available at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/diin

Digital Investigation

Impersonator identification through dynamic fingerprinting

*Chad M.S. Steel**, *Chang-Tien Lu*

Computer Science Department, Virginia Polytechnic Institute, Haycock Road, Falls Church, VA 22043, USA

ARTICLE INFO

Article history:
Received 18 August 2006
Received in revised form
10 March 2008
Accepted 17 March 2008

Keywords:
Digital watermark
Phishing
Intellectual property
Forensics
Fingerprinting
Impersonation

ABSTRACT

Tracking the source of impersonation attacks is a difficult challenge for investigators. The attacks are frequently launched from botnets comprised of infected, innocent users and web servers compromised by malware. Current investigative techniques focus on tracking these components. In this paper, we propose the Automated Impersonator Image Identification System (AIIIS), which allows investigators to track images used in impersonation attacks back to the original download from the source. AIIIS accomplishes this by digitally encoding the IP address, server, and time of the image download into the image itself through a digital watermark. AIIIS differs from other image fingerprinting techniques in its combination of dynamic fingerprinting and spread spectrum data hiding. Additionally, the intended goal of AIIIS is tracking impersonation attacks, where the image is a tool used, whereas in most digital rights management techniques, the misuse of the content itself is the primary concern. Our experiments show that the AIIIS system permits recovery even after post-acquisition manipulation of the image, making it a significant addition to the fight against impersonators. The deployment of a pilot of AIIIS was successful in identifying the source of an impersonation attack, and further success is expected with full deployment.

Published by Elsevier Ltd.

1. Introduction

Phishing schemes and online trademark infringement are both on-the-rise crimes which rely on impersonation. Phishing attacks hit one in four Internet users each month (AOL/NCSA, 2005), and annual losses due to phishing are projected to be anywhere from US$100 Million to US$1 Billion (Goth, 2005), and are expected to continue to increase. Online trademark infringement is used to lure unsuspecting users to buy from less than reputable sources. An example trademark infringement incident may involve an attacker who uses Viagra branding and images to lure an individual into purchasing stolen or counterfeit pharmaceutical products.

A typical impersonation attack uses two components – an email message sent to a large number of users and a website those users are encouraged to visit. Through obfuscation of the URL and basic social engineering tactics, an unsuspecting user is enticed to click on a link in the email. The link takes the user to a look-alike site, where their username and password (or other personally identifiable information) are captured for malicious use or they are tricked into purchasing goods from a shady source. Typical targets include large financial institutions, pharmaceutical companies, and online merchants, e.g., Citibank, Bank of America, Pfizer, and eBay.

The value of the Automated Impersonator Image Identification System (AIIIS) is based on the methods used in many impersonation schemes. The emails sent are frequently transmitted from infected zombie machines, making tracking ineffective. Likewise, many schemes now use infected web servers not directly tied to the attackers as well. The construction of the emails and sites, however, involves the acquisition of the original site graphics and layout to be able to fool the unsuspecting victims. Frequently, the site is visited from the origin machine of the phishing attacks – since the visit is not

* *Corresponding author*. Tel.: +1 610 639 3884.
E-mail address: csteel13@vt.edu (C.M.S. Steel).
1742-2876/$ – see front matter Published by Elsevier Ltd.
doi:10.1016/j.diin.2008.03.001

malicious, server-based analysis isn't likely to pick up the first visit as suspicious (or later identify it as such) using existing techniques. A review of 148 phishing sites listed on millersmiles.co.uk revealed 42 sites active and viewable over the course of three weeks. Of the active phishing sites, 23 of the 42 (55%) used local copies of images that were verified to be exact matches to those on the corresponding real sites. The other 19 active phishing sites linked directly to images on the corresponding real sites (Millersmiles, 2006).

To-date, most impersonation defense mechanisms are provided through prevention and/or detection controls. Products like AntiPhish, which comprises a browser extension that warns users when visiting untrusted sites (Kirda and Kruegel, 2005), fall into the prevention category (as do anti-phishing toolbars (Netcraft Inc., 2006), image-based authentication (Topkara et al., 2005), and two-factor authentication (Geer, 2005)). Techniques like Dynamic Security Skins, a method of user image comparison for authentication (Dhamija and Tygar, 2005), fall primarily into the detection category (as do server-log analysis tools like Corillian and chat monitoring from MarkMonitor (Geer, 2005)). Limited work has been done in relation to response controls – those which allow after-the-fact investigation of activity.

To address the response gap, AIIIS provides a mechanism by which images can be tracked from the point of origin (the website being spoofed in an impersonation attack) through the insertion of unique watermarks for every download. By generating a string containing details from each image request, including server name, requesting IP address, and date/time of the initial request, a unique token is created which can be correlated with entries in the web server log. This unique token is dynamically embedded in each individual image download as a symmetrically encrypted and hidden watermark for later recovery from spoofed sites. In addition to the challenges that watermarking focuses on, AIIIS has a similar purpose to steganography in that both the image and the covert channel information need to be transmitted and reconstructable, and that the uncovering and altering of the covert channel information is an undesirable event (Wang and Wang, 2004).

AIIIS is novel in that it differs from prior art both in its execution and its intent. In its execution, AIIIS uses overinsertion, the practice of inserting multiple copies of the message, to provide further resilience to alteration over single-insertion techniques. Additionally, AIIIS applies transform-based insertion over the entirety of an image instead of on a block-basis. For intent, AIIIS is designed to investigate attacks after-the-fact and dynamically embed user data in the image at the time of request, instead of static strings identifying the copyright holder. Finally, to protect the string from disclosure or effective alteration, AIIIS uses simple symmetric encryption with Advanced Encryption Standard (AES) (National Institute of Standards and Technology, 2001) and/or the Data Encryption Standard (DES) (National Institute of Standards and Technology, 1993) and a server-specific key.

This paper details how AIIIS applies spread-spectrum techniques and dynamic fingerprinting to contribute a robust solution to track impersonators. In Section 2, a pilot implementation is used to illustrate the use of the system. In Section 3, the four major approaches to investigate impersonation schemes are outlined and the applications of prior art to the schemes are defined. Section 4 introduces the AIIIS system and details the processes for string extraction and insertion. An error rate analysis is also provided which mathematically quantifies the resiliency of the system against individual bit error probabilities. In Section 5, the experimental results of image recovery are provided for an array of image sizes and three key operations – scaling, compression, and cropping. Finally, a comparison of single-insertion, block-based spread spectrum recovery likelihood is compared with that of AIIIS. Section 6 discusses the results and applicability to other areas, and Section 7 provides concluding remarks.

2. In practice

AIIIS was used successfully by a major retailer to track down the source of a counterfeit operation. AIIIS was deployed by the retailer as a pilot project, and had success within the first two months in thwarting a sophisticated attack.

