Adatrejtés és vízjelezés

Eltemetett bitek

Tóth Tamás

2006

A *Kriptográfia* című Ph.D kurzushoz a dolgozatot Tóth Tamás II. informatika Ph.D hallgató készítette.

Konzulens: Prof. Dr. Pethő Attila

Az alcím Virasztó Tamás Titkosítás és adatrejtés című könyvének 11. fejezete alapján.

Tartalomjegyzék

Bevezetés

Ebben a dolgozatban a titkostási módszerek *szteganográfia*, azaz adatrejtés című fejezetés mutatom be. A szeganográfia görög szó, szó szerint azt jelenti, hogy *„rejtett írás”*. Az adatrejtést egyszerűen definiálhatjuk úgy, hogy abban különbözik a többi titkosítási módszertől, hogy igazából nem a titkosításon van a hangsúly, hanem magának a titkosításnak az elrejtéséről. Ha belegondolunk, sokkal kisebb az esélye, hogy megfejtik a titkos adatainkat akkor, ha nem is tudnak a létezésükről.

A még zajlanak viták, hogy a szteganográfia titkosító módszernek számít-e. Mivel más az elvük, más a céljuk, jól elkülönülnek az egyéb kritogáfiai módszerektől, ezért közvetlenül nem sorolhatók a titkosító módszerek közé. Viszont Mégis együtt kell velük tárgyalni, mert kiegészíti, sőt fel is válthatja a titkosítást az adatrejtés. Nem az a cél, hogy az üzenet hozzáférhetetlen legyen illetéktelen számára, hanem, hogy magát az információcserét „fedje” el. Bár elég gyanússá teszi a kommunikációt, ha a felek csak képeket küldözgetnek egymásnak.

Ha titkosított üzenetet rejtünk el, az kétszeres biztosítást jelent, mert először az illetéktelen személynek észre kell vennie magának az adatrejtésnek a tényét, majd az előásott titkosított adatot meg is kell fejtenie. Ezekről a módszerekről a dolgozat első részében lesz szó.

A szteganográfia egyik felhasználási lehetősége, ha az adatok – illetve szerzői jogok – védelmében szeretnénk használni adatrejtő eszközöket. A valós, kézzelfogható világban is találunk példát ezekre, pl. italokon a zárjegy garantálja a termék eredetiségét. A dolgozat második részében az adatrejtés alkalmazási lehetőségeiről és tulajdonságairól fogunk szólni pár mondatban.

1. fejezet

Szteganográfia

1.1. Történelmi áttekintés

Az adatrejtésre már az ókori Görögországban találunk példát. A görögök viasszal bevont táblát használtak írásra. Egy történet szerint Demeratus figyelmeztetni akarta Spártát, hogy Xerxés inváziót tervez, lekaparta a tábláról a viaszréteget, ráírta az üzenet a csupasz fára, majd újra bevonta viasszal. Egyetlen ellenőrzés során sem derült ki, hogy a tábla rejtett információt tartalmaz. Egy másik módszer, hogy egy megbízható szolga haját leborotválták, és a fejbőrre tetoválták az elrejtendő információt. Majd megvárták míg a haj kinő, hogy a tetoválás észrevehetetlen maradjon. Az információhoz újbóli tetoválás után fértek hozzá.

A későbbiekben, a II. világháborúig a láthatatlan tinták uralták a területet. Különböző anyagokat használtak fel információ elrejtésére. Ezek jellemzője, hogy a „tinták" száradás után láthatatlanok lesznek, majd hő vagy valamilyen kémiai anyag hatására válnak illetve láthatóvá. A II. világháború idején a láthatatlan tintával való írás szinte kizárólag a jobbnál-jobb kémiai anyagok kifejlesztésére korlátozódott.

A világháborúban a sok titkos üzenet közt egy-egy nem titkosított üzenet is átkelt a kommunikációs csatornán, de korántsem volt biztos, hogy nincs-e benne rejtett információ. Az alábbi szöveg egy időjárásjelentés a világháborúból:

`News Eight Weather: Tonight increasing snow. Unexpected precipitation smothers eastern towns. Be extremely cautious and use snowtires especially heading east. The highways are knowingly slippery. Highway evacuation is`

`suspected. Police report emergency situations in downtown ending near Tuesday.`

Jelentése a következő: „Nyolc órás hírek, időjárásjelentés: Ma növekszik a havazás. Hirtelen havazásra lehet számítani a keleti városrészben. Óvatosság és hólánc használata ajánlott. Az autópályák csúszósak, lezárásuk várható. A rendőrség nagy fennakadásokról számol be a belvárosban, és keddig nem várható javulás."

Ha minden szó kezdőbetűjét összeolvassuk, a következőt kapjuk:

`Newt is upset because he thinks he is President.`

„Newt megőrült, mert azt hiszi, hogy ő az elnök."

A következő szöveget egy német kém küldte a II. világháborúban:

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

Jelentése: „A semlegesek tiltakozását nyilvánvaló módon figyelmen kívül hagyták. Isman érzékenyen érintve. A blokád következményei érintik az embargós termékekért emelt kifogásokat is a faggyú és a növényi olaj visszautasítását." Ha minden szó második betűjét összeolvassuk, a jelentés jelentősen megváltozik:

`Pershing sails from NY June 1.`

Azaz, „Pershing június elsején kihajózik New Yorkból".

Virasztó Tamás, a [6] szerzője nyilatkozik arról, hogy nem pontos az egyezés az elrejtett adat és annak értelmezése közt, vagyis előfordulhat eltérés pár betűben, illetve ahogy a fenti példa is mutatja az ‚l' betű értelmezhető 1 (számjegy)-ként is. Az ilyen eltéréseket a titok megfejtője képes kijavítani.

A második világháború idején az adatrejtés és a rejtett adatok utáni keresés (*Office of Censorship*) odáig „fajult", hogy még keresztrejtvények postai úton történő küldése is tilos volt. Ha a cenzor végképp nem tudott „belekötni" a levélbe vagy képeslapba, összecserélgette a bélyegeket, a szavak sorrendjét, vagy egyes szavakat már rokonértelmű szavakkal helyettesített. Vagy éppenséggel átfogalmazta az egész levelet. A vietnámi háború idején a frontról hazaérkező leveleken az USA cenzorai képesek voltak a bélyeget kicserélni, ha az nem a szabályos helyén volt [3].

1.2. A szteganográfia jelene

A világháború idején a szteganográfia meglevő módszereket használt fel. Az igazi újdonság az volt, amikor új módszerrel ötvözték a szteganográfia alapjait: változó hosszúságú vonalak, színek és egyéb képi elemek hordozhattak titkos adatokat.

A digitális technika fejlődése új utat nyitott a szteganográfia számára, és ma is aktív kutatási területnek számít, főleg a szerzői jogvédelem és a rejtett csatornákat illetően.

2. fejezet

Alapelvek

Ebben a fejezetben bemutatjuk a szteganográfiával kapcsolatos célokat és elvárásokat, a szakkifejezések jelentését, stb.

2.1. Terminológia

A szteganográfia célja, hogy nagy tömegű „lényegtelen" adat közt rejtük el a titkos információt. Ezt a nagy tömegű adatot hívjuk **hordozónak** (*cover, cover-text*), míg a titkos információt **üzenetnek** (*plain-text, embedded-data*). A hordozó és az üzenet együttesét (adatrejtés után) *stego-text*-nek nevezzük.

Az, hogy az üzenet titkosított-e, nem lényeges az eljárás szempontjából, tehát az üzenet szempontjából nincsenek megkötések. A szteganográfia módszerei az adatrejtést a hordozó egyes bitjeinek megváltoztatásán alapszanak. Emiatt csak az lehet hordozó, aminek értelmezésében nem jelent problémát egyes bitjeinek megváltoztatása. A megváltoztatott tartalom egyfajta zajként jelenik meg a hordozón, minél több bitet változtatunk meg, annál nagyobb a zaj. Célszerű tehát olyan hordozót választani, ami *emberi* információt tartalmaz[1].

