A Key-Recovery Attack on Authentication Watermarking by Li and Yuan

Hans Georg Schaathun[*]
University of Surrey
Department of Computing
GU2 7XH Guildford
England
H.Schaathun@surrey.ac.uk

ABSTRACT

We present a key-recovery attack on a watermarking scheme for authentication and localisation due to Li and Yuan. The attack exploits a lack of diffusion in the system. Each bit of the key affects only a small region of the watermarked image. Even though a brute-force attack is intractible, an exhaustive search is possible by considering one region at a time, and thereby recover the key.

General Terms

security

Keywords

Image authentication, watermarking, MAC

Categories and Subject Descriptors

E.m [**Data**]: Miscellaneous—*Watermarking*; E.3 [**Data**]: Data Encryption—*Hash functions, MAC*

1. INTRODUCTION

In many contexts it is essential to be able to tell if a message is authentic or if it has been modified by an adversary. This applies for instance to forensic photography, where someone may want to forge or disable evidence, by doctoring an image.

The most well-established solutions are cryptographic techniques such as digital signatures or message authentication codes (MAC). These solutions are mature and widely trusted. The signature or the MAC is transmitted together with the message, and the receiver can verify that the message fits the MAC or signature. While an adversary can modify the message, it is computationally infeasible to generate a matching MAC or signature.

Alternative solutions have been proposed in digital watermarking. Watermarking, in general, allows us to embed a message (*watermark*) within another data file (such as an image) called the *host*. The embedding is done by imperceptible changes to the host, so that the watermarked host can replace the original for all practical purposes. A basic application of watermarking is to extend a legacy data structure. Metadata can be embedded in the original data, without requiring any aditional fields.

Fragile watermarks are designed such that any change to the host will destroy the watermark. Such watermarks can be used for authentication of the host; if the watermark cannot be recognised, then the host will not be authenticated. Semi-fragile watermarks goes one step further. They are designed to survive legitimate changes, possibly such as compression or noise, but will be destroyed by unauthorised modification such as doctoring.

Cox *et al.* [1] present two advantages of digital watermarking for authentication. Firstly, the authentication information is hidden as an inherent part of the message (host). Thus it can be incorporated in a legacy data structure. It avoids the appended signature of cryptology.

The other advantage is more subtle. Because the watermark is hidden in the data, it will undergo the same transformations as the data. By observing the transformed watermark, it may be possible to determine the exact transformation the message has undergone, and possibly undo it.

Recent fragile and semi-fragile watermarking schemes offer additional functionality on top of the basic authentication. Most importantly they aim to identify the location of any changes, flagging any regions where the host has or may have been modified. Some solutions also offer *self-recovery*, i.e. the ability to recover the modified regions in the event of unauthorised modification.

Cryptographic solutions still have the advantage of being based on a mature theoretic framework, which has evolved over centuries. New solutions are scrutinised by a score of independent researches within few years. Unfortunately, the methodology developed in cryptology is rarely applied in watermarking.

In this paper we study the security of a particular watermarking scheme due to Li and Yuan [3], and we develop a key-recovery attack for this scheme. The attack requires

[*]The author wishes to thank Dr. Chang-Tsun Li for helpful discussion in person and by email. The work was partially funded by EPSRC under Grant EP/E056407/1.

a few known watermarked images. For Li and Yuan's suggested parameters two such images suffice. Our attack works even though the secret key of the system is long enough to make a brute-force attack intractible.

The attack is significantly faster than brute force. By exploiting the knowledge of the watermark algorithm, we are able to recover the key by an exhaustive search on a small key segment at a time. For Li and Yuan's suggested parameters, we consider 25 key bits at a time. Although the attack is straight-forward from a cryptographic viewpoint, we believe that the paper will be very usefull for designers of watermarking systems, to avoid similar vulnerabilities.

In the next section we present Li and Yuan's scheme, and establish notation to be used in the sequel. In Section 3, we present the attack, and in Section 4 we discuss possible improvements to the attack and the watermarking scheme. Finally, there is a concluding section.

