج الـ Binary الخاص بالألوان فيه، ثم نُغير قيمة الـ Bit الأقل تأثيرا من كل لون إلى قيمة أحد الـ Bits الخاصة بالنص أعلاه، فالـ Pixelالأول من اليسار مثلا سيحتوي أول 3 Bits من النص السابق حيث يحتوي كل لون على Bit واحد من الـ Bits الخاصة بالنص، ستوضح لك هذه الصورة آلية عمل الخوارزمية نوعا ما، وستتضح أكثر عندما نكتب الكود الخاص بها:

المثال السابق يوضح الـ Binary الخاص بأول ٧ بكسلات من صورة، وكما تلاحظ فإن كل لون يمثل بايت واحد كما وضحنا مسبقا، في الأسفل ترى الـ Binary الخاص بكلمة "code" وقد قمت بتلوين أول ٥ bits لإيضاح طريقة الإخفاء في الجدول على اليمين، بالطبع لن تكفي ٧ بكسلات فقط لإخفاء هذه الكلمة، لكن المثال للتوضيح فقط.

مثال على استخدام خوارزميةLSB

سنكتب ونشرح الكود الخاص بالخوارزمية باستخدام لغةphp ، في هذا الشرح سنقوم باستخدام إضافة ImageMagickـ PHP للتعامل مع الصور، قم بتنصيبها قبل بدء التطبيق على الدرس.

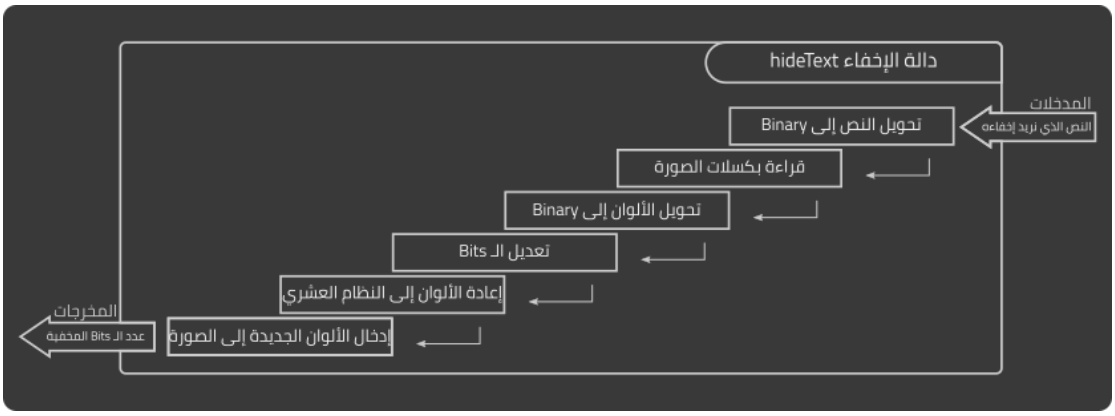
سنقوم بكتابة كلاس (Class) باسمStegHide ، وسيحتوي على جميع الدوال التي ستلزمنا في عملية إظهار/إخفاء البيانات داخل الصورة:

```
class StegHide {

    protected $image;

    public function __construct(string $imagePath) {
        $this->image = new Imagick($imagePath);
    }

    public function setImage(string $path) {
        $this->image = new Imagick($path);
    }

    public function saveImage(string $name) {
        $this->image->writeImage($name . '.png');
```

```
    }
}
```

في الكلاس السابق لدينا متغير باسم image وسيحتوي على الصورة التي سنتعامل معها، عند انشاء Object من هذا الكلاس ستحتاج إلى تمرير مسار الصورة التي تريد إخفاء أو إظهار البيانات الخاصة بها، والدالة setImage ستستخدم لإعادة تعيين الصورة في حال اردنا استخدام نفس الـ Object مع صورة أخرى، وأخيرا الدالة saveImage ستقوم بتخزين الصورة كملف بصيغة.png