[1] Például a kép vagy hang, ahol egy-egy bit megváltozása nem befolyásolja a tartalom megértését, sőt, a legtöbb esetben észrevétlen marad.

2.2. A szteganográfia céljai

Két fő irányt kell megkülönböztetnünk a robosztusság és a rejtettség viszonyában. Az első esetben egyszerűen „csak" információt akarunk elrejteni a hordozóban, ahogy erre már példákat is láttunk az előző fejezetben. Ekkor a rejtettség áll minden szempont előtt – azaz másodlagos szempont a hordozóba rejthető adat mennyisége. A második esetben a titkos üzenet robosztussága az elsődleges szempont, akár a rejtettség rovására is. Ebben az esetben az üzenetnek lehetőleg eltávolíthatatlanul kell a hordozóba kerülnie, sőt ki bizonyos esetekben ki kell állnia a hordozóra általánosságban alkalmazható transzformációkat is. Ez utóbbi eset fedi le a *vízjelezés* alapgondolatát. A vízjel több célt is szolgálhat, egyrészt biztosíthatja a hordozó eredetiségét (azaz a hordozó nem változása esetén a vízjel ezt egyértelműen jelzi – *törékeny vízjel*), másrészt a hordozó eredetét (azaz a nyomon lehet követni a hordozó digitális útját – robosztus vízjel).

Az értékelhető tulajdonságok:

1. Elrejthető adatmennyiség (*payload*)
2. Észrevehetetlenség
3. Az alkalmazott technika megbízhatósága (*reliability*)
4. Biztonság (*security*)
5. Az elrejtett adat túlélési lehetőségei (*survivability*)
6. Az elrejtett adat hatása a hordozóra

2.3. Támadásfajták

A szteganográfiát is kétféle támadás fenyegetheti. **Passzív támadásnak** nevezzük azt, amikor a támadó felfedi az adatrejtés tényét – talán el is tudja olvasni –, de nem tudja vagy nem akarja megváltoztatni azt. Ezzel szemben **aktív támadás** az, amikor a támadó elsődleges célja, hogy a rejtett adatot – tartalmának ismerete nélkül –, megváltoztassa, vagy eltávolítsa.

Emellett a hordozót számos olyan behatás érheti, ami egyfajta támadásnak minősülhet, a szándékosság feltételezése nélkül. A következő felsorolásban ezeket a behatásokat soroljuk fel:

- Veszteséges tömörítés. Kép vagy hang alapú hordozó esetén fordulhat elő, mégpedig az „emberi információ” redundáns mivoltát kihasználva. Ez a módszer épp a legkisebb helyiértékű biteket módosítja, amibe a legtöbb adatrejtő algoritmus az üzenetet rejti. Kép esetén leginkább a JPG, míg hang esetén az MP3 formátum a legjellemzőbb.
- Szűrések. Szintén kép és hang alapú hordozóra jellemző. Például alul- és felüláteresztő szűrő eyidejű alkalmazása.
- Szerkesztés. Részek kivágása, áthelyezése„ átméretezés, hangok esetén erősítés, stb.
- Újabb adatrejtési eljárásban a sztegotext hordozóként szerepel.
- Zaj. Főleg analóg átviteli csatorna esetén.
- A/D és D/A konverzió. Azaz az analóg és a digitális világ közti határ átlépésekor fellépő nehézségek.
- Kép alapú hordozó nyomtatása és beszkennelése.

3. fejezet

Az adatrejtés alkalmazásai

Ebben a fejezetben a két legegyszerűbb esetet vizsgáljuk meg, vagyis a legegyszerűbb kép és hangformátumot használjuk hordozóként. Az ilyen adatokat digitalizálással készítik, vagyis az adott jelenséget mérik, és a mért adatokat digitális értékként – valahány biten – tárolják. És mivel kettes számrendszerről van szó, lehetőség van a legkisebb helyiértékű bitek megváltoztatására. Ez a hordozón egyfajta mérési hibaként jelenik meg, és mivel ez az emberi észlelést alapvetően nem befolyásolja, tökéletesen alkalmas adatrejtésre.

Túl sok bitet azonban nem szabad megváltoztatnunk, mert ezzel azt kockáztatjuk, hogy a túlságosan zajos kép mögött valaki felismeri, hogy nem egy egyszerű képről van szó, bár ezáltal több adatot rejthetünk el a hordozóban.

3.1. LSB módszer

3.1.1. Képek

A legegyszerűbb képformátum 24 bites színmélységű, tömörítés nélküli és RGB színrendszert használ. A 24 bites színmélység azt jelenti, hogy egy képpont (*pixel*) tulajdonságainak leírására 24 bit áll rendelkezésre. Az, hogy nem tömörített pedig azt jelenti, hogy a képpontok valamilyen sorrendben helyezkednek el a képfájlban, általában sorfolytonosan. Így egy-egy képpont hogy csak egy tulajdonságot kell rendelnünk, a színét. Az

RGB-rendszerben egy szín három komponensből áll elő (*Red, Green, Blue*), a komponensek változtatásával az összes szín kikeverhető. Mindhárom komponens ugyanannyi bájton tárolódik, tehát 24 bites kép esetén minden komponensre 8-8 bit, azaz egy bájt jut.

Ilyen formátumok pl. a BMP és a TGA. A kettő közt a legejjelmzőbb különbség, hogy a kép meta-adatait (függőleges és vízszintes felbontás, színmélység, nyomtatási felbontás *dpi*, stb) tartalmazó fejrész mérete eltérő. Alkalmas hordozó lehet a PNG formátum, ami ugyan tömörített, de ún. „veszteségmentes" tömörítést használ, azaz a formátum nem okoz adatvesztést, bár ezt a képméret rovására teszi. Nem alkalmas hordozónak a JPG formátum, mert olyan „veszteséges" tömörítést használ, ami a homogénnek mondható területeket összemossa, éppen a legkisebb helyiértékű biteket manipulálva. alkalmassá tehető viszont, ha saját JPG kódolót, és dekódolót írunk.

Már többször említettük a legkisebb helyiértékű bitek megváltoztatását, ezt idegen szóval LSB (*Least Significant Bit*) módszernek nevezzük. A módszer – ahogy arról már szó volt – a legkisebb helyiértékű bitek módosításán alapszik. A fejezet elején részletezett formátum ugyanis lehető teszi ezeknek a biteknek a módosítását anélkül, hogy az észrevehető lenne az ember számára.

Példaként vegyünk egy 1024×768 méretű RGB képet. A képpontok eltárolásához $1024 \times 768\times = 2359296$ bájtra van szükség. Ha minden képpont mindhárom színkompensének felhasználjuk az legalsó helyiértékű bitjét, akkor ugyanennyi, 2359296 bitet – vagyis pontosan 288 Kb adatot – tudunk belesűríteni a képbe. Természetesen lehetőség van színkomponensenként egynél több bit felhasználására is, akár a 3-4 legalsó bit módosításával is próbálkozhatunk, sőt figyelhetünk arra a tényre is, hogy az ember a zöld színre a legérzékenyebb, a vörösre a legkevésbé. Nem szabad megfeledkeznünk arról, hogy minél több bitet változtatunk meg, annál nagyobb a rejtett adat felismerésének a veszélye.

A módszer hátránya, hogy a tömörítetlen képek mérete hatalmas: 1024×768 felbontás esetén $2,25$ Mb, holott egy ilyen méretű kép JPG tömörítéssel pár tíz kilobájt. Érdemes tehát a sztego-képet valamilyen nem veszteséges formátumba betömöríteni, pl.: JPEG200, PNG, TIFF.

JPG formátum

Képek esetén két dolgot kell még megemlítenünk. Az egyik, hogy lehet JPG formátumba is adatokat rejteni. Ekkor tudnunk kell azt, hogy a tömörítés előtt egy diszkrét koszinusz transzformáció (DCT) hajtódik végre, azaz a frekvenciatartományba konvertálják

a képtartomány adatait, ahol az együtthatók nem egyformán fontosak: a magas frekvenciájú tagok módosíthatók anélkül, hogy látható (vagy MP3 esetén hallható) változást okoznánk. Tömörítés során ezt ki is használják, a legmagasabb frekvenciájú tagokat kinullázák. Minél több egy adatblokkban a nulla érték, annál redundásabb, annál nagyobb mértékben tömöríthető. Ha ezeket a biteket adatrejtésre akarjuk használni, olyan saját JPG kódolót kell írnunk, ami a DCT és a tömörítés között hozzáfér az adatokhoz. A tömörítés hatékonysága jelentősen romlik, de az adataink megmaradnak.

