Hemliga meddelanden

Att gömma text i bilder utan att det märks

Jonathan Nissov Skarman, Victor Lindgren von Corswant

NTI GYMNASIET JOHANNEBERG

Länk till github repository

Bakgrund

Att gömma data eller hemliga meddelanden är inget nytt och har gjorts på olika sätt genom tiderna. Att specifikt gömma data i en digital bild däremot är något som blev populärt när datorernas minnen expanderades. Det finns olika sätt att gömma data i digitala bilder och vissa bildformat gör det enklare eller svårare. Enligt artikeln "*What is Steganography?"* How to hide data inside data [1]" (Freecodecamp.com, av Daniel Iwugo) finns det olika sätt att både gömma och kryptera data. Artikeln tar upp fem sätt, text, bild, video, ljud och nätverk. Då alla alternativ är bra sätt att gömma data på, står bild ut över de andra då den är mer säker och relativt enkel att genomföra. Det finns två sätt att gömma data i bilder, med nyckel och utan nyckel. Med nyckel kommer datan skrivas om efter en angiven nyckel när den göms i bilden, detta gör att läsning av datan kommer att vara krypterad om man inte har nyckeln. Utan nyckel är som det låter, man har ingen nyckel som krypterar och datan är då väldigt enkel att läsa av om man vet vart att leta.

Hur gömmer man data i bilder? Den vanligaste bildformaten använder sig av Hexadecimal där varje pixel består av en Byte som i sig består av åtta Bits. Hexadecimal använder sig av RGB färg mellan 0 till 256 och skrivs av antingen två bokstäver, två siffror eller en blandning av båda. Det krävs då tre hexadecimalt för att få en komplett färg där röd, grön och blå använder varsin hexadecimal. Om du vill ha färgen vitt, alltså 256 på alla färger, blir det FF i hexadecimal och svart blir 00. Genom att ändra på en eller alla hexadecimaler av varje pixels färg kan man gömma en Bit av data i varje pixel. Detta kommer såklart att ändra färgen på pixeln, men förhoppningsvis är det tillräckligt liten ändring för att det mänskliga ögat inte ska kunna se skillnaden. Har man tillräckligt många pixlar att redigera ska man kunna gömma upp till teoretiskt sett hur mycket data som helst.

Bild 1: Demonstration av förändring av pixlars färgvärden

Syfte

Detta projekt ämnar till att undersöka hur data kan gömmas i bilder, och hur mycket data som kan gömmas utan att bildens utseende ändras tydligt.

Frågeställningar

- Vilken mängd data går att gömma i en bild innan bilden märkbart skiljer sig från originalbilden.
  - Spelar bildstorlek roll i mängd data som kan lagras
- Hur signifikant kan varje pixel ändras utan att det märks

Metod

För att undersöka hur effektivt text kan gömmas i bilder, valde vi att genomföra en serie experiment där olika steganografiska tekniker tillämpas. Syftet med vår metod är att dölja data på ett sätt som inte påverkar bildens visuella kvalitet, samtidigt som informationen kan extraheras utan att den förstörs. Processen består av flera steg, från förberedande undersökning av befintliga tekniker och utveckling av ett anpassat program, till insamling och analys av data för att mäta teknikernas effektivitet. Nedan beskrivs i detalj de metoder vi använt oss av.

Förberedelse

Förundersökningen gjordes via googling av nyckelord, där dom som gav bäst resultat var "steganography", och "how to hide text in image". Via detta googlande hittades två huvudmetoder, där den första var att sätta ihop en GIF fil och en ZIP fil, vilket då GIF filer läses framifrån och ZIP filer läses bakifrån ger en fil med datan från båda. Den andra metoden som hittades var att ändra den sista 'bit'en i varje byte, vilket skulle ge en extremt liten förändring i färg som nästan alltid kommer vara omärkbar. I förundersökningen hittades även information om hur olika bildformat fungerar, och då både PNG och JPG har former av komprimering, som gör att förändringar till enskilda bytes kan helt ändra hur bilden ser ut, valdes BMP formatet för bilderna som datan kommer gömmas i.

Utveckling

Programmet kommer utvecklas i Ruby, och kommer fokusera på att kunna ändra den minst märkbara datan för att kunna gömma olika textsträngar i BMP filer.

Datainsamling

Datan kommer samlas in via att en originalbild och en lista med bilder, varav vissa original, och vissa förändrade, kommer ges till test subjekten, för att undersöka hur ofta de förändrade bilderna ses skillnad på från bilderna som är identiska till originalbilden.

Data analys

Dataanalysen kommer att baseras på skillnaden i frekvensen av identifierade skillnader från originalet jämförelsevis mellan de förändrade bilderna, och kontroll bilderna. Mängden identifierade skillnader från originalet kommer plottas som en funktion av skillnad mot originalbilden, och då en signifikant ökning i identifieringar sker kommer gränsen sättas.