2. THE LI-YUAN SCHEME

Let $\mathcal{I}(x,y)$ be an image, and b a natural number. Following [3], we assume (without loss of generality) that the image is 8-bit grayscale. The b least significant bits from each pixel are discarded, and will be replaced by watermark information. The $8-b$ most significant bits form the *significant image* $\mathcal{J}(x,y) = \lfloor \mathcal{I}(x,y)/2^b \rfloor$. Li and Yuan suggest $b = 2$.

The system uses a *secret watermark image* $w(x,y)$; with the same dimensions as $\mathcal{I}(x,y)$ and b bits per pixel. The original paper generates w using a pseudo-random number generator (PRNG) with a secret key K as the seed. However, K has no other use, and it is sufficient for the attacker to recover w. Our attack makes no assumptions about the PRNG, and it will work even if w is a truly random image.

We write the watermarked image $\mathcal{W}$ as

$$\mathcal{W}(x,y) = 2^b \mathcal{J}(x,y) + a(x,y),$$

where $a(x,y)$ is b bits of authentication information. Let $N_k(x,y)$ denote a $k \times k$ square region centred at (x,y). The authentication information $a(x,y)$ is a function of $w(i,j)$ and $\mathcal{J}(i,j)$ for all $(i,j) \in N_k(x,y)$. In principle a non-square neighbourhood could be used instead of $N_k(x,y)$

Firstly, we define the so-called *secret sum*,

$$S(x,y) = \sum_{(i,j)\in N_k(x,y)} (-1)^{w(x,y)+w(i,j)} \mathcal{J}(i,j). \qquad (1)$$

Note that the secret sum depends only on the least significant bit of each pixel of w. We define the *embedding key* to be $\kappa(x,y) = w(x,y) \mod 2$, i.e. the least significant bit plane of the watermark image. In the formula for S we can replace w by κ.

Li and Yuan use $S(x,y)$ as a seed for a PRNG, from which they then extract b random bits $v(x,y)$. The authentication information is defined as $a(x,y) = v(x,y) \oplus w(x,y)$ where $\oplus$ is bit-wise exclusive or.

The extractor, knowing κ, can calculate $v(x,y)$ from the significant image in the same way the embedder does. It can also extract $a(x,y)$ from the image, and calculate the watermark as $w'(x,y) = a(x,y) \oplus v(x,y)$. If the image is authentic, then $w' \equiv w$.

```
function TestKey ( $\kappa$, $(x,y)$ )
    $v_1 := \mathrm{Extract}_\kappa((x,y), \mathbf{x}_1)$
    if $v_1 \mod 2 \neq \kappa(x,y)$ return FALSE ;
    $\mathbf{v}_2 := \mathrm{Extract}_\kappa((x,y), \mathbf{x}_2)$
    if $v_1 \neq v_2$ return FALSE ;
    return TRUE
END function
```

Algorithm 1: Subroutine testing if a given key is consistent with the known watermarked images.

The *localisation map* is defined as

$$L(x,y) = \begin{cases} 0 \text{ if } w'(x,y) = w(x,y), \\ 1 \text{ if } w'(x,y) \neq w(x,y). \end{cases} \qquad (2)$$

All the ones in the localisation map correspond to pixels (x,y) where tampering has occured in the neigbourhood $N_k(x,y)$.

Li and Yuan [3] suggest a 5×5 square neighbourhood ($k = 5$). This means that the secret sum depends only on 25 key bits. The authentication information $a(x,y)$ depends on the same 25 key bits, as well as $b-1 = 1$ additional bit(s) from $w(x,y)$. This is only 26 key bits with the proposed parameters.

An important principle in cryptography is that of *diffusion*. Every key bit (and message bit) should be 'diffused' so that it affects every bit of the output. The Li-Yuan scheme clearly exhibits insufficient diffusion from a cryptographic point of view.

Finally, we will summarise the security model of Li and Yuan. The threat they consider is an adversary who tries to pass off a false image as a genuine one. The false image could either be a new creation or a modification of a genuine one. The watermarking system is intended to control this threat.

The system is Kerckhoffs compliant, in other words, the complete algorithm is public knowledge (it has been published [3]). Only the key (K or w) is kept secret. If the system is comprimised, only that instance is affected. Other instances, using a different key, would not be affected. We assume that the system is used to protect several images with the same key (and Dr. Li has confirmed this assumption in private correspondence). Having one key per image would clearly be impractical and unscalable for many applications.