Szürkeskálás képek

A másik megjegyeznivaló a szükreskálás képek esete. Az emberi szem sokkal érzékenyebb a szürkeskálás képekre, mint a színes képekre – nyilvánvalóan könnyebb megkülönböztetni 256 színt, mint nagyságrendileg 16 milliót. Ezért vigyáznunk kell az ilyen képek hordozóként történő felhasználásánál.

Az előbbi esetben a szürkeárnyalatos képet tipikusan úgy képzeltünk el, hogy egyszer 8 bit áll rendelkezésre, hogy a fekete (0) és a fehér (255) szín közti átmenetet leírja. Viszont nem csak 8 bites szürke képek léteznek, a CT-képek például 12 bites szürke képekkel dolgoznak.

Másik buktató lehet az indextábla. Egyes főleg 256 színű képek esetén a képponthoz tartozó érték nem egy világosságkód a fekete–fehér skálán, hanem egy sorszám, ami egy úgynevezett indextáblában lévő színre utal. Egy egy képben bármelyik 256 szín szerepelhet, de csak 256, ilyen pl. a GIF formátum. Ilyen képbe nem tudunk adatot rejteni, mert a legkisebb helyiértékű bit megváltoztatása lehet, hogy teljesen más színt eredményez.

3.1.2. Hangok

Természetesen az LSB módszer segítségével az adatainkat nem csak képekbe, hanem hanganyagokba is belerejthetjük. Míg a képek esetén a legegyszerűbb mindenki számára hozzáférhető formátum a BMP, addig ugyanez a hangoknál a WAV formátum.

A WAV az egyik legrégebbi hangok tárolására alkalmas formátum. A legegyszerűbb a tömörítés nélküli PCM (*Pulse Code Modulation*) kódolású formátum. A formátum a tömörítés hiánya miatt archiválásra kevésbé alkalmas, viszont jól használható olyan alkalmazásokban, ahol a kódolásra csak kis erőforrás áll rendelkezésre.

Hangokban történő adatrejtés esetén szintén a legkisebb helyiértékű bitek módosíthatók, akár az alsó nyolc bit is megváltoztatható, jelentős minőségromlás nélkül. Ha sok bitet használunk fel, célszerű nagy dinamikájú zeneszámot választani hordozóként, amin kevésbé hallatszik az adatrejtésből származó zaj. Az algoitmusról bővebben a B függelékben.

3.2. Egyéb technikák

Természetesen a fent bemutatott LSB módszer a legegyszerűbb módja adatok elrejtésének, és egyszerűségéből kifolyólag a legkevésbé biztonságos módja is. A következőkben a teljesség igénye nélkük szeretnénk felvázolni egyéb lehetséges megoldásokat, amik az LSB módszertől biztonságosabban képesek az adatrejtésre.

3.2.1. Szórt adatrejtés

Az alap LSB módszerben az üzenet bitjeit folyamatosan ráültetjük a hordozó bitjeire. Vagyis ha az üzenet kisebb, mint a hordozó, a hordozó egy összefüggő része fog módosulni, a másik összefüggő része érintetlen marad. Kiolvasásnál egyszerűen megnézzük az adat elejét, és addig próbáljuk a biteket összerakni, amíg az eredményre nem vezet, vagy fel nem adjuk, és azt mondjuk, hogy nincs titkos üzenet a sztego-adatban.

Jelentősen megnehezíti a helyzetet, ha az üzenetet nem egy egységként rejtünk a hordozóba, hanem valamilyen „véletlen” módszerrel szétszórjuk a hordozóban [3]. Ekkor a szétszórás kulcsként szerepel a technikában, tehát a kulcs ismerete nélkül nem, vagy nagyon nehezen fejthető meg a rejtett üzenet.

A szórás meghatározható valamilyen pszeudovéletlen módszerrel, aminek kezdeti értéke a módszer kulcsa.

A módszer nagyobb védelmet nyújt, mint az LSB, bár a tömörítésre továbbra is érzékeny, sőt mivel tömörítetlen adatot küldünk (bitmap képet, vagy WAV formátumű hangot) gyanús lehet.

3.2.2. Szórt spektrumú adatrejtés

Az SSIS (*Spread Spectrum Image Steganography*), azaz a szórt spektrumú adatrejtés egy modulációs technika, amit a hírközléstechnikában (szórt médiában) használnak.

Képernyőn való kép-megjelenítéskor a képernyőt elektronsugarak pásztázzák, meghatározott sorrend szerint végigjárják a képpontokat. Egy időskálán a sugár intenzitása egy mintasor. Mint ilyen létezik spektruma, és értelmezhetők a digitális jelekkel kapcsolatos fogalmak is[1]. Az üzenet is ugyanígy értelmezhető, illetve a frekvenciaként értelmezett üzenettel modulálható a frekvenciaként jelenlevő hordozó[2].

A modulációs technikától eltérően a hordozó egy adott frekvencia ablakában – a zajtartományban – történik az adatrejtés, így nehezebb észlelni. Ez a modern adatrejtésnek követelménye is.

Fontos megjegyezni, hogy egy ilyen technika alkalmazása nem függ a hordozó típusától, attól, hogy az tömörítve van-e vagy sem, mivel a tömörített képeknek is van spektrumuk. Tehát az SSIS módszerrel nyugodtan rejthetünk el adatokat pl. JPG fájlokban.

3.2.3. Maszkolás, különbségi adatrejtés

A legtöbb képszerkesztő program képes a képet rétegekként kezelni, illetve a különböző rétekeget valamilyen módon összeolvasztani. Az egyik réteg tartalmazza a hordozót, egy másik réteg a valamilyen elven elrendezett üzenetet, Lehetőség van a két réteg összeadására, aminek eredményeként paraméterezéstől függően látható vagy láthatatlan vízjelet kapunk – ha látszik az üzenet a hordozón, látható vízjelről van szó. Bővebben ld. 3.7 fejezet.

3.3. Információrejtés mozgóképbe

Képekbe történő adatrejtés után felmerülhet a kérdés, hogy tudnánk adatokat rejteni mozgóképbe. Általában a mozgókép minden másodperce 25 képkockából áll. Megtehetjük, hogy bizonyos képkockákat cseréljünk ki teljes egészében a saját üzenetünkre. Az ilyen képkockákat úgysem lehet észrevenni[3] (emberi szemlélő által, ha nem számít rá), viszont egyes kódolók épp úgy próbálnak tömöríteni, hogy egyes képkockákat elhagynak. Másrészt a videó kockánkénti ellenőrzésével könnyen felfedezhetők és eltávolíthatók a rejtett

[1] Egyszerűbben fogalmazva ugyanezt kapjuk, ha egy képet a frekvenciatartományba transzformálunk pl. FFT vagy DCT transzformációval.

[2] Ezt már az 1940-es években is alkalmazták az úgynevezett Strogoff kerék segítségével.

[3] Tudat alatt viszont észlelhetők, sőt felhasználták reklám-célokra is, bár ezt később betiltották.

információk. Alkalmazhatók viszont olyan eljárások, amik a tömörített képekbe rejtenek információkat (pl. SSIS).

3.4. Kivételek

Eddig azt neveztük jó hordozónak, amik sok redundáns adatot tartalmazanak, mint a képek és a hangok, amikben kis mértékű változás nem befolyásolja (jelentősen) az emberi megértést. Ebben a szakaszban bemutatunk pár technikát, amiknek segítségével a hordozónak teljesen alkalmatlan szöveges fájlokba rejthetünk el adatokat.