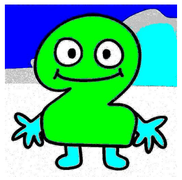


Bild 2 : Aggression 1

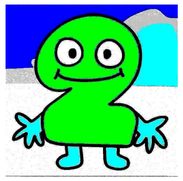
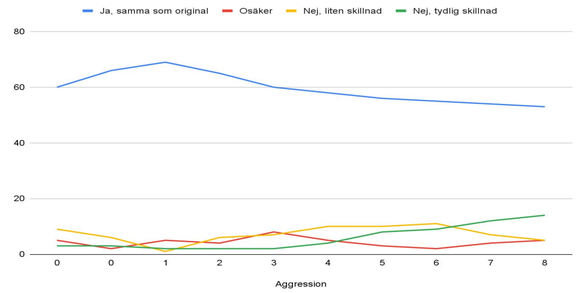


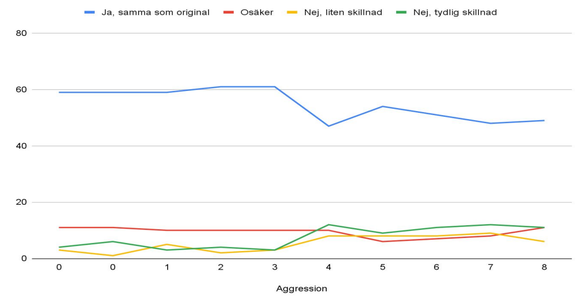
Bild 3: Aggression 8

Resultat

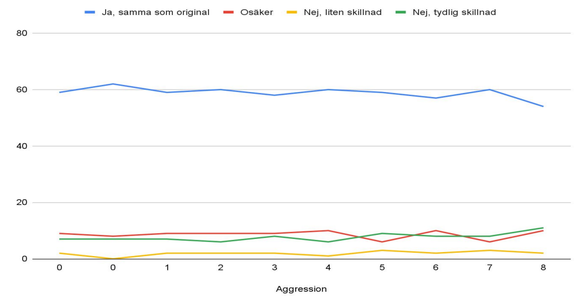
Nedan redovisas svarsresultatet på undersökningsformuläret. *Graf 1* visar tydlig korrelation mellan aggression och möjligheten att identifiera förändringar i bilden. *Graf 2* visar liknande resultat till Graf 1 men har en större andel identifieringar vid aggression 4. *Graf 3* visar inget tydligt samband mellan aggression och identifieringar, vilket visar på att bilder med mer data kan gömma data bättre.



Graf 1: 500 x 500 pixlar



Graf 2: 500 x 500 pixlar



Graf 3: 3840 x 2160 pixlar

Diskussion

I Graf 1 visas en tydlig korrelation mellan aggression och andel korrekta identifieringar. Graf 2 hade mer noise i grundbilden än Graf 1, vilket gjorde att det var generellt svårare att märka av skillnaderna i dom bilderna, men då aggression 4 orsakade lite grundläggande förändringar i bilden var det en större andel som märkte av skillnaden. Graf 3 var för en mycket större bild än Graf 1 och Graf 2, så även om mängden data som skulle gömmas ökades för Graf 3 visade det sig att större bilder gör det extremt mycket svårare att märka skillnaderna som uppstår.

Slutsats

Mängden data som går att gömma i en bild är betydligt stor utan att bildens synliga yta ändras så att det märks. Vid en bild på 500x500 pixlar går det att trycka in runt 100,000 tecken i bilden på aggression 1. Detta betyder att det skulle gå att trycka in enorma mängder data i bilderna. Mängden tecken som går att trycka in multipliceras med aggression nivån, vilket betyder att aggression 8 klarar runt 800,000 tecken, vilket bör vara mer än vad som behövs.

Hur mycket varje pixel ändras har liten påverkan på ändringens synlighet och större aggressioner som upp mot fem till åtta krävs för att se skillnaderna i pixlarnas färg värden.

Utvecklingen av arbetet har visat att små förändringar av pixlarnas värden på pixlar inte har stora verkningar på synligheten, däremot har stora ändringar på pixlars värden större förändringar i synlighet.

Ytterligare har det demonstrerats flera gånger att optimering av kod har stora inflyttningar på programmets hastighet och beteende under programmets gång. Flera timmar av skrivning och läsning har sparats in på optimering. Samtidigt är tiden beroende på bildens storlek, vilket ökar exponentiellt.

Källor

Wikipedia.com "*BMP file format*" (Hämtad 4 September) https://en.wikipedia.org/wiki/BMP_file_format#Pixel_array_(bitmap_data)

Youtube.com *"How Bitmap file format works"* (Hämtad 4 September) https://www.youtube.com/watch?v=kpHFFFu9qeU

Ruby Documentation "*Packed data"* (Hämtad 11 September) https://docs.ruby-lang.org/en/master/packed_data_rdoc.html#label-Hex+String+Directives