سنعمل أولا على آلية إخفاء البيانات، ثم نقوم بكتابة دالة إظهارها، لكن أولا يجب أن نعرف ما هي الخطوات التفصيلية لإخفاء البيانات، هذه صورة توضيحية (قد تختلف في بعض تفاصيلها بحسب طريقة برمجتك للخاصية) :



كما ترى فإن هذه هي خطوات اخفاء الصورة، بعضها سيُكتب داخل دالة hideText مباشرة، والبعض الآخر سنضعه في دالة أخرى لتسهيل قراءة الكود وتنظيمه، وتسهيل اعادة استخدامه.
لاحظ الدالة ترجع قيمة عدد الـ Bits التي تم إخفاؤها، فالنص "abc" مثلا حجمه 3 Bytes أي ما يساوي 24 Bits، وسنكون بحاجة لهذا الرقم لاستخدامه في عملية إظهار البيانات، فيجب على دالة اظهار البيانات أن تتوقف عن الدوران على البكسلات بعد قراءة كافة الـ Bits الخاصة بالنص، الأفضل طبعا إخفاء هذا الرقم داخل الصورة أيضا ضمن Byte محدد مثلا، حتى لا تضطر إلى تزويد الدالة بأي شيء عند الإظهار، لكن لتقليل الخطوات ضمن الشرح سنزود الدالة بعدد الـ Bits يدويا.

الخطوة الأولى هي تحويل النص إلى نظام العد الثنائي(Binary) ، وأُفضل أن تكون هذه دالة مستقلة كالآتي:

```
protected function stringToBin(string $string): string {
    $binary = '';

    for ($i = 0; $i < strlen($string); $i++)
        $binary .= sprintf('%08b', ord($string[$i]));

    return $binary;
}
```

في البداية قمنا بتعريف متغير باسم binary، والذي سيحتوي على قيمة النص بعد التحويل، ثم قمنا بالدوران على جميع أحرف النص، وباستخدام الدالة ordيتم تحويل الحرف إلى رقم بنظام العد العشري Decimal ، وهذا سيساعدنا في تحويله من نظام العد العشري إلى نظام العد الثنائي، فالدالة sprintfتقوم بإعادة ترتيب أو تنسيق النص الذي يُدخل إليها بحسب الصيغة التي تمررها لها وهي في هذه الحالة (08b%) وبهذا نكون قد طلبنا من الدالة باستخدام الحرف b أن تعامل المدخلات كرقم وتمثلها كـ) Binary أي أنها تقوم

بتحويل الـ Decimal إلى (Binary أما (٠٨) فوظيفتها إضافة ٠ إلى يسار النص الناتج في حال كان طول النص أقل من ٨، إلى أن يكتمل طول النص ويصبح ٨، وفائدتها تحديد البيانات على أنها بحجمByte ، فمن دون هذه الإزاحة سيمثل الرقم (٤) مثلا بهذه الطريقة (١٠٠) بينما نحن نريده هكذا.(00000100)

وبعد تحويل جميع الأحرف إلى نظام العد Binary تقوم الدالة بارجاع المتغير.binary

الآن سنبدأ بالخطوة الثانية وهي قراءة بكسلات الصورة، كما في المثال:

```
public function hideText(string $text): int {
    $binaryText = $this->stringToBin($text);

    $iterator = $this->image->getPixelIterator();

    foreach ($iterator as $row => $pixels) {
        foreach ($pixels as $column => $pixel) {

 # Pixel يجب أن نقوم بتحويل الألوان إلى النظام الثنائي هنا بعد قراءتها من
 #                المتغير
 # وهنا يجب أن نقوم  Bits الخاصة بالألوان وإدخال الـ Bits الخاصة بالنص
 #       بتعديل الـ
 # كما يلزم نقوم بإعادة تحويل الألوان إلى النظام العشري لإعادة إدخالها إلى
 #    بعد تعديل الـ Bits الصورة

 # بعد الانتهاء من الخطوات السابقة يجب  Bits فلا توجد حاجة لمتابعة الدوران
 #    إيجاد آلية لإيقاف الدوران عند ادخال جميع الـ

        }
        $iterator->syncIterator();
    }

    return strlen($binaryText);
}
```

لاحظ التعليقات المكتوبة بداخل الدالة، وهي توضح الأجزاء الناقصة من هذه الدالة والتي سنكتبها واحدة واحدة، لكن قبل ذلك لنشرح ما كتبناه حتى الآن حتى يكون كل شيء واضحا قبل الانتقال للخطوة التالية.