3.4.1. Sorkiegészítés

A szöveges állományba történő adatrejtés legegyszerűbb módját a sorok végén jelentkező „láthatatlan" redundáns szóközök biztosítják. A módszer alapötlete, hogy a sorok végén lévő szóközök nem látszanak, tehát annyi szóköz karaktet írhatunk a sorok végére, ahányat csak akarunk.

Hordozónak olyan szöveget kell választanunk, ami hosszú, és a sorai tördelve vannak. Maga az algoritmus a következő lépésekből áll:

1. Olvassuk be a következő sort a fájlból.
2. Ha a sor végén felesleges szóközök vannak, azokat töröljük.
3. Vegyük az üzenetünk következő n bitjét, és képezzünk belőle decimális számot.
4. Az n bitből képzett számnak megfelelő számú szóköz karaktert szúrjunk be a sor végére.
5. Írjuk vissza a feldolgozott sort.
6. Ismételjük a lépéseket, amíg el nem fogynak az üzenet bitjei.

Minél több bitnek megfelelő számú szóközt írunk a sor végére, annál nagyobb a szöveg kapacitása, és egyuttal a felismerés veszélye is. Fontos megjegyezni, hogy a sztego-text nyomtatása teljesen mértékben törli a titkos üzenetet, ugyanis egyis szövegfelismerő program sem fogja tudni, hogy a sorok végén felesleges szóköz karakterek szerepelnek.

3.4.2. Leírónyelv-manipuláció

A HTML és RTF formátumú allományok szöveges fájlok, ahol a szöveg formázásáról ún. *tag*-ekben lévő formázó utasítások gondoskodnak. Általában mindegy, hogy ezeket az utasításokat, vagyis a leírónyelvet kis- vagy nagybetűkkel írjuk, esetleg vegyesen. Tehát felhasználhatjuk a tageket az üzenetünk hordozására. Ahol az üzenet bitjeiben 1-es van, ott nagybetűt használunk, ahol 0, ott kisbetűt, pl.: a `<HtMl>` szöveg a 1010_2 bitsorozatnak felel meg.

Bizonyos tagek esetén viszont fontos, hogy kis- vagy nagybetűvel van-e írva. Ha Unix rendszer estén a fájlok hivatkozásainál összekeverjük a kis- és nagybetűket, hibát okozunk.

3.5. Zajterhelés

Példaként tekintsük a képfájlunkat 24 bitesnek (3×8 bit) és a hangfájlunkat 16 bitesnek. Ha minden mintából 2 bitet használunk fel az üzenet rejtésére, a képünket a legrosszabb esetben -36 dB, míg a hangot -84 dB zaj terheli.

$$Q_Z = 20 \lg \frac{4}{256} < -36 \text{ dB}, \qquad Q_Z = 20 \lg \frac{4}{65536} < -84 \text{ dB},$$

ami jóval kisebb, mint az észrevehető.

Képek esetén célszerű szkennelt képeket használni hordozóként, mivel itt a belerejtett adat – ha észreveszik – úgy néz ki, mintha az a szkennelés hibája volna, vagyis a felhasználó kevésbé fog gyanakodni adatrejtésre, mint a szkennelés hibájára.

Általánosságban felírható a zajszámításra vonatkozó képlet. A hordozó egy mintája K biten reprezentálható, ebből S bitet használunk fel, ezzel

$$Q_Z[\text{dB}] = 20(S - K) \lg 2$$

zaj terheli a sztego-textet.

Érdekes elgondolás lehet, hogy adatokat rejtsünk a zenei CD-kbe. A 16 bites analóg CD-k elméletileg olyan elő- és végerősítőket követelnek meg, amiknek a jel/zaj viszonya jobb, mint 96 dB. Ezt a követelményt az erősítők általában nem követelik meg, tehát a 16 bit tekinthető túlmintavételezésnek. Ha a minták valódi 16 bitesek, azaz csak zenei jeleket tartalmaznak, nem zajt, akkor a visszajátszás minősége a visszajátszó eszköztől függ.

Ha felhasználjuk az alsó 4 bitet adatrejtésre, akkor

$$Q_Z = 20(4 - 16) \lg 2 \approx -72,2 \text{ dB}$$

zaj terheli a hordozót. Tekintve, hogy egy átlagos végfok nem garantál 60-80 dB-nél nagyobb jel/zaj arányt, ez a -72 dB zaj el fog veszni az erősítő termikus zajában. Tehát nyugodtan felhasználhatjuk CD-ink negyedét adatrejtésre.

A probléma az, hogy a CD olvasásakor nem mindig azt kapjuk, amit a CD-re írtunk. Egy-egy bithiba a zenehallgatás élményét nem befolyásolja, viszont a rejtett adatokat tönkreteheti. Célszerű ezért valamilyen hibatűrő vagy hibajavító kódolást alkalmazni.

3.6. Saját implementációk

Természetesen egy ilyen izgalmas téma esetében születnek saját implementációk. Az képbe történő adatrejtést valósítottam meg, két megközelítésből. A programok Java nyelven íródtak, mivel az lehetőséget biztosít több képformátum kezelésére is (*javax.imageio, java.awt.image*).

3.6.1. Kép rejtése képbe

Először is vitassunk meg pár alapvető dolgot.

1. Milyen típusú képet akarunk használni? Természetesen a legegyszerűbb, tömörítés nélküli 24 bites RGB formátuma van szükségünk. A Java-ban az ilyen kép pontjait egy-egy 32 bites `int` típusú érték reprezentálja. A 32 bitet fel tudjuk osztani 4×8 részre, vagyis a képpont A–R–G–B formátumban van, ahol a legfelső 8 bitre, az *alfa-csatornára* (átlátszóság) nincs szükségünk.

2. Hány bitet szeretnénk felhasználni? A legcélravezetőbb az lenne, ha az üzenet egy bájtját egy képpontban el tudnánk rejteni. Vagyis a probléma az, hogy 8 bitet kell 3 részre osztanunk. Segít a döntésben az a tény, hogy az ember a zöld színre a legérzékenyebb, tehát választhatunk úgy, hogy a piros és a kék csatorna alsó három bitjét, míg a zöld csatorna alsó két bitjét módosítjuk. Legrosszabb esetben képcsatornánként 8 érték eltérést hozva létre.

A módszer feltétele, hogy a hordozó és az üzenet felbontása ugyanakkora legyen, továbbá, hogy az üzenet-kép szürkeárnyalatos legyen. Ugyanis csak így garantálható, hogy az üzenet beleférjen a hordozóba. Természetesen megoldató, hogy egy 24 bites üzenet-képet rejtsünk a hordozóba, de ahhoz három hordozóra van szükség.

A módszer maga a következő lépésekből áll:

1. Különböző ellenőrzésekkel (létezik-e, meegyezik-e a méretük) megnyitjuk a képeket.
2. Két `int` tömbbe betöltjük a hordozót és az üzenetet, majd a biztonság kedvéért az üzenetet szürkeárnyalatossá alakítjuk.
3. Bejárjuk a hordozó összes képpontját (az aktuálisat i-vel jelöljük), és mindegyikre végrehajtjuk a következő lépéseket:
   (a) A hordozó i-edik (`cover[i]`) képpontját felbontjuk `r`, `g`, `b` értékekre.
   (b) Az üzenet i-edik (`plain[i]`) képpontjának az alsó bájtát (mindhárom csatorna ugyanazt az értéket tárolja) felbontjuk, három részre, hogy azok rendre 3–2–3 bitet tartalmazzanak.
   (c) Egyesre állítjuk a megfelelő számú legkisebb helyiértékű bitet (R és B csatorna esetén 3-at, G csatorna esetén kettőt).
   (d) Létrehozunk egy maszkot, ami az alsó bitjeiben tárolja az üzenet megfelelő bitjeit, a többi bitje csupa egyes.
   (e) A megfelelő maszkot a megfelelő csatornához logikai ÉS operátorral összekapcsoljuk.
   (f) A maszkolás után a három csatorna értékét egyesítjük, ez lesz a sztego-kép i-edik bitje.
4. Kimentjük a sztego-képet PNG formátumban.
5. DEBUG: Különbségképet készítünk a sztego-képről és a hordozóról.