2.1. *Background*

The retailer in question was subject to frequent phishing attacks, with an additional problem of having counterfeit goods sold using their branding. As such, the retailer was looking for solutions to track down the counterfeiters and piloted AIIIS as one of the tools in their suite. As the impersonation attacks became more sophisticated, simple monitoring of brand misuse online was insufficient. Shutting down rogue websites as they appeared became a fruitless exercise, and the company decided it needed to pursue the individuals perpetrating the attacks. The details of one attack and its investigation are noted below.

2.2. *Attack methodology*

The attack in this case consisted of several different ''companies'' setup with the same basic website template, but slightly different company names and text. The websites were all hosted on what appeared to be hijacked machines, primarily in Southeast Asia. Similarly, the DNS entries for the company were numerous and appeared to be registered through a series of hacked accounts. The websites used the branding and logos of the original retailer to give the appearance of being authorized merchants, but in reality were counterfeiters using the brand recognition to push their product. Purchases had their credit cards charged from one of several China-based businesses, and received forgeries of the real product which looked similar but did not meet the quality standards of the originals.

2.3. *Investigation*

The impersonating sites began coming to the attention of the retailer after consumer complaints about substandard goods. The goods were purchased online, and the individuals who complained provided the names of the sites where the purchases occurred.

The first investigative step taken was an attempt to identify all of the domain names associated with the counterfeiters. Because the impersonator's web design team had stolen the source code from an obscure bulletin board software package as part of its deployment (copyright infringement!), there was a unique search term that could be run against the major search engines to identify similar pages. In total, 20 separate domain names were identified which were related to the counterfeiting operation.

The next step was identifying the locations and registration information for the 20 domain names. A review of DNS entries showed a wide variety of registering individuals, mostly using second and third tier registrars. By doing a correlation of these entities, we were able to identify an additional 20 domain names associated with the counterfeiters. Because unwitting individuals had their registrar accounts hijacked, there was no easy way to trace the registrations back to the offenders.

Identifying the locations of the web servers hosting the sites proved equally challenging. Each of the domain names resolved to an IP address which appeared to change every 15 min. Working with one of administrators on one of the DNS servers of record, we were able to determine the server IP was changing every 15 min. All of the servers were running Windows XP, but with what appeared to be different configurations (based on versions, ports open, and other responses to queries). The majority of the servers appeared to be located on residential IP address blocks in the Philippines and other locations. After contacting several ISPs, we were able to determine the servers were part of a botnet and belonged to unsuspecting users.

One commonality amongst the servers was the branding images. We evaluated the branding images, and found that *all* of the branding images for a particular product came from a pilot website running AIIIS. Additionally, all of the images were the same download by the same user. This IP address was linked to a single broadband connection in Poland.

After identifying the common site, we worked with law enforcement in-country as well as law enforcement at the target location. The suspect who was linked to the IP address was questioned and admitted to creating the websites for another party he believed to be located in China. The Chinese government was engaged for investigation in-country, and that investigation is progressing. As of this writing, the others involved in the impersonation scheme are still under investigation.

3. Prior work

Prior work relevant to AIIIS can be divided into two categories – investigative techniques for impersonation attacks and approaches to fingerprinting images. The investigation of impersonation attacks is a relatively new area of interest, and no work has been done to provide a taxonomy of impersonation investigations based on the current, accepted investigation techniques. We provide a classification of investigation techniques from prior work based on the target of the initial investigation. Image fingerprinting is a more mature field, and we present several techniques currently used to protect copyrighted information, as well as spread-spectrum techniques to embed data in a way that preserves the fingerprint integrity even after image alteration.

3.1. *Impersonation investigation taxonomy*

From an investigative standpoint, there are four key points at which an investigation into impersonation schemes can be launched – during the withdrawal of stolen funds, from the website hosting the impersonation pages, from the email source, or from the initial site construction, as shown in Fig. 1. The solid arrows indicate direct connections, while the dotted arrows indicate connections through proxies or other intermediaries. Each connection point is detailed as follows:

Withdrawal point. Withdrawal is the final step in the phishing chain. The traditional law enforcement approach to impersonation focuses on catching perpetrators when they retrieve the money (after the fraud). This is frequently accomplished by tracking the accounts used for money transfers (Cumming, 2004) or through involvement in other criminal enterprises (Pallack, 2005). This approach is becoming less successful as attackers have adapted their methods to avoid directly handling money and have even used e-mules as intermediaries (Peachey, 2005).

Fake site web servers. Tracking the web servers used in impersonation attacks is rarely fruitful. The servers are frequently compromised machines, are transient in nature, and often reside in countries with a poor track record in prosecuting cybercrime (Anti-Phishing Working Group, 2005).

Infected mail servers. Almost all email sources for impersonation attempts come from Botnets – groups of compromised machines that are used exclusively for spam (CMP TechWeb, 2005). Success in this form of tracking relies on the impersonator being careless as opposed to the investigator being clever (Weiss, 2006).

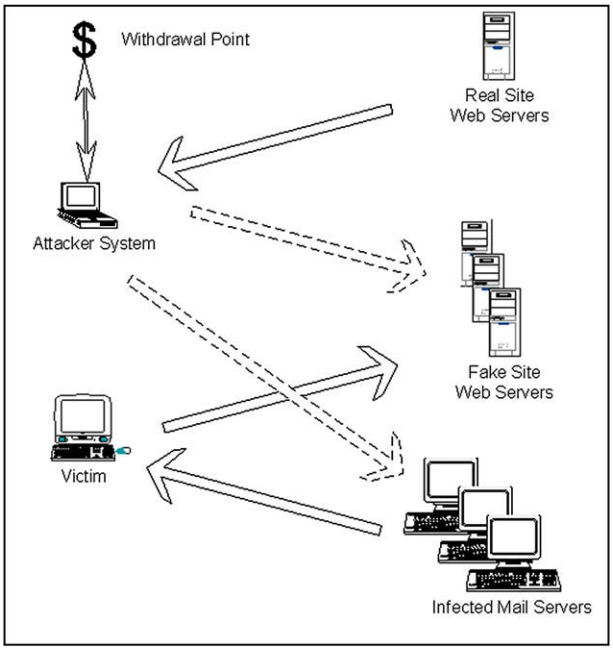

Fig. 1 – Investigation points in a phishing attack.

Real site web servers. AIIIS addresses tracking from the real site's web servers, which has not been used regularly to-date as an investigative technique. Tracking from the originating web server of the images has the advantage of being more likely to come from the actual source of the attacks – web design is more likely to occur on a local machine instead of a distant "owned" machine. Also, there is a lower likelihood of needing to go through multiple hops or proxies to get to a real point of origin instead of a proxy. These images are tracked through an image fingerprinting process.

3.2. *Fingerprinting*

Fingerprinting images involves placing a unique digital watermark into the image content (Cox et al., 1997; Brassil et al., 1994; Soriano et al., 2005). This can be a hidden watermark, or a visible watermark, and is frequently used for purposes of copyright protection. Prior work on fingerprinting can be further divided into two areas – insertion techniques and applications.