في البداية قمنا باستدعاء الدالة التي كتبناها مسبقا لتحويل النص إلىBinary ، وقمنا بتخزين ما ترجعه في المتغيرbinaryText ، بعد ذلك قمنا بجلب الـ Iterator الذي يساعد بالدوران على بكسلات الصورة (ميزة جاهزة في كلاس Imagick اضغط هنا لتفاصيل أكثر) وقمنا بكتابة جمل الدوران، حيث أن كل صف من البكسلات يكون مخزنا في المتغيرpixels ، بينما الـ index الخاص بهذا الصف سيخزن فيrow ، من ثم سيبدور الدوران الثاني على كل بكسل في ذلك الصف، حيث يخزن الـ index الخاص بالعمود في column، بينما يخزن الـ Object الخاص بالتعامل مع البكسل في pixel، وباستخدام هذا الـ Object نستطيع القيام بالعديد من العمليات، وما يهمنا منها هنا هو جلب اللون أو تغييره.

بعد الانتهاء من الدوران على البكسلات داخل كل صف، يتم تنفيذ الدالة syncIteratorوهي المسؤولة عن حفظ التعديلات التي نقوم بإجرائها على البكسلات.

الآن سنكتب دالة أخرى تستقبل Object من نوع (ImagickPixelسنرسل لها المتغير pixel الموجود في الدالة السابقة(، ثم تقوم بإرجاع مصفوفة (Array) فيها نفس ألوان البكسل الذي قمنا بإرساله لكن بنظام:Binary

```
protected function colorToBin(ImagickPixel $pixel): array {
    $colors = $pixel->getColor();

    return [
        sprintf('%08b', $colors['r']),
        sprintf('%08b', $colors['g']),
        sprintf('%08b', $colors['b'])
    ];
}
```

كما ترى فالدالة بسيطة جدا، نقوم بجلب الألوان من البكسل باستخدام getColor التي سترجع Associative Arrayتحتوي على الكثافة اللونية للألوان الثلاثة (RGB) والـ key الخاص بكل لون هو أول حرف من كل لون (واضح بالمثال)، ثم نُرجع هذه القيمة على شكل array بعد تحويلها إلى binary باستخدام.sprintf

انظر الآن كيف أصبحت دالة الاخفاء بعد استدعاء الدالة السابقة والبدء بتعديل الـ:Bits

```
public function hideText(string $text): int {
    $binaryText = $this->stringToBin($text);
    $pointer = 0;
    $shouldStop = false;

    $iterator = $this->image->getPixelIterator();

    foreach ($iterator as $row => $pixels) {
        foreach ($pixels as $column => $pixel) {

            ######################################################
            $colors = $this->colorToBin($pixel);                 #
                                                                 #
            for ($i = 0; $i < 3; $i++) {                         #
                $colors[$i][7] = $binaryText[$pointer];          #
                                                                 #
            $shouldStop = (++$pointer == strlen($binaryText));   #
            if ($shouldStop)                                     #
                break;                                           #
            }                                                    #
            ######################################################

            # كما يلزم نقوم بإعادة تحويل الألوان إلى النظام العشري لإعادة إدخالها
بعد تعديل الـ Bits إلى الصورة

            ####################
            if ($shouldStop)   #
                break;         #
            ####################


        }
        $iterator->syncIterator();

        ####################
```

```
        if ($shouldStop)    #
             break;         #
        ####################

    }

    return strlen($binaryText);
}
```

لاحظ الأجزاء الجديدة (داخل المستطيلات)، في البداية قمنا باستدعاء الدالة التي كتبناها لتحويل الألوان من نظام العد العشري إلى نظام العد الثنائي، وقمنا بتخزينها في المتغير.colors

كذلك أضفنا for جديدة تقوم بالمرور على الألوان (الأحمر والاخضر والازرق) وتعدل الـ index السابع من كل Byte (لأن السابع هو الأقل تأثيرا)، وبدلا من قيمته نضيف Bit واحد من الـ binary الخاص بالنص، ونقوم بتحديد المكان الذي وصلنا إليه باستخدام المتغيرpointer ، الذي تبدأ قيمته من صفر، وتزداد كلما اضفنا bit آخر.