3.6.2. Adat rejtése képbe

Az előző problémához hasonló ha adatokat akarunk a képbe elrejteni. Minden ugyanúgy történik, az egyetlen különbség, hogy az üzenet nem egy kép egyik csatornája, hanem egy fájl bájt-tömbként beolvasott tartalma.

Ebben az esetben viszont egyáltalán nem biztos, hogy az üzenet éppen olyan hosszú, mint a hordozó, tehát valamilyen meg kell határozni, hogy milyen hosszú is a hordozóba rejtett üzenet. Két lehetőségünk van:

- Az üzenetet olyan formátumúvá alakítjuk, ami biztosan nem tartalmaz nulla-bájtot[4], majd az utolsó bájt után írt nulla-bájtból tudjuk, hogy pontosan mennyi az üzenet hossza.
- Másik lehetőség, hogy megjegyezzük, hogy mekkora az üzenet mérete, és kiolvasásnál annyi bájtra van szükségünk. Ez úgy nézhet ki, mint egy kulcs a redszerhez, de ettől még a módszer nem ad több biztonságot. A „kulcs" nem-ismerete nem állít megoldhatatlan feladat elé, csak kissé kényelmetlenné teszi a rejtett információ kiszedését.

3.6.3. Adatok kinyerése

Természetesen nem elég az üzenetet elrejteni a hordozóban, tudni kell szideni is. A 3.6.1 szakaszban leírtak szerint egyszerű a dolgunk, ha az üzenet a hordozó felbontásával azonos felbontású kép. Minden képpontot fel kell bontanunk a színcsatornáknak megfelelően, és a piros, zöld és kék csatornák alsó 3–2–3 bitjét össze kell illesztenünk egy nyolc bájtos egésszé. Ez a módszer alapja. Mivel ez egy színcsatornát jelent, utófeldolgozáskánt meg kell háromszroznunk, hogy valódi szürkeárnyalatos RGB színrendszerű képet kapjunk.

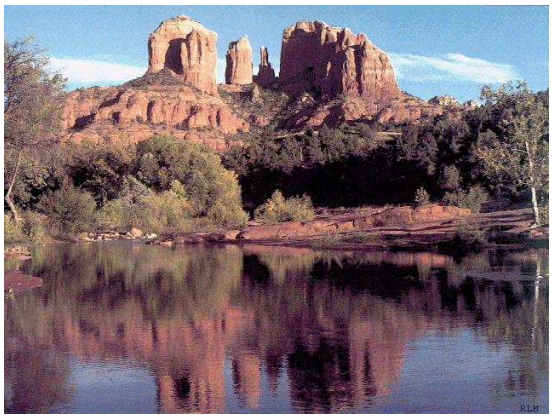
A 3.1 ábrán láthatjuk az adatrejtés „fázisait".

A másik esetben természetesen nem képként kell kimentenünk a kinyert adatot, hanem bájtonként ki kell írnunk egy fájlba. A módszer viszont ugyanaz.

3.6.4. Adatrejtés bináris képbe

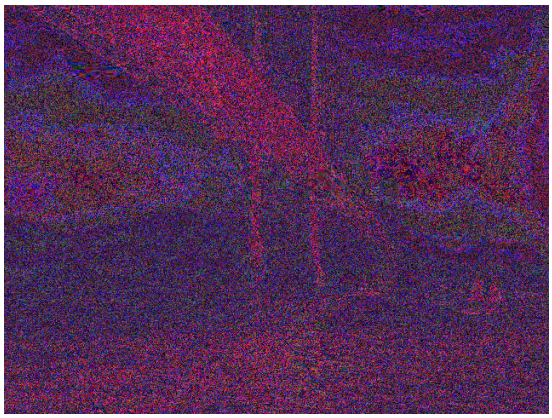
A kétszínű képek speciális kivételt képeznek. Míg RGB képek esetén hozzávetőleg 16 millió szín állt rendelkezésre, addig kétszínű képek esetén a színek száma értelemszerűen kettő: azaz egy képpont vagy fekete, vagy fehér. Természetesen nem változtathatjuk meg egyik színt sem, mert ezzel hatalmas változásokat hoznánk létre a képen.

[4] Épp megfelelő a MIME kódolás, ami kizárólag ún. „nyomtatható" karaktereket tartalmaz, bővebb információk az RFC 2045, RFC 2046, és az RFC 2047 szabványban találhatók.



(a)

(b)



(c)

3.1. ábra. Adatrejtő algoritmus lépései: (a) sztego-kép (és gyakorlatilag a hordozói is, mivel nem látszik köztük különbség), (b) a titkos kép, amit elrejtettünk / kinyertünk a hordozóból és (c) a hordozó és a sztego-kép közti különbség – Minden színcsatornát megszoroztunk 32-vel, hogy a különbség látható legyen. (A „legrosszabb" esetben a hordozó és a sztego-kép egy-egy pixeljének valamelyik színcsatornája közti eltérés 8, ha ezt 32-vel szorozzuk, 256-ot kapunk – magyarul a legnagyobb eltéréshez skálázzuk a képet.)

A megoldás az, hogy egy előre meghatározott pszeudo-véletlen sorozat szerint töröljük a kép pixeljeit, majd a vízjel bitjeit az előre törölt pixelekbe rakni (ahol a bit értéke 1, az a képen előtérként, ahol 0 ott háttérként jelenik meg) [2]. Ilyenkor a kulcs az álvéletlen sorozat generálási szabálya és kezdőértéke.

Másik lehetőség, hogy álvéletlen sorozatok helyett sémákat (maszkokat) használunk, amivel meghatározhatjuk, hogy mik azok a része a képnek, amit felülírhatunk.

1	1
1	0

(a)

0	1
1	1

(b)

3.2. ábra. Maszkok az adatrejtéshez: (a) 1-es, (b) 0-ás bitet reprezentáló maszk.

A módszer lépései a következők:

1. A 3.2 ábrán lévő maszkokat első lépésben el kell távoltani a képről. Ez történhet a maszk egyik véletlenszerűen választott elemének megváltoztatásával (a véletlen pontcserét addig alkalmazzuk egy-egy maskelemre, amíg az már nem egyezik meg egyik maszkkal sem).
2. Az összes olyan 2×2 méretű maszk használható bit rejtésére, ami a maszk egy elemének megváltoztatásával 1-es vagy 0-ás maszkká válik. Erre a feltételre azért van szükség, hogy minél kisebb módosításokat hajtsunk végre.

Természetesen ez egy parazló módszer, olyan értelemben, hogy sokkal kevesebb 2×2-es terület felel meg bitek tárolására, és egyáltalán a bitek tárolására, mint pl. az álvéletlen sorozattal működő adatrejtés.

A rejtett adatok kinyeréséhez egyszerűn csak 1-es és 0-ás maszkokat kell keresni a képen, és ezeket a megtalálás sorrendjében „összeolvasni”.

3.7. Szimmetrikus és asszimetrius szteganográfia

Az eddig bemutatott alkalmazásokban a titkos adat kinyeréséhez nem volt szükség az eredeti hordozóra. Ennek éppen ez az előnye, vagyis az eredeti hordozót, mint kulcsot nem

kell előzetesen eljuttani a célszemélyhez – ahol az átvitel vethet fel biztonsági kérdéseket. Az ilyen eljárásokat *vak eljárásokank* (*blindly methods*) nevezzük – a jelenleg használt technikák többsége ilyen.

Másik módszer – mikor az eredeti információ kinyeréséhez szükség van az eredeti hordozóra – leginkább egy példán keresztül mutatható be:

- A hordozó legyen egy kép.
- A titkos információ bitjeit valamilyen szisztéma szerint „terítsük" rá a képre.
- A titkos biteket adjuk hozzá a pixel értékéhez.

A titok kinyeréséhez szükség van az eredeti hordozóra is, a sztego-képből egyszerűen ki kell vonni az eredeti hordozót, hogy megkapjuk a titkot. Az ilyen eljárásokat nevezzük *letételjárásoknak* (*cover escrow*).