3.2.1. *Insertion techniques*

The primary method for inserting fingerprints in AIIIS builds on prior art in the area of spread spectrum insertion. Previous work has covered inserting both static and dynamic contents, but not using the techniques outlined in AIIIS and not for the purpose of impersonation investigation.

Bansal et al. (2003) took a similar approach algorithmically to the techniques described for AIIIS – they dynamically encrypt a unique identifier into images but have the viewer and image itself bundled. They utilized basic techniques for encrypting the data with public key technology proposed by Harn (1991). The decryption is done offline on an as-needed basis (non-blind).

Zheng et al. (2005) applied an approach to video fingerprinting using two transform operations. Gaussian models have been used by Zhao et al. (2005) as well in general multimedia fingerprinting, though hiding the data in the image in AIIIS uses spread-spectrum techniques from Cox et al. (1997) to reduce the impact of image changes.

3.2.2. *Applications*

Practical implementations of fingerprinting to-date have been implemented for copyright protection purposes. The two largest implementations are by the Motion Picture Association of America (MPAA) to detect movie theft (Van Tassel, 2006) and the Recording Industry Association of America (RIAA) to track file sharing (Boutin, 2001).

Recently, the Fraunhofer Institute, the developer of the MP3 format, implemented an anti-piracy tool which hides a unique hash value in downloaded music files. The details provided by the Institute on the project are non-specific as to hash insertion, but resiliency claims point toward the use of a frequency domain technique (Blau, 2006). This implementation would potentially be the most similar to AIIIS, but algorithm details are not provided and the target media are audio content, not images.

The primary goal of all these systems is fighting piracy (as opposed to fraud), and an excellent overview of anti-piracy watermarking is provided in Barni and Bartolini (2004). Additionally, these approaches are static and rely on manual assignment mappings of a fingerprint to an individual. AIIIS uses a dynamic implementation and the mappings need not be stored a priori.

4. AIIIS methodology

AIIIS has two major components: string insertion and string extraction. String insertion occurs in real-time on a standard web server as requests are received. Extraction occurs only after an attacker uses an image from the website. At that point, the image is compared with an original and the details of the download extracted from the image. For insertion, a unique (for each web request) string is embedded into each image downloaded with details on the request itself. The embedding is done using spread-spectrum techniques to prevent signal loss due to ex-post-facto image alteration by the attacker. For extraction, the encrypted string is recovered from the altered image through a comparison to the original image. The string is then decrypted and the source of the original download is identified by matching it with web server logs. Each of the two operations is detailed as follows.

4.1. *String insertion*

The basic methodology for dynamically embedding data into an image is shown in Fig. 2. Insertion consists of two stages – insertion string generation, and image fingerprinting with the encrypted string.

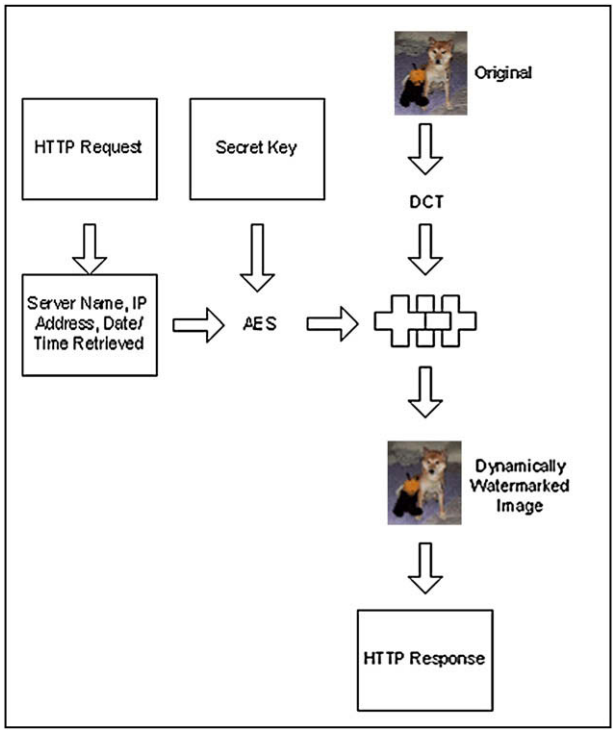

Fig. 2 – Insertion of a dynamic watermark into an image on a web server.

4.1.1. *String generation*

An HTTP request is made from an end user. Ordinarily, an image would be served using a standard HTML IMG tag. For AIIIS, the image tag is replaced with a dynamic call to the encoding program, written in C# using the ASP.Net framework. The image itself is then dynamically generated from an unwatermarked logo.bmp file (a decoder would require potential additional processing) not directly accessible to the requestor and inserted at runtime. Embedded in the bitmap is a watermark that contains an encrypted version of the unique information on the request which allows it to be mapped to a specific web session. The string contains the following information at a minimum:

- A unique identifier for the web server.
- The time and date stamp of the request.
- The IP address the request originated from.

The insertion process is space sensitive, so a minimal string size is desirable. A reasonable string size of 9 bytes (72 bits) can be achieved using 4 bytes each for the time (stored in Unix time format) and IP address, with another byte for the web server, which provides for a server pool of 256 systems.

The web session string information is concatenated and encrypted using the AES or DES algorithms, which has a side benefit of evening out the distribution of the string data. The encryption is performed using a key file specific to the pool of servers that house the image. An alternative approach would be the generation of a random key which linked back to a database containing the server information – this was rejected as it required additional alterations on the web server environment.

4.1.2. *Image fingerprinting*

The Discrete Cosine Transform (DCT) coefficients are altered on the original image based on the individual bit contents of the encrypted insertion string. The string is overinserted, inserted c times into the image, to compensate for bit errors generated during quantization rounding, malicious image alteration, and/or format transitions.

The altered DCT is then transformed back into a bitmap, or alternatively into another image format such as JPEG and sent to the requestor.

The dynamically generated AES or DES encrypted string is hidden in the source image through the use of spread spectrum techniques pioneered by Cox et al. Our approach differs from Cox in that the string data are spread across the entire image instead of being broken up into 8×8 blocks, and the string is inserted multiple times to provide additional resilience to alteration.

Like traditional watermarking techniques, the insertion must be resilient against both intentional alteration and accidental alteration (through cropping, resizing, etc.). Unlike traditional techniques (and closer to the requirements in steganography), the exact AES or DES string must be recoverable. Traditional watermarking techniques generally rely on a similarity function, whereby the likelihood of a similar random watermark is reduced to a statistical improbability. In this particular application, there is a need for exact recovery of the origin string – a probabilistic match isn't enough.

Using the same notation as Cox, the encrypted string is transformed into series of bits (bits were chosen instead of a smaller amount of real numbers for their discrete nature – remember the goal isn't just a probabilistic match), $X = x_1,\ldots,x_n$. To reduce the impact of individual value corruption, the insertion string x is a concatenation of the original encryption string with itself c times.