The attack we design is a watermarked only attack, i.e. it requires a few images watermarked with the same key, as well as the algorithm. We do not need access to a keyed decoder (oracle), nor do we need any known host images. The attack allows the adversary to recover the key (watermark image), which allows him validly to watermark any image.

3. THE ATTACK

Pseudo-code for the `attack` is given in Table 2. We assume that two validly watermarked images $\mathbf{x}_1$ and $\mathbf{x}_2$ are known to the attacker. The algorithm returns a list L of possible keys.

We need to introduce a notation for the restriction of a vector or matrix. Let $\mathbf{a}$ be a vector or a matrix, and α a set of indices (coordinate positions). The restriction $\mathbf{a}|\alpha$ is the

```
function Attack
$L := \{\mathbf{empty\_string}\}$ (* Return list *)
$B := \emptyset$ (* Key bits scanned so far*)
for each pixel $(i,j)$
  $L_0 := \emptyset$ (* Work space for $L$ *)
  $\beta := N(i,j) \cap B$ (* Old key bits used *)
  $\alpha := N(i,j) \backslash B$ (* New key bits considered *)
  $\kappa := \mathbf{0}$
  Sort $L|\beta$
  (* elements equal on $\beta$ are consecutive *)
  for $W_1 \in L|\beta$, $W_2 \in 2^\alpha$
    $\kappa|_\beta = W_1$
    $\kappa|_\alpha = W_2$
    if not TestKey($\kappa$, $(i,j)$) then break
    for $\kappa' \in L$, such that $\kappa'|_\beta = W_1$
      $\upsilon := \kappa'$
      $\upsilon|_\alpha := \kappa|_\alpha$
      $L_0 := L_0 \cup \{\upsilon\}$
    end for
  end for
  $L := L_0$
end while
RETURN $L$
```

Algorithm 2: The main attack routine.

subvector formed by taking only the elements of $\mathbf{a}$ which are referenced by the members of α.

For each pixel (i,j), the `Attack` considers every possible subkey on $N_k(i,j)$, and tests if it is consistent with the known watermarked images $\mathbf{x}_1$ and $\mathbf{x}_2$.

When a subkey κ is accepted, the inner `for` loop of the `attack` will form all possible subkeys by merging κ with compatible subkeys identified in previous iterations (as stored in L). The resulting keys are stored in the list L_0, which will in turn become the new set L for subsequent iterations.

The consistency check is performed by the `TestKey` subroutine shown in Table 1. Two tests are made. The first compares the extracted watermark for $\mathbf{x}_1$ with the appropriate bit of κ. Since one bit is compared, a random key has probability $\frac{1}{2}$ of passing the test. The second test compares the extracted watermarks from the two images. Since b bits are compared, a random key has probability 2^{-b} of passing the test. For $b = 2$ we expect 7 keys in 8 to fail at least one of the tests.

The auxiliary function $\text{Extract}_\kappa(i, \mathbf{x})$ extracts the watermark bits from pixel i of image $\mathbf{x}$ using key κ.

It is important that the main `for` loop traverses the pixels in an efficient order. We suggest the following. Start with pixel $(0,0)$. If pixels (i,j) have been scanned for $0 \le i \le m$ and $0 \le i \le m$ for some m; we then scan pixel $(i', m+1)$ for $0 \le i' \le m$, and subsequently pixel $(m+1, j')$ for $0 \le j' \le m+1$. This way, we minimise the number of new key bits which will have been considered at each iteration of the loop.

With this order, we expect $E(\#L) = 2^{22}$ after the first pixel. The second and third pixels $(0,1)$ and $(1,0)$ each adds five key bits, and gains three information bits (on average). Thus we have $E(\#L) = 2^{26}$ after the third iteration. The fourth pixel $(1,1)$ only add one new key bit, but gains three information bits. Hence the expected list size $E(\#L)$ starts to decrease.