A letételjárások hátránya, hogy az eredeti hordozóra a rejtett adat kinyerésénél is szükség van. Bár ez biztonságosabbá teszi a módszert, az eredeti hordozó eljuttatása a „felhasználóhoz" újabb problémákat vet fel. Ezért a letételjárásokat nem üzenetváltásra, hanem vízelezésre szokták használni.

3.8. Megjegyzés

A fejezet végén fontos megemlíteni, hogy az LSB módszer ugyan sok adat elrejtését teszi lehetővé, ismertsége miatt azonban nem alkalmas biztonságos adatrejtésnek. Továbbá az eddigi példákban nem használtunk semilyen kulcsot az adatrejtő algoritmusokban, és tudjuk azt, hogy kizárólag az nem nyújt védelmet, hogy csak mi ismerjük a titkosító algoritmusunk működését. Célszerű tehát gyakorlati implementációk esetén erre külön figyelmet fordítani.

4. fejezet

Vízjelezés

Ebben a fejezetben a szteganográfia egy kiemelkedő alfejezetét, a vízjelezés témakörét vizsgáljuk meg. A vízjel egy olyan rejtett (vagy épp nem rejtett) információ egy „hordozóban", ami a hordozó eredetiségét, vagy eredetét igazolja.

4.1. Alapok

Mielőtt mélyebben belemélyednénk a vízjelek tanulmányozásába, nézzük meg a szteganográfia alkalmazásait a 4.1 ábrán.

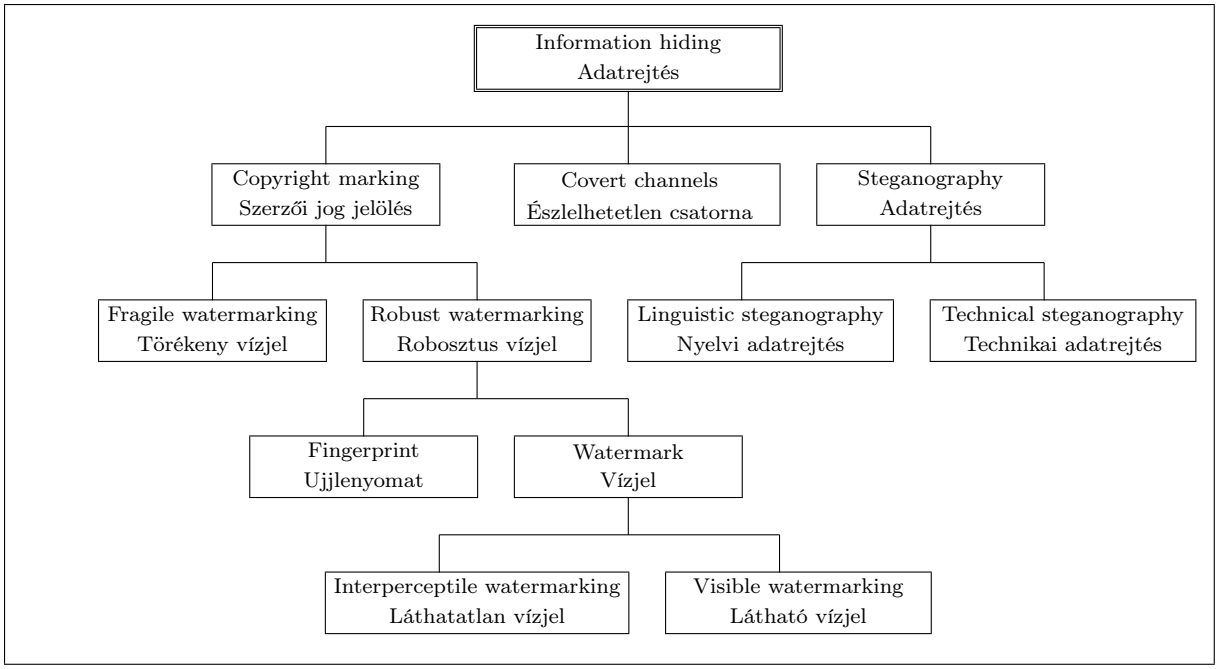
A szteganográfia leggyakoribb felhasználási módjai napjainkban az elektronikusan terjesztett publikációk védelme: az eletronikus vízjel vagy ujjlenyomat alkalmazása. Emellett természetesen képeket (ld. Adobe Photoshop) vagy audió CD-ket is el lehet látni egyedi azonosítóval. A módszer biztosítja, hogy az egyedi azonosító minden másolatban – gyakorlatilag eltávolíthatatlanul – benne legyen. Ezt nevezzük vízjelezésnek (*watermarking*), és szinte kizárólag a digitális médiában használják. A módszer maga nem más, mint egy kisebb adat elrejtése egy hordozóba, ahonnan később kinyerhető, vagy legalább detektálható. A vízjel lehet *látható* (*visible watermark*) vagy láthatatlan (*invisible watermark*), attól függően, hogy a vízjel észrevető-e vagy nem – és ez nem korlátozódik a képekre.

Másik alkalmazási lehetőség az ujjlenyomat (*fingerprint*), amit elektronikus információk (képek, hangok, stb.) nyomon követésére használnak. Az ujjlenyomat ott van minden másolatban, így alkalmas annak azonosítására. Az ujjlenyomat tulajdonsága, hogy ha

valaki egy másik ujjlenyomatot tesz pl. a képre, akkor a régit nem írja felül, hanem mindkettő rajta lesz a képen – csak azt tudjuk, hogy ott van, azt nem, hogy hol, ezért nem lehet felülírni. Ha ujjlenyomatként egy eleve redundáns információt használunk (pl. egy képet), akkor nagyobb az nagyobb valószínűséggel áll ellen a különböző behatásoknak. Nem elvárás az ujjlenyomat teljes egészének kinyerése a médiából, elegendő annak a detektálása.

Általánosságban kijelenthetjük, hogy míg a törékeny vízjel az edetiséget, addig a robusztus vízjel az eredetet igazolja. Ha a törékeny vízjel sérül, az eggyértelműen jelzi, hogy a hordozója megváltozott. A robusztus vízjel ellenállóbb a külső behatásokkal, ezért ha egy megváltoztatott képben is kimutatható, az az eredetet bizonyítja.

Meg kell jegyeznünk, hogy a vízjelezés, és ugyanígy a digitális ujjlenyomat az adatrejtés egy-egy alkalmazási területe. Tehát ha egy hordozóban saját névjegyünket rejtjük el, és hordozónak tekintjük minden képünket, akkor ujjlenyomatról beszélünk.



4.1. ábra. Az adatrejtés alkalmazási területei.

4.2. Vízjelek

A vízjel, mint fogalom az adatrejtés egy részhalmaza. A rejtett adat attól lesz vízjel, hogy arra szeretnénk használni. Ha például a titkos üzenet a kép készítőjének a neve, és annyiszor belerejti a képbe, ahányszor csak tudja, akkor vízjelről beszélünk. Ha a készítő tudja, hogy hányszor rejtette el a vízjelét a képen, kiolvasásnál ezeket megszámolva tudni fogja, hogy megváltoztatták-e a képét, és ha igen milyen mértékben.

A fenti példa egyben példa a láthatatlan vízjele is, mivel kihasználtuk az adatrejtésnek azt a tulajdonságát, hogy nem okoz észrevehető módosításokat képen. Emellett természetesen megjelölhetjük úgy a képeinket, hogy az látható legyen a képen is. Radikális esetben a kép közepére beleírjuk a copyrihgt információt úgy, hogy minden képpont intenzitását egy adott értékkel megnöveljük. Ez egy szemmel is észrevehető módosítás, észlelhető, viszont eltávolíthatatlan, mert azzal a képet is tönkretesszük.

4.2.1. Ötletek

Mivel a vízjel egy üzenet, amit elejtünk a hordozóban (most képben), célszerű elgondolkozni azon, hogy tudnánk ezt megvalósítani, vagy kis ötlettel javítani a módszeren.