The original image is stored as a 24-bit RGB image to allow for quick DCT transformation (in practice, the source format is irrelevant as the intermediary formats can be stored directly in memory). First the image is padded to be $N \times N$, where N is the smallest multiple of 2 that is larger than the largest original image dimension. The padded image is then converted to a YCbCr format using a standard transformation equation (Payette, 2002):

$$Y_i = 0.257 * R_i + 0.504 * G_i + 0.098 * B_i;$$

$$Cb_i = -0.148 * R_i - 0.291 * G_i + 0.439 * B_i;$$

$$Cr_i = 0.439 * R_i - 0.368 * G_i - 0.071 * B_i;$$

The luminance (Y) component of the image is used in this technique, though the chrominance (Cb, Cr) components could similarly have been used.

An $N \times N$ DCT is performed on the Y components of the image to generate a list of coefficient values, $V = v_1,\ldots,v_n$. A full transformation is done on the image as a whole to prevent blocking effects generated when 8×8 block coefficients are individually altered, and to increase the survivability of x when cropping occurs. The coefficients from V are then sorted from the highest to the lowest absolute value as $V^* = v_1^*,\ldots,v_n^*$. The transformed coefficients are generated from the list V^* in order to facilitate the ease of future comparison using a scaling function α, set to .1 for the purposes of this system (though testing showed little difference in error rates with other values of α, higher values produced noticeable image degradation):

$$v_i' = v_i^*(1 - \alpha x_i)$$

The scaling is inverted from the typical $1 + \alpha$ to downscale values instead of upscaling (or using of negative α values) – this is done to decrease the range of values generated and to reduce the number of values outside of the 0–255 range when later YCbCr → RGB conversion is performed. Since the largest absolute value coefficient is used, the equation remains invertible as $v_i^* \neq 0$.

Once insertion is done, the image itself is outputted as a standard JPEG image with a 100% (lossless) value. Precompression of the initial bitmap can be done prior to insertion to achieve optimal image size.

4.2. *String recovery*

If an image with embedded data is used at a later date as part of a phishing scheme or trademark infringement case, the reverse of the insertion steps can be performed to determine the point of origin for the image. The string recovery process, as shown in Fig. 3, is the process of extracting a string, X', from an image V'' in an attempt to reconstruct the server name, IP address, and download time. To recover

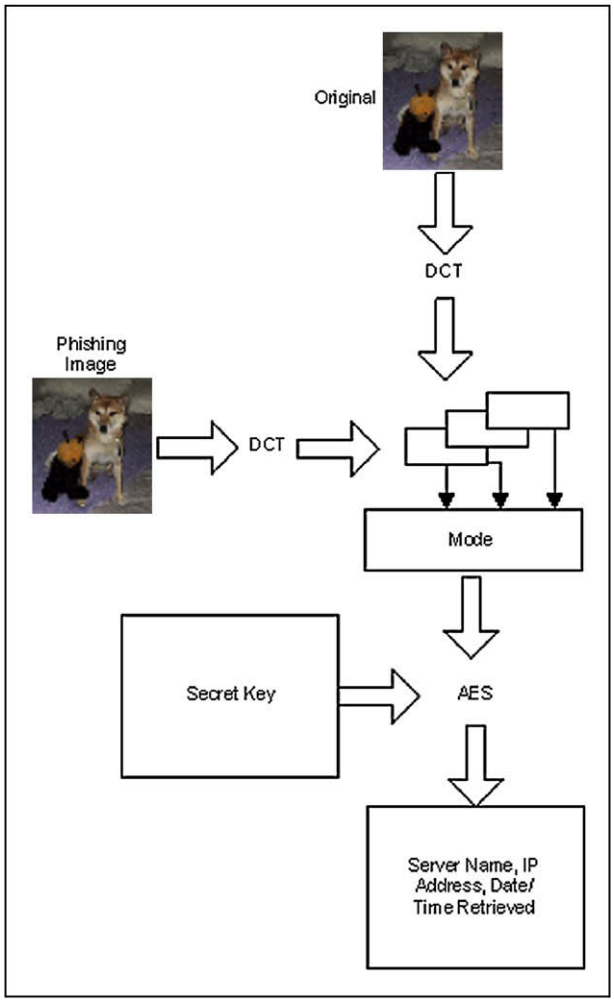

Fig. 3 – Extraction of a dynamic string from a recovered image.

the string, the original image prior to insertion is required, as is the AES or DES encryption key used to encrypt the original string. First the DCT coefficients of the original and altered images are generated in the image preprocessing step. The difference of the two is then compared against a threshold value, and the original AES or DES bits recovered as part of the string extraction. The encrypted string is then decoded and the string can then be directly matched to server logs for further investigation (and for evidence of the original download for legal use).

4.2.1. *Image preprocessing*

V and V'' are both converted to YCbCr format using the same equations in Section 4.1.2 above (V'' is preprocessed to make the images the same size and fill in cropped areas with similar areas from the original image, as well as convert to 24-bit RGB format if needed). The DCT of both images is done to generate V and V'', respectively. The highest absolute value coefficients from the initial image V^* are obtained, then matched to the same coefficient locations in V'', represented as $V^{**} = v_1{}^{**},\ldots,v_n$.

4.2.2. *String extraction*

Each value in the string X' is recovered by comparing the values in V^* and V^{**} as follows:

$$X' = T\left(\frac{1 - \frac{v_i''}{v_i'}}{\alpha}\right)$$

where T is a simple threshold function with the threshold value set to .5 to allow X' to be set to the nearest binary result. Given the binary nature of the data inserted, T operates as

$$X' \leftarrow \begin{cases} 1, & \text{if } > \text{threshold} \\ 0, & \text{if } < \text{threshold} \end{cases}$$

This is performed on all values to the length of the original string X (the procedure assumes a uniform string length for all insertions). To reduce the number of errors present, the c overinserted values in X' are compared with each other and the mode taken of the values, specifically:

$$x_i' = \overline{x}_i'' = \frac{\sum_{n=1}^{c} x_{nc+i}''}{c}$$

If c is an odd number, the same threshold value and function T above can be applied to regenerate the likely original bit (with no danger of bimodality). X can then be decrypted using the known passphrase, and the resulting string is compared to the weblogs to identify the source of the original image download.

4.2.3. *Recovery likelihood*

Using the values above and an overinsertion of $c = 1$, the probability of a single bit being wrong was measured to be around .03 in a series of 128×128 pixel images that were quantized and compressed then recompared. For a 128 bit string, this represents an average of 3.84 erroneous bits with no overinsertion.

To achieve a more reasonable bit error rate, overinsertion of the original string is performed. Since there is no correlation between the coefficients in the location of the overinserted string value and the original location, they are independent for the purposes of the bit error calculation, though total string length does effect the error rates. Overinsertion is used instead of traditional error correcting codes for portability to other insertion methods – specifically methods in the spatial domain which are susceptible to cropping attacks. Ideally, both error correcting codes and overinsertion could be used to reduce error rates.