The iterations with the most work are thus for the third, fifth, and seventh pixels, when $E(\#L) = 2^{24}$ at the start. Adding 5 key bits, means that at most 2^{29} subkeys will be searched on average. (The actual number is likely to be slightly lower even than this, as some keys in L are going to be equal on the twenty bits considered in the round.) Cryptologists have long considered exhaustive searches over 40-bit keys feasible. Hence we are well within the bounds of feasibility.

Implementation issues are essential for efficiency, so it is instructive to consider how much we can achieve within a single word of computer memory. Suppose we consider an 8×8 image block. The 64 bits of the corresponding subkey fit within one long integer on a modern CPU. Sixteen 5×5 blocks fit within the 8×8 block. Thus we get to run sixteen tests, gaining 48 bits of information on average. The expected size of the remaining search space is $E(\#L) = 2^{16}$.

This means that the first 16 iterations can be implemented with a minimum of overhead, representing the subkeys as 64-bit integers. Starting on iteration no. 17, the number of potential keys is down to a very managable 2^{21}, and we can afford more overhead.

Note that if we have more known watermarked images, `TestKey` can be extended to reject more keys (on average), and the keyspace will shrink faster.

4. VARIATIONS AND IMPROVEMENTS

We have based our presentation so far on the design parameters suggested in [3]. In this section, we will first discuss a possible improvement of the attack. Subsequently we will discuss the effect of varying the design parameters on the efficiency of the attack. Finally, we will briefly discuss a fix to the watermarking system, which could prevent the attack.

4.1 Exploiting the secret sum

The attack as described does not exploit the structure of the secret sum, nor the PRNG. It would still work if these funcions were replaced by any other hash function with the same key length.

In many cases it is possible to speed up the attack by exploiting the simple form of $S(x,y)$. Let's illustrate this with an extreme, but simple example. Suppose you can find a $k \times k$ block of identical pixel values in the significant image. In this case

$$S(x,y) = (-1)^{\kappa(x,y)}(k^2 - 2w_\text{H}(\kappa|_{N(x,y)}))\mathcal{J}(x,y), \qquad (3)$$

where w_H denotes the Hamming weight. In other words, $S(x,y)$ depends only on the weight of the subkey and one bit $\kappa(x,y)$. Moreover, one subkey κ and its negation $(\kappa \oplus \mathbf{1})$ give the same secret sum.

This approach reduces the number of keys which have to be tested in the first iteration from 2^{k^2} to k^2, which is a massive saving. Even if the block is not completely monotonous, the approach can be applied to an extent colour by colour. The secret sum would be a sum of one term per colour, of the form of (3).

The effectiveness of the improvement depends on the host image, and a complete study is out of scope for this paper. However, most images do have considerable monotonous background regions, such as sky, and we only need very small regions to be monochrome. We also note that we do not require the region to be exactly the same colour; only the significant image matters.

4.2 Design parameters

The watermarking scheme has two important security parameters, k and b, and it is instructive to look at how they affect our attack.

For b, Li and Yuan identify a tradeoff between security and distortion. Increasing b will increase the distortion, but reduces the probability that a randomly attacked image will pass as authentic.

Unfortunately, increasing b makes our attack *more effective*. This is because `TestKey` would then have more bits to compare in the second test, and the probability of accepting a key is reduced. Reducing to $b = 1$ would slow our attack down, rejecting 3/4 of the keys per iteration, instead of 7/8. However, this could be compensated by having one extra known watermarked image.

Increasing k is more effective against our basic attack. Running an exhaustive search for $k = 7$ (2^{49} keys) is borderline feasible or worse. With $k = 9$ it is infeasible at present. However, the improvement described in Section 4.1 would reduce the impact of increasing k. Many images with large or moderate-sized monotonous regions would almost certainly still be vulnerable to attack.

Another disadvantage of increasing k [3] is that it decreases the accuracy of the localisation. An increasing number of watermark pixels would be affected for a single pixel changed in the significant image.

4.3 Fixing the vulnerability

The weakness of the Li-Yuan scheme is in the hash function, computing $v(x, y)$ from $\mathcal{J}$ in the region $N_k(x, y)$. For brevity, we let $m = (\mathcal{J}(x, y)|(x, y) \in N_k(x, y)$, and denot the corresponding hash value by $h(m) = v(x, y)$. The property required by the hash function h is known as *computation-resistance* [4].