Négyzetes vízjelek

Nyilvánvaló, hogy minél többször szerepel a vízjel a képen, annál biztosabban megtalálható lesz a módosítások után is. Ha például a támadó kivág egy részt a képből, a több elrejtett vízjel közül egy-kettő valószínűleg megmarad. Természetesen a tömörítésre továbbra is érzékeny a módszer – viszont a tömörítésre nem érzékeny SSIS esetén belerejthetjük a vízjelet több frekvencia-ablakba is.

Az alap adatrejtésben az üzenetet a hordozóba folyamatosan helyezzük el, például ha az üzenet hossza megegyezik a kép egy sorának hosszával, akkor minden sorban ugyanaz az információ lesz megtalálható. Ha a kép bal vagy jobb oldalából levágunk egy részt, az információ sérül.

Ha az üzenetet négyzetesre alakítjuk, azaz pl. egy 32×32-es négyzetbe írjuk (és így 1024 bájtos kapacitást éünk el a 3.6 fejezetben bemutatott módszer szerint), akkor kevésbé lesz

érzékeny az információ az oldalirányú levágásra. Beláthatjuk, hogy egy ez szórt adatrejtő módszer, ahol a szórás módszere nem más, mint egy adott méretű négyzetbe történő beírás.

Ha a vízjel elhelyezője eltárolja – akár a képbe rejtett titkos információként –, hogy hány vízjelnek kell lennie a képen, akkor ellenőrzésnél tudni fogja, hogy a kép módolsult-e, és ha igen, mennyire.

Típusát illetően ez egy törékeny vízjel, mert a sztego-kép megváltozása esetén a vízjel is módosul.

Több vízjel alkalmazása

Természetesen merülhet fel a kérdés, hogy csak egyfajta vízjel használható-e egy-egy képen. Természetesen igen, ahogy ez felmerült [4]-ben is. Nagyobb biztonságot kapunk, ha a láthatatlan vízjelet kombináljuk a látható vízjelekkel. Ha előbb a szakasz kezdetén leírtaknak (vagy a maszkolásnak) megfelelően valamilyen látható vízjelet rakunk a képre, és aztán pl. négyzetes láthatatlan vízjelezést alkalmazunk, a képünk kétszeresen védetté válik.

4.3. Digitális ujjlenyomat

A számítástechnikában a digitális anyagokból nagyon egyszerűen készíthető másolat. Ha valaki ezzel visszaél – sokszorosítja és eladja a terméket, és ezzel jelentős kárt okoz a gyártónak –, azt kalózkodásnak nevezzük [5]. Aki ilyen tevékenységet folytat *kalóznak* vagy *árulónak* neveznek. Ha több áruló szövetkezik, azt *koalíciónak* nevezzük.

Természetes igény, hogy valamilyen módon megjelöljük a digitális anyagokat (szoftvereket, kép- ill. hanganyagokat), hogy aztán később azonosítani tudjuk őket. Megoldás lehet például a sorozatszám. Ha felbukkan egy másolásgyanús anyag, elegendő ezt a sorozatszámot ellenőrizni. Természetesen regisztrálni kell a sorozatszámot, és a hozzá tartozó felhasználó adatait. Ezt nevezzük *ujjlenyomatnak*.

Az ujjlenyomat használatára évszázados példák léteznek. A logaritmusos táblákat úgy jelölték meg – látták el ujjlenyomattal –, hogy bizonyos értékekbe apró hibákat csempésztek. Például a $\log 3$ értékében a tizedesvessző utáni tizedik számjegyet megváltoztatták[1],

[1] A példa [5]-ből való.

ami a számolást nem befolyásolta, viszont egyértelműen azonosította a „művet”. Garantálni kell azt, hogy az ujjlenyomatot se észrevenni, se eltávolítani ne lehessen. Ma számítógéppel könnyen ellenőrizhetők a logaritmusértékek, tehát egy ilyen megvalósítás már nem lenne életképes. Másik igény, hogy az ujjlenyomat ne okozzon „működésbeli eltérést” az eredetihez képest. A logaritmusos táblák esetén a tizedesvessző utáni tizedik számjegy megváltoztatása nem okoz gyakorlati hibákat, míg képek esetén az ujjlenyomat jelenléte nem befolyásolja a kép élvezetét.

4.3.1. Igazságos kódok

Új probléma merül fel akkor, ha több áruló koalíciót alkot, hogy így felfedjék az ujjlenyomat jelenlétét. Továbbra is vegyük példaként a logaritmustáblák esetét. Tegyük fel, hogy minden táblában máshol van az ujjlenyomat. Ha többen szövetkeznek, és összevetik a példányaikat, ki tudják szűrni belőle a megváltozott pozíciókat, ahol valószínűleg az ujjlenyomat található. Ekkor azt mondjuk, hogy a pozíció a koalíció számára látható.

Tegyük fel, hogy a koalíció három árulóból áll, és az ujjlenyomat is három helyen található az anyagban. Tegyük fel továbbá, hogy az egyik ujjlenyomat-bit (*jelölés*) mindhárom árulónál ugyanaz. Ekkor természetesen a koalíció nem tudja kiszűrni mindhárom jelölést, tehát nem észlelik az ujjlenyomat egészét.

Természetesen, ha egy koalíció felderítette az ujjlenyomat jelöléseit, megteheti, hogy kicseréli őket, így esetleg más felhasználóhoz rendelt ujjlenyomatot állítva elő, így „megvádolva” az adott felhasználót. Ha a kód teljesíti azt a tulajdonságot, hogy egy k árulót tömörítő koalíció nem tud megvádolni egyetlen koalíción kívüli felhasználót sem, akkor k-igazságos kódról beszélünk. Egy kód pontosan akkor k-igazságos, ha egy k méretű koalíció által generálható kódszavak közül csak a koalíció tagjainak ujjlenyomatai lesznek valódi kódszavak.

4.3.2. Megjegyzés

Megjegyzendő, hogy az ujjlenyomat létrehozható úgynevezett *hash* függvényekkel is, mint az MD5[2], vagy SHA1[3]. Ezek a függvények a klasszikus kriptogáfia tudományába

[2] RFC 1321 (rfc1321) – The MD5 Message–Digest Algorithm.
[3] RFC 3174 (rfc3174) – US Secure Hash Algorithm 1 (SHA1).

tartoznak, és egy fix hosszúságú (SHA1: 160 bit; MD5: 128 bit) „ujjlenyomatot” hoznak létre a bemenetükről, ahol az ujjlenyomat kifejezés természetesen nem vízjel, mint az előző példában. Az ilyen lenyomatra igaz, hogy az input legkisebb módosulása esetén is megváltozik. Természetesen ha egy 128 bites ujjlenyomat egy pl. 2 GB-os fájl minden megváltozásakor módosul, akkor előbb-utóbb ugyanazt a ujjlenyomatot vissza kell kapnunk. Ennek a problémának a megoldása az, hogy gyakorlatilag lehetetlen úgy megváltoztani a bemenetet, hogy az ujjlenyomat és a bemenet tartalma (tág értelemben) is változatlan maradjon. Például egy csekken nem írhatunk egy egy százast egy ezressé.

4.4. Vízjelek tulajdonságai

A következő tulajdonságok, és azok magyarázatai [1]-ból valók.

Robusztusság A beágyazott vízjel robusztus, ha az akkor is megbízhatóan kinyerhető a hordozóból, ha a hordozót módosították (de nem torzították el teljesen). Vagyis a változtatások (transzformációk) ellenére is visszaállítható legyen a vízjel. Ezek a transzformációk lehetnek:

- lineáris és nem lineáris szűrők,
- veszteséges tömörítés,
- átméretezés, skálázás
- forgatás,
- zaj hozzáadása,
- képrészlet kivágása,
- A/D és D/A átalakítások,
- egy szűk környezet permutálása (képen egy maszk pixeljeinek permutálása),
- környezeti átlagolás,
- formátumkonverziók,
- ...

A robusztusság nem véd azok ellen a támadások ellen, amik a vízjel-beágyazás módszerét ismerik, és úgy próbálják károsítani a vízjelet.