The number of bits in the bitstream acts as a serial system for the purposes of calculating the likelihood of the complete string being recovered. For this reason, minimizing the length of the string is valid. Specifically, with a probability R of a single bit being good the probability of the string of length m bits being recovered correctly is R^m.

Overinsertion assists in the reliability of string recovery by introducing a K of N parallelism to each bit. Specifically, the probability of a specific bit being accurate can be calculated as

$$\sum_{k=N \text{ DIV } 2+1}^{n} \left(\frac{n!}{k!(n-k)!}\right)(R)^k(1-R)^{n-k}$$

Adding in the serial component, we achieve an overall string recovery likelihood (probability) that can be calculated as

$$\left(\sum_{k=N \text{ DIV } 2+1}^{n}\left(\frac{n!}{k!(n-k)!}\right)(R)^{k}(1-R)^{n-k}\right)^{m}$$

Assuming coefficient independence on an arbitrary string of 128 bits, $R = .95$, and $c = 3$ results in an overall string recovery likelihood of .394. At $c = 5$ the overall recovery likelihood is .862, at $c = 7$ it is .976, and at $c = 9$ it is .996. The recovery likelihood can be tweaked by reducing both the size of the input string and the overinsertion value. By setting $c = 25$ and using a string of 32 bits, a 99% recovery rate can occur with a single bit accuracy as low as .8. The selected tolerance can be adjusted based on the image size and the specific application – a larger c value is warranted if the image is likely to be greatly altered.

5. Experimental results

The AIIIS system was tested by inserting strings into various images, manipulating those images, then attempting to recover the initial strings. First, the AIIIS system itself was tested by varying the insertion string sizes and determining the recovery likelihood given a series of image manipulations. Next, testing the impact of image size and image composition on the recovery likelihood was done. Finally, testing was done against the parameters set forth in Cox to compare the value of overinsertion and the use of full image transforms.

5.1. *Experimental design*

The overall experimental design is shown in Fig. 4. First, to test resiliency to alteration, random strings of 32, 64, and 128 bits were then inserted into the same 128×128 image at overinsertions of 1, 5, 9, and 13 times. The resultant images were then scaled, compressed, and cropped to varying degrees and the likelihood of whole string recovery was provided.

Second, to measure the effect of image size, the same image represented at sizes between 64×64 and 512×512 had the above alterations performed to show the effect of image size on recovery likelihood.

Third, to test the effects of image composition on recovery likelihood, three images of 128×128 had the same string inserted, and the resilience to alteration was measured. A grayscale image, a logo, and a photograph were used as the sample images.

Finally, a 512×512 image size was examined using a production version of AIIIS with an α value of .25. At an α of .25, image quality reduction was sufficient for photo-type images so as not to impair visual appeal. An actual server string comprised of 9 bytes (72 bits) was dynamically inserted in the image using an 8×8 block size with no overinsertion as noted in Cox, and the same insertion was then performed using a 512×512 block size and an overinsertion of 3 to test the AIIIS settings. Both images were subjected to the alterations above and the results provided (Cox et al., 1997).

For the test environment, a copy of AIIIS was installed on a Windows 2003 web server running IIS, with version 2.0 of the .Net framework. The base system was a 1 GHz Pentium 4 with 512 MB of RAM.

5.2. *Image alteration*

The image alteration tests used a single 128×128 photo image with significant color diversity. The AIIIS system was set to a constant α value of .1 to reflect the settings present in Cox, though visual degradation was not significant until higher α values were used. On each experiment, the string size and overinsertion values were the variables, with image size, α value, and image composition remaining constant.

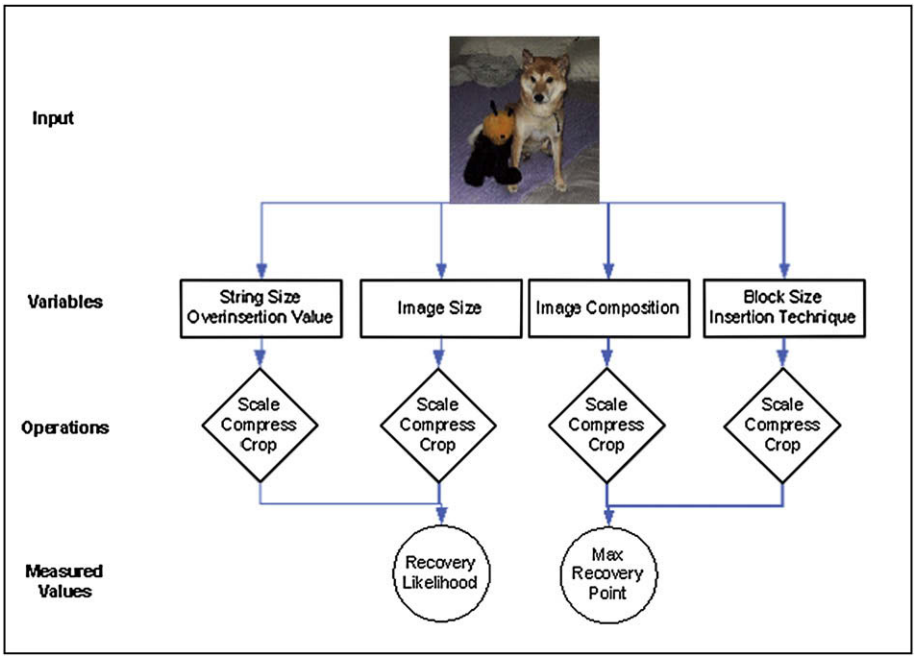

Fig. 4 – Experimental design.

5.2.1. *Scaling*

The rescaling of an image is a common alteration for image reuse in a website or email format. Rescaling consists of changing the image size, with a resultant loss in pixels. To simulate scaling, the output image was rescaled to 80, 60, 40, and 20% of its original size. For recovery, the smaller images were then rescaled to the original 128 × 128 size, without the use of a smoothing algorithm.

Scaling showed greater than 90% recovery at 80% scaling with smaller overinsertion values (1× and 5×) and a smaller string size. The detail present in the high-frequency components was lost due to the scaling algorithm used in Corel Photoshop – specifically, the high-frequency components are removed due to smoothing and only the highest-magnitude low frequency components remain. As a result, the available space for insertion is greatly reduced and the string size needs to be appropriately handled. If scaling is the primary concern, a small insertion string with a minimal overinsertion (3×) offers the best chances for recovery. Specific probabilities of recovery after overinsertion are shown in Fig. 5. Larger overinsertion values resulted in string values which were placed in coefficients that had a higher probability of being removed during smoothing, showing a preference for minimizing overall insertion length.

Though string recovery isn't probable with extreme scaling, a signal is still present. At 20% of its original size, an individual bit probability of accuracy of .625 on a 32 bit string was obtained with no overinsertion – indicating the presence of a signal well above random noise.

5.2.2. *JPEG compression*

For email-based image reuse, further compression is a secondary alteration possibility to reduce image size. In its most common form, the compression is JPEG-based. Testing was performed with compressed versions of the altered image at varying overinsertion levels to determine effect. Compression was performed without smoothing using Photoshop.