وتلاحظ أننا أضفنا شروط داخل كلfor ، وهذه الشروط تتأكد من قيمة المتغيرshoudStop ، الذي يحتوي ناتج المقارنة بين القيمة التالية لـpointer ، وطول الـBinary ، ففي حال أصبح الـ pointer مساويا للطول، فهذا يعني أنه تم ادخال جميع الـ Bits إلى الصورة.

بقي الآن فقط أن نكتب الدالة التي تحول مصفوفة الألوان من Binary إلى Decimal واضافة الألوان إلى الصورة:

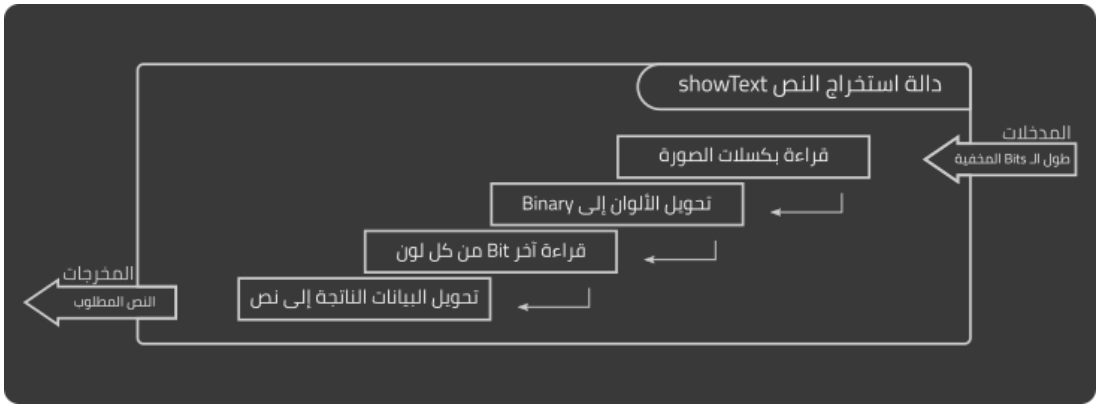
```
protected function binToColor(array $colors): string {
    $r = bindec($colors[0]);
    $g = bindec($colors[1]);
    $b = bindec($colors[2]);
    return "rgb($r, $g, $b)";
}
```

كما ترى، فالمصفوفة تستدعي فقط دالة bindecلتحويل كل لون، وتضع الألوان بداخل نص بصيغة مدعومة من كلاس ImagickPixel، الآن ننتقل إلى الاستدعاء وتكون دالة الإخفاء جاهزة:

```
public function hideText(string $text): int {
    $binaryText = $this->stringToBin($text);
    $pointer = 0;
    $shouldStop = false;

    $iterator = $this->image->getPixelIterator();
    foreach ($iterator as $row => $pixels) {
        foreach ($pixels as $column => $pixel) {
            $colors = $this->colorToBin($pixel);
            for ($i = 0; $i < 3; $i++) {
                $colors[$i][7] = $binaryText[$pointer];
                $shouldStop = (++$pointer == strlen($binaryText));
                if ($shouldStop)
                    break;
            }
```

```
            ################################################
                                                           #
            $pixel->setColor($this->binToColor($colors));  #
                                                           #
            ################################################

            if ($shouldStop)
                break;
        }
        $iterator->syncIterator();
        if ($shouldStop)
            break;
    }

    return strlen($binaryText);
}
```

هكذا نكون قد انتهينا من كتابة دالة إخفاء النص بداخل الصورة. وإليك نبذة عن خطوات العمل لاستخراج النص:



أغلب الخطوات شرحناها مسبقا في شرحنا لإخفاء النص، لذلك سنكتب الدالة كاملة ونشرحها بشكل مختصر:

```
public function showText(int $length): string {
    $iterator = $this->image->getPixelIterator();
    $binaryText = '';

    foreach ($iterator as $row => $pixels) {
        foreach ($pixels as $column => $pixel) {
            $colors = $this->colorToBin($pixel);
            foreach ($colors as $color) {
                $binaryText .= $color[7];
                if (strlen($binaryText) == $length)
                    return $this->binToString($binaryText);
            }
        }
    }
    return false;
}
```

في البداية قمنا بتعريف الـ iterator ومتغير سيحتوي على الـ Binary التي سنحولها إلى نص. ثم قمنا بجلب لون البكسل داخل الدوران باستخدام دالة colorToBin التي عرفناها مسبقا.
ثم قمنا بالدوران على جميع الألوان وإضافة آخر Bit من كل لون إلى المتغير، ثم في حال تحقق الشرط بحيث أصبح طول الـ Bits المستخرج يساوي الطول الذي تم تزويد الدالة بها يتم إرجاع النص بعد تحويله باستخدام دالة binToString (لم نعرفها بعد) وهذه الدالة هي المسؤولة عن تحويل سلسلة الـ Binary التي حصلنا عليها إلى نص، وهو النص الذي تم إخفاؤه.

وهذا تعريف هذه الـMethod:

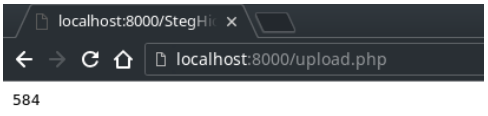
```
protected function binToString(string $binary): string {
    $string = '';
    $start = 0;
    while ($start <= strlen($binary)) {
        $string .= chr(bindec(substr($binary, $start, 8)));
        $start += 8;
    }
    return $string;
}
```

ستقوم هذه الميثود بالدوران على كل Byte باستخدام substr، والمتغير start يزداد بمقدار ٨ (لكي يقرأ النص على شكلBytes ، وكل بايت يتم تحويله إلى رقم بنظام العد العشري باستخدامbindec ، من ثم تحول هذا الرقم إلى حرف باستخدام chr، وتضيف كل حرف إلى المتغير string وتقوم بإرجاعه.

لنقم الآن بتجربة الكلاس، سنقوم بإخفاء عبارة "تجربة كتابة خوارزمية LSB على عالم البرمجة:"

```
$stegObj = new StegHide('img.png');
# تعريف كائن جديد من الكلاس، وتحديد مسار الصورة

echo $stegObj->hideText('على عالم البرمجة LSB تجربة كتابة خوارزمية');
# استخدام دالة الإخفاء وطباعة القيمة التي سترجعها لاستخدامها لاظهار النص

$stegObj->saveImage('3alampro');
# حفظ الصورة الجديدة التي تحتوي على البيانات المخفية
```

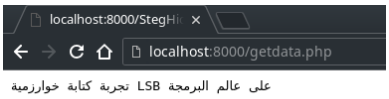
والنتيجة:



إضافة إلى أنه قد تم إنشاء صورة جديدة باسم ٣alampro.png، الآن لنجرب استرجاع البيانات المخفية:

```
$stegObj = new StegHide('3alampro.png');
echo $stegObj->showText(584);
```

والنتيجة:



الآن بامكانك ارسال الصورة الجديدة التي تحتوي على البيانات المخفية لأي شخص دون أن يلاحظ أحد أي تغيير على الصورة، هذه الصورة المستخدمة في المثال السابق مع الصورة التي تم إخفاء البيانات بداخلها:

الصورة الأصلية

الصورة الجديدة التي تحوي بيانات مخفية

يمكنك التجربة بنفسك وتحميل الصورة واظهار النص الموجود،وهذا الكلاس كامل على:Github

https://github.com/3mmarg97/StegHide

ملاحظة :هذا الكلاس لا يُعد مشروعا كاملا، ولا يُنصح باستخدامه في بيئة إنتاجية أو في مشروع حقيقي، إنما هو كلاس بسيط كُتِب لغايات تعليمية فقط