Definition 1 (Computation-resistance) *The goal is to make it computationally infeasible for the attacker to generate, with a probability significantly larger than 2^{-b}, a new pair $(m, h_\kappa(m))$ of image data and corresponding hash value without knowledge of the secret key κ.*

This property is provided by so-called *message authentication codes* (MAC) in cryptography. If the property does not hold, an attacker can generate false image data m, with the corresponding hash value $h(m)$, and then embed $h(m)$ in the image. It follows from the property that the adversary must be unable to recover the secret key κ, lest he have all the information needed to compute $h_\kappa(m)$ for an arbitrary m.

This problem has been carefully studied in cryptography, and there is a range of MACs which could have been used in place of the secret sum and PRNG. Our attack would not have been feasible, if a secure MAC had been used, and there are examples of watermarking systems which do employ MACs. Puhan and Ho [5] use a so-called hash-based MAC (HMAC) [4].

Culnane *et al.* [2] study authentication watermarking for binary image representations of text. They use OCR to extract the text (corresponding to our significant image), and hash this using an OMAC, which is embedded by modulating on inter-word spaces. A similar problem is discussed in [6].

It should be noted that an arbitrary cryptographic hash function should not be used. Wong et al. [7] suggest to use a collision-free hash function such as MD5. This makes it impossible for the adversary to find a message pair (m, m') such that $h(m) = h(m')$, but not impossible to find a new pair $(m', h(m'))$ where $h(m') \neq h(m)$. The way they hash image data together with a secret key is known to be insecure when applied as a MAC [4].

5. CONCLUSION

We have devised a key-recovery attack for Li and Yuan's watermarking scheme for authentication. With Li and Yuan's suggested parameters, the computational complexity is well within what we normally consider feasible in cryptography. However, implementing the attack and establish the exact running time is an interesting open question.

This kind of cryptanalytic attacks have to be designed for each particular watermarking system. The principles behind the attack are, on the other hand, the same as those used to break ciphers for several centuries. Without using the understanding developed in cryptology during these centuries, we can hardly hope to resist such attacks in watermarking.

There are cryptographic primitives which can be used to generate watermarks which resist our attack. Now it is necessary to move past the misconception that watermarking be an alternative to cryptography. Watermarking is a complement to cryptography, and cryptographic primitives and methodology are essential to secure a watermarking system.

Although some watermarking schemes do employ cryptographic principles, and some employ them soundly, there are many celebrated scheme where no cryptological theory has been used. New schemes still appear without cryptological backing, especially in conferences. It is an interesting open problem to check if these schemes are sound, or if they can be broken using simple cryptanalytic techniques.

6. REFERENCES

[1] I. Cox, M. Miller, J. Bloom, J. Friedrich, and T. Kalker. *Digital Watermarking and Steganography.* Morgan Kaufmann, 2nd edition, 2007.

[2] C. Culnane, H. Treharne, and T. Ho. Authenticating binary text documents using a localising omac watermark robust to printing and scanning. In *The 6th International Workshop on Digital Watermarking*, volume 3304 of *Springer Lecture Notes in Computer Science*, 2007.

[3] C.-T. Li and Y. Yuan. Digital watermarking scheme exploiting nondeterministic dependence for image authentication. *Optical Engineering*, 45(12), Dec. 2006.

[4] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, Inc., 1997.

[5] N. B. Puhan and A. T. S. Ho. Secure authentication watermarking for localization against the holliman-memon attack. *Multimedia Systems*, 12(6):521–532, 2007.

[6] R. Villan, S. Voloshynovskiy, O. Koval, F. Deguillaume, and T. Pun. Tamper-proofing of electronic and printed text documents via robust hashing and datahiding. In *Proceedings of SPIE-IS&T Electronic Imaging 2007, Security, Steganography, and Watermarking of*

Multimedia Contents IX, volume 6505, page 65051T, San Jose, USA, 28 Jan. – 1 Feb. 2007.

[7] P. W. Wong and N. Memon. Secret and public key image watermarking schemes for image authentication and ownership verification. *Image Processing, IEEE Transactions on*, 10(10):1593–1601, Oct 2001.