By compressing the image with lossy JPEG compression image recovery at up to 40% compression was determined to be probable as indicated in Fig. 6. As with scaling, the amount of data recovered was directly related to the overall string length. As a result, the impact of overinsertion was minimal on the image size and string sizes selected, though its value was not directly determinable for larger string sizes. As with scaling, JPEG compression removes the values of some coefficients through quantization – this was more likely to happen to the smaller coefficients where longer string values were more likely to be placed.

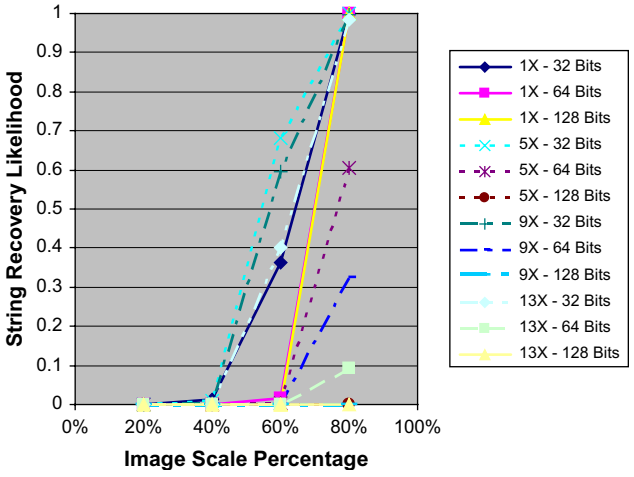

Fig. 5 – Scaled image recovery.

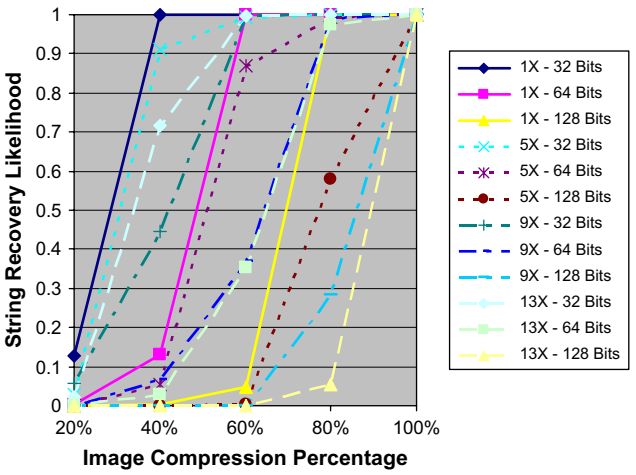

Fig. 6 – Compressed image recovery.

5.2.3. *Cropping*

To test the resilience to cropping, the original image was cropped in 5% increments and cropped portions replaced with the same sections on the original image. Cropping was performed by removing the leftmost portions of the image in Photoshop, then saving the results as bitmaps. With the images used, the leftmost portion contained the most high-frequency components and represented the worst case scenario for cropping this particular image.

The cropped image represents the cutting of an image portion for use in another image – common in trademark theft cases. The cropping results showed promise for overinsertion. With an overinsertion of 13× and a 32 bit insertion string, probability of recovery was over .9, even with 15% of the image removed. The recovery rates for cropped images are directly proportional to the amount of original image remaining and overinsertion amount and inversely proportional to the string size, as shown in Fig. 7.

5.3. *Image size*

In addition to the basic alterations above, similar alterations were made on different sized images ranging from 64 × 64 to 512 × 512 to gauge the impact of image size on the ability to resist alteration. Each size had the same random, 64-bit string inserted with an overinsertion value of 25. As with the alterations, an α of .1 was used. The α values, string size, and overinsertion rates remained constant in this experiment and the image size was the variable. The results are shown in Fig. 8.

The image size highly effected both resizing and rescaling operations – the more bits available to the image, the less likely each operation would result in a reduction in recovery

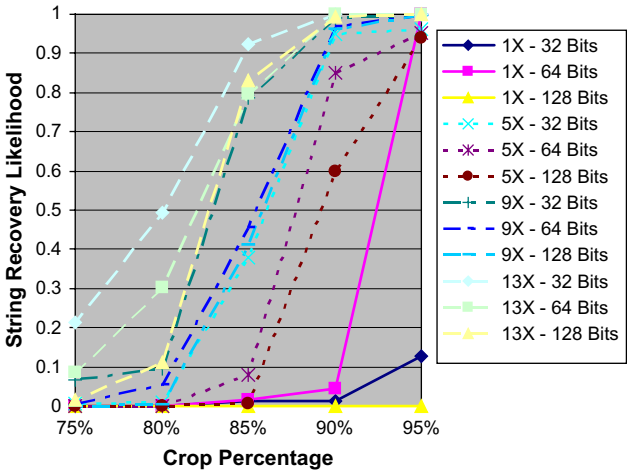

Fig. 7 – Cropped image recovery.

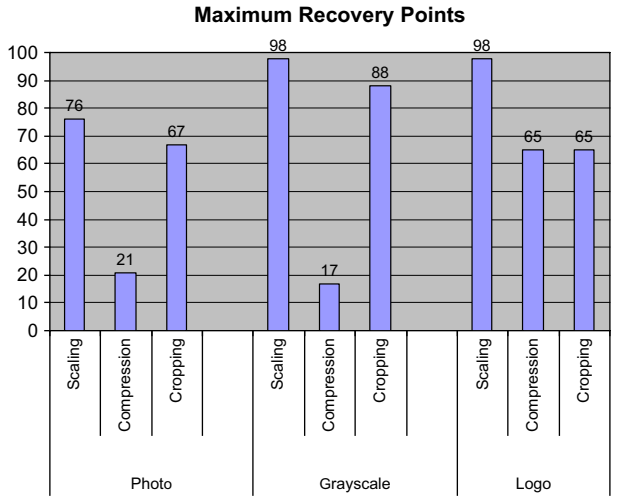

Fig. 9 – Maximum alteration where recovery was still possible.

likelihood. At the largest image size, greater than 95% recovery likelihood was possible with both 50% scaled and 50% compressed images.

Cropping appeared to be largely unaffected by image size. Given a cropped image size of 70%, a low recovery likelihood was shown for all image sizes tested.

5.4. *Image composition*

The type of image used was varied in the fourth test to determine the effect of composition on the recovery likelihood. Using the same experimental setup as the first three experiments, the same string was inserted into three different images of 128 × 128 pixels. The first image was a complex color photograph, the second was the same image in grayscale, and the third a 16 color logo. All of the images were downloaded from the Bank of America website to simulate images from a frequent target of attacks.

Each of the above alterations – scaling, compression, and cropping – was performed on the inserted images. The point at which complete recovery was no longer possible was noted, and is shown in Fig. 9. Smaller numbers reflect a greater chance for fingerprint recovery.