Nem-detektálhatóság Ez a tulajdonság a legfontosabb a rejtett kommunikációban. A rejtett információ (vízjel) nem detektálható, ha a sztego-adat (vízjelezett kép) konzisztens az eredeti hordozóval (ugyanaz a kép vízjel nélkül). Ha például egy módszer a képen jelenlévő zajt használja a rejtett információ tárolására, akkor a sztego-kép zaja nem okozhat statisztikai eltéréseket.

Észlelhetetlenség (láthatatlanság) Ez a tulajdonság az emberi észlelést veszi alapul. A rejtett információ akkor észlelhetetlen, ha az emberi szemlélő emberi tulajdonságaiból adódóan nem veszi észre. Erre példa, hogy az emberi szem sokkal érzékenyebb a zöld a színre, mint a pirosra.

Biztonság A vízjel biztonságosnak minősül, ha a beágyazott információ nem észrevehető, nem pusztítható el, nem hamisítható meg, habár a támadó rendelkezésére áll a vízjel előállításának ismerete a kulcs nélkül.

Komplexitás Azt az erőforrás-ráfordítást jelöli, ami ahhoz szükséges, hogy a vízjelet kiolvassuk és újra elrejtsük.

Kapacitás Ez a paraméter jelöli, hogy mennyi információ rejthető el a hordozóban, és hogy hány vízjel vihető fel a hordozóra.

5. fejezet

Összefoglalás

Végezetül egy pár jellemző gondolatot szeretnénk megosztani. Azt már láttuk, hogy az adatrejtő algoitmusok nagyon sokfélék lehetnek, sőt kis gondolkodással saját magunk is állíthatunk elő újabb és újabb algoritmusokat – persze nem árt pár biztonsági elvárást szem előtt tartani.

Igazából az, hogy mi a vízjel vagy az ujjlenyomat, az adatrejtés felhasználásától függ. Ha valamilyen egyedi információt akarunk elrejteni a képben, akkor az vízjel, ha ez épp copyright információ akarunk nevezhetjük ujjlenyomatnak – természetesen elég homályos megközelítésben.

A következőkben segítséget nyújtunk a vízjelek csoportosításához. Először nézzünk néhány példát az igények szerinti csoportosításra:

Szerző-azonosítás: Robusztus vízjel. A szerző, stb. egy egyértelmű jelölést rak a hordozóra, amivel tanúsíthatja, hogy ő a szerző.

Felhasználó-azonosítás Ujjlenyomat-vízjel. A felhasználótól függően kerül különböző vízjel a hordozóba.

Megjegyzés Ilyen típusú vízjelekkel megjegyzések fűzhetők a hordozóhoz.

Az eljárás-szerinti osztlyozásra példa a következő felsorolás:

Láthatóság szerint

- látható vízjel
- láthatatlan vízjel, ahol a láthatóság az emberi észlelhetőséget jelenti.

Robusztusság szerint

- robusztus vízjel
- törékeny vízjel

A vízjel ellenőrizhetősége szerint szerint

- titkos (szimmetrikus)
- nyilvános (asszimetrikus)

Kapacitás szerint

A. függelék

A BMP formátum

Minden BMP fájl egy fejléccel kezdődik, ld. A táblázat.

A.1. táblázat. BMP formátum fejléce, és a mezők jelentése.*Csak azok jelentése van feltüntetve, amik az adatrejtés szempontjából érdekesnek mondhatók.*

Mezőnév	**Méret**	**Jelentés**
BM	2	„BM” – a fájl belső azonosítása miatt
Size	4	a fájl mérete
Reserved1	2	
Reserved2	2	
OffBits	4	
HeadSize	4	a fejléc további mérete
Width	4	a kép szélessége
Height	4	a kép magassága
Planes	2	bitsíkok száma
BPP	2	Bit Per Plane
Compression	4	0 = nincs tömörítés
IMGSize	4	a képleíró rész mérete
DPI_X	4	vízszintes nyomtatási felbontás
DPI_Y	4	függőleges nyomtatási felbontás
Reserved3	4	
Reserved4	4	

A BMP fáj fejlécét a képleíró rész követi. Egy átlagos BMP fájl esetén a Planes érték általában 1, a BPP pedig 24.

B. függelék

A WAV formátum

A BMP formátumhoz hasonlóan a WAV fájl is egy fix hosszúságú fejléccel kezdődik, ld. B.

B.1. táblázat. WAV formátum fejléce, és a mezők jelentése. *Csak azok jelentése van feltüntetve, amik az adatrejtés szempontjából érdekesnek mondhatók.*

Mezőnév	**Méret**	**Jelentés**
RIFF	4	„RIFF” – a fájl belső azonosítása miatt
rLen	4	a RIFF blokk hossza
WAVE	4	„WAVE” – a fájl belső azonosítása miatt
FMT_	4	„FMT_”
fLen	4	formátum rész hossza, ami tulajdonképpen az eddigi 16 bájt
Format	2	1 = PCM
Channels	2	1 = monó, 2 = sztereó
SamplesPerSec	2	mintavételezési freqvencia
AvgBytesPerSec	2	
BlockAlign	2	
Reserved1	2	
Reserved2	2	
BitsPerSample	2	felbontás
DATA	4	„DATA”
dLen	4	az ez után következő DATA blokk hossza

A következő rész a hangadatok tárolására szolgál. Vegyünk példaként egy 16 bites PCM kódolású tömörítetlen fájlt. A 16 bit két bájton tárolódik, a szokásos Intel-platformon

megszokott formában[1]. Ha a minta sztereó, a bal és a jobb csatorna adatai felváltva tárolódnak[2].

B.1. LSB módszer

A módszer nagyon hasonlít az alap BMP fájl feldolgozására, annyi különbséggel, hogy egy-egy minta nem egy bájton található, hanem 16 bit esetén két bájton. A legegyszerűbb a dolgun, ha 8 bitet változtatunk meg. Ekkor a fejléc beolvasása után felváltva olvassuk be az alcsony és magas helyiértékű biteket tartalmazó bájtokat. Minden alcsony helyiértékű bájtot kicserélünk az általunk elrejteni akart üzenet soron következő bájtjára.

Természetesen, ha nem 8 bitet kívánunk felhasználni, akkor hordozó soron következő bájtjának a bitjeit egyenként kell megfognunk, célszerű tehát egy pufferben tárolni az elrejtendő biteket. Ez kissé nehezíti az implementációt.

Célszerű készíteni egy kiegyészítő rekordot, amiben leírjuk az üzenet méretét, az üzenet-fájl nevét, azt, hogy hány bitet használunk fel a hordozó egy-egy mintájából, és egy esetleges ellenőrző összeget, amiből tudni fogjuk, hogy rendben ment-e a kiolvasás. Az üzenet méretének, és fájl-nevének ismerete nem okoz különsebb problémát a megfejtésnél, viszont a ha nem tudjuk, hogy hány biten rejtettünk el adatokat, az jelentősen megnehezíti a rejtett adat kiszedését – természetesen próbálkozással ez is megoldható.

[1] Először az alsó 8 bit, majd a felső 8 bit.
[2] Előbb a jobb, majd a bal csatorna.

Irodalomjegyzék

[1] Jana Dittmann: *Digitale Wasserzeichen*. 2000, Springer–Verlag. (*német nyelven*).

[2] Hea Yong Kim – Amir Afif: A secure authentication watermarking for halftone and binary images. 2003., *XVI Brazilian Symposium on Computer Graphics and Image Processing. (angol nyelven).*

[3] Tóth Mihály: Szteganográfia avagy adatrejtés és elektronikus vízjelek. 2004. (SZGTI jegyzet).

[4] Saraju P. Mohanty – K.R. Ramakrishnan: A dual watermarking technique for images. 1999., *Proc. 7th ACM International Multimedia Conference, ACM-MM'99, Part 2, pp. 49-51, Orlando, USA, Oct. 1999. (angol nyelven).*

[5] Mátray Péter: A szerzői jog védelme a matematika eszközeivel. Szakdolgozat (Eötvös Loránd Tudományegyetem, Budapest). 2003.

[6] Virasztó Tamás: *Titkosítás és adatrejtés*. 2004, NetAcademia.