As shown in Fig. 9, the more complex the image, the easier it is to embed data. The photo was able to withstand the most alteration, followed by the grayscale image and finally the logo. The cropping resilience of the logo was noted as higher due to the method used – the majority of the image information present was in the center of the image and cropping was done from the sides to simulate an actual cropping operation by an impersonator.

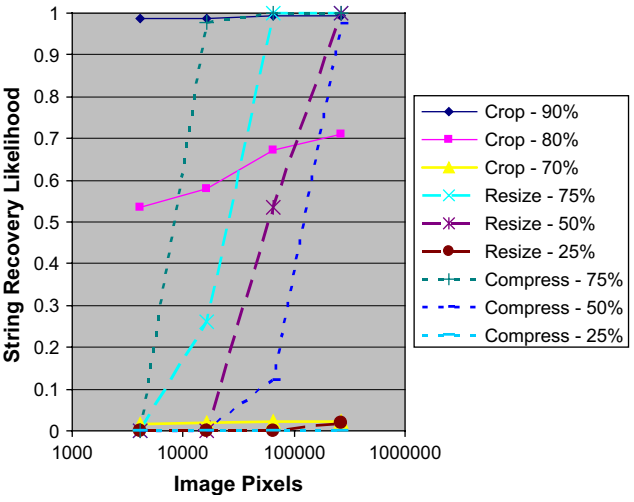

Fig. 8 – Image size effect on recovery likelihood.

5.5. *Comparison to block techniques*

For a head-to-head comparison we ran sample insertions against the base algorithms from Cox and a production version of AIIIS. A full production deployment was used to test the feasibility from a time perspective as well as the recovery likelihood with actual strings.

To perform the test, two different settings were altered. First, block size was set to 8 × 8 for the Cox algorithm and 512 × 512 (the whole image) for the AIIIS algorithm. Second, the Cox algorithm did not use overinsertion while the AIIIS algorithm used an overinsertion value of 3. Both used an α value of .25 (which was visually acceptable on the test images). Images were generated with both algorithms, then each of the above operations, scaling, compression and cropping, was performed and the values at which recovery failed were noted. The results are shown in Fig. 10. An 8 × 8 block comparison of the original and altered images is shown in Fig. 11. The comparison shows no visible change, and a minimal change to the low frequency areas.

Scaling. For scaling, neither performed remarkably well. The string was recoverable in Cox when scaled to 96%, and in AIIIS when scaled to 78% at the largest image size, showing a slight advantage to the AIIIS settings. Scaling ability decreased as image size decreased.

Compression. Performing compression, both algorithms showed very good resilience with Cox allowing compression to 46% of

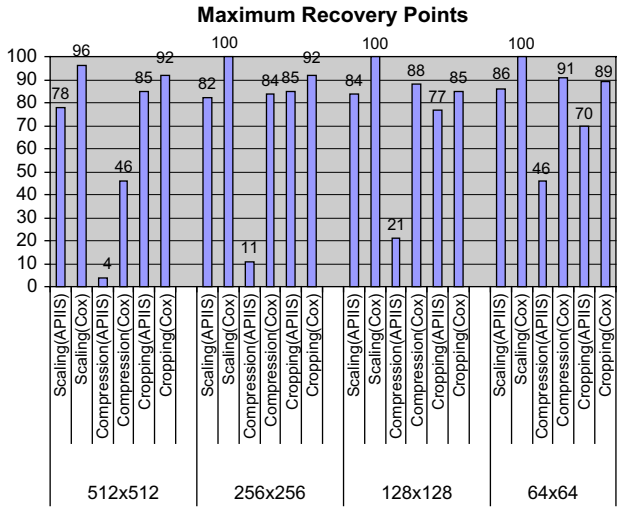

Fig. 10 – Maximum alteration where recovery was still possible.

the original image size and AIIIS to a remarkable 4% of the total image size at 512×512. The results are likely due to the increased α value, producing higher coefficient differentials. As with scaling, the compression ability decreased with image size, to a maximum of 46% with AIIIS and 91% with Cox with a 64×64 image.

Cropping. Cropping yielded similar results to scaling, with AIIIS having a slight advantage. With Cox, the removal of a single key block was enough to make recovery impossible, leading to a maximum cropped size of 98% of the original image at 512×512. With AIIIS, the image could have 85% removed and recovery was still possible. Surprisingly, a larger percentage of the image could be successfully cropped as image size decreased, resulting in 70% of the image remaining with AIIIS and 89% with Cox on a successful recovery from a 64×64 pixel image, indicating that cropping resilience is due to an absolute loss in pixels rather than a relative percentage of the image.

Though the results are positive for AIIIS, the actual dynamic image generation at 512×512 was too slow to be practical *on current machines* – taking 2 min to perform. Cox performed the same image generation in well under 1 s with an unoptimized 8×8 block size. The 256×256, 128×128, and 64×64 pixel images were adequately rapid under both methods.

6. Discussion

The capability of both altered and unaltered recovery of the images was determined to be significant through basic testing. Recovery is highly probable for unaltered images as long as there are sufficient image coefficients to support the length of the inserted bit string. The maximum coefficients available are Length $\times$ Width $-$ 1 (DC coefficients) in the proposed method (if the Cb and Cr coefficients are used this is increased threefold). Therefore, an 8×8 image would not support insertion of more than 63 bits – since recovery is binary (either the string is recovered or it isn't because of the encryption). The recovery likelihood scales linearly with image size, corresponding to a linear increase in the number of available insertion coefficients.

The results obtained are in-line with those identified in Cox et al. (1997), though the goal of recovery was different – Cox required showing a signal that was significantly more probable than random as opposed to all-or-nothing string recovery. Because of this, direct comparison to Cox's signal strength results isn't possible, but using the same settings as noted in Cox, AIIIS showed a higher resilience to alteration, due to the non-localization of values to a specific block in AIIIS. The results are significant enough for implementation in practice with the appropriate application-specific tweaking. The use of the entire image for insertion showed a larger tolerance for high α values due to the lack of a blocking effect, and overinsertion provided extra promise against cropping-based alterations (though it showed to be outweighed by overall string size in both scaling and compression operations).

The dynamic embedding techniques presented are subject to the same attacks as most DCT-based watermarking for the jamming of the inserted signal, but the intentional alteration of the signal to correspond to an accurate, reversible AES or DES string that matches a web log entry is minimal. Minimization of the

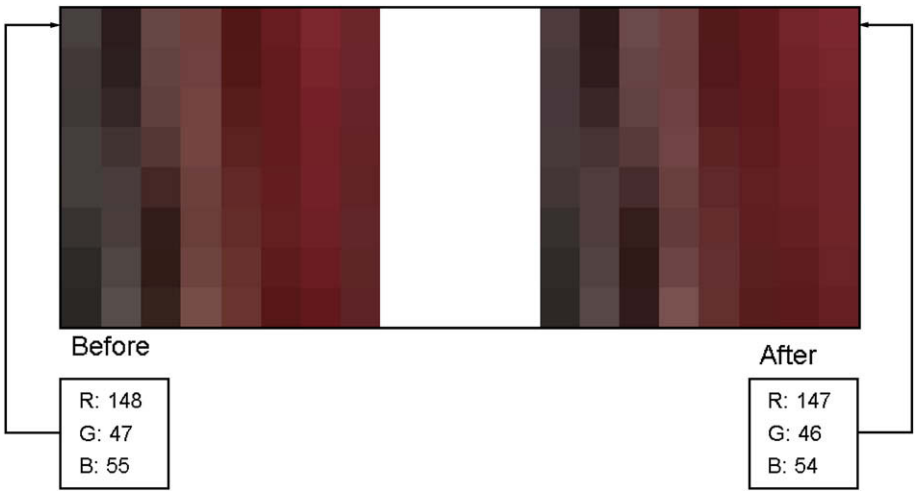

Fig. 11 – Visual differences in an 8×8 section – before and after insertion.

impact from multi-image attacks can be achieved by random insertion of values to other string images if such an attack is expected.

The same dynamic techniques can be applied to Least Significant Bit (LSB) based alterations as well, without the requisite resilience. Depending on the attack scenarios expected, alteration of the dynamically watermarked images may not be a high probability, and the speed of LSB-based methods over transforms provide a viable alternative for production implementations. Additionally, the use of smaller block sizes (e.g., 8×8 or 16×16) allows for optimized implementations of the DCT, which can provide major improvements in large image insertion with a requisite tradeoff in recovery likelihood.

Because string length is a key factor in recovery likelihood, smaller insertion strings can be utilized with some basic tradeoffs. The string size can be compressed by eliminating the server byte and using a different encryption key for each server, though this requires trying each individual key in a recovery. Additional savings can be achieved by changing the time resolution and starting point, since historical times before the current date are not relevant to this application, and individual requests mapped within a minute's resolution may be applicable to lower traffic servers.

Optimization of the code for production use, specifically speed enhancements to the DCT and inverse DCT operations, will need to occur for high volume usage. Pregeneration and caching of the original (before insertion) coefficient array would provide further improvement.

7. Conclusions

The presented results show that AIIIS permits recovery even after post-acquisition manipulation of the image, providing another tool in anti-phishing efforts. The spread spectrum dynamic insertion techniques outlined are feasible for tracking of image origin when unaltered images or further-compressed images are recovered in attempted phishing schemes. Resilience to rescaling, compression, and cropping shows promise with the addition of error correcting code and/or the tweaking of the insertion algorithms.

Future work needs to be done to determine the optimal application-specific insertion algorithm. In addition to the applicability to general web-based phishing investigations, the AIIIS system is directly applicable to both trademark and Digital Rights Management (DRM) protection as well. Applications of the technique could be used by a music or movie download service, for example, to uniquely watermark each individual download dynamically with an encrypted version of the downloader's information – if that watermark later appeared on a peer-to-peer client, the source could be identified.

REFERENCES

Anti-Phishing Working Group. Phishing activity trends report. Anti-Phishing Working Group; July 2005.

AOL/NCSA. Online safety study. America Online/National Cyber Security Alliance; December 2005.

Bansal M, Yan WQ, Kankanhalli MS. Dynamic watermarking of images. In: Proceedings of the 2003 joint conference of the fourth international conference on information, communications and signal processing and the fourth pacific rim conference on multimedia, vol. 2; 2003. p. 965–9.

Barni M, Bartolini F. Data hiding for fighting piracy. IEEE Signal Processing Magazine 2004;21(2):28–39.

Blau J. MP3 inventor develops tool to fight piracy. PC World 2006;2:1.

Boutin P. The RIAA's low watermark. Wired 2001;9:15–7.

Brassil J, Low S, Maxemchuk N, O'Gorman L. Electronic marking and identification techniques to discourage document copying. Proceedings of IEEE INFOCOM '94. 1994;3:1278–87.

CMP TechWeb. Dutch police crush big botnet, arrest trio. CMP TechWeb 10 October 2005:1.

Cox IJ, Kilian J, Leighton FT, Shamoon T. Secure spread spectrum watermarking for multimedia. IEEE Transactions on Image Processing 1997;6(12):1673–87.

Cumming J. Arrest in city after GBP 3m bank scam. Evening News – Scotland 24 November 2004:1.

Dhamija R, Tygar JD. The battle against phishing: dynamic security skins. In: Proceedings of the 2005 symposium on usable privacy and security; 2005. p. 77–87.

Geer D. Security technologies go phishing. IEEE Computer 2005;38: 18–21.

Goth G. Phishing attacks rising, but dollar losses down. Security & Privacy Magazine, IEEE 2005;3(1):8.

Harn L. A public-key based dynamic password scheme. In: Proceedings of the symposium on applied computing; 1991. p. 430–5.

Kirda E, Kruegel C. Protecting users against phishing attacks with AntiPhish. In: Proceedings of 29th annual IEEE conference on computer software and applications; 2005. p. 517–24.

Millersmiles.co.uk. Recent phishing scams. Available from: http://www.millersmiles.co.uk/; 15 March 2006.

National Institute of Standards and Technology, Data encryption standard. NIST. Federal Information Processing Standard 46-2; 1993.

National Institute of Standards and Technology, advanced encryption standard. NIST. Federal Information Processing Standard 197; 2001.

Netcraft Inc.. Netcraft anti-phishing toolbar. Available from: http://toolbar.netcraft.com/; January 5, 2006.

Pallack B. 2 held in 'phishing' ID-theft scheme. The Arizona Daily Star 2005;B1.

Payette B. Color space converter: R′G′B′ to Y′CrCb. vol. XAPP637 (V1.0): Xilinx Aplication Note; 2002. p. 2–3.

Peachey K. Internet mafia made me an e-mule. Nottingham Evening Post 10 May 2005:6.

Soriano M, Fernandez M, Cotrina J. Fingerprinting schemes. Identifying the guilty sources using side information. Lecture notes in computer science, vol. 3710; 2005. p. 231–43.

Topkara M, Kamra A, Atallah MJ, Nita-Rotaru C. ViWiD: Visible watermarking based defense against phishing. In: Lecture notes in computer science; 2005. p. 470–83.

Van Tassel J. High-tech sleuths fight pirates. Available from: http://www.digimarc.com/media/news_comm_link.asp?newsID=400; January 5, 2006.

Wang H, Wang S. Cyber warfare: steganography vs. steganalysis. Communications of the ACM 2004;47(10):76–82.

Weiss TR. Feds shut down PayPal, AOL scams. Available from: http://www.pcworld.com/article/id,115329-page,1/article.html; January 23, 2006.

Zhao HV, Min W, Wang ZJ, Liu KJR. Forensic analysis of nonlinear collusion attacks for multimedia fingerprinting. IEEE Transactions on Image Processing 2005;14(5):646–61.

Zheng L, Xue L, Zhaoyang D. Direct fingerprinting on multicasting compressed video. In: Proceedings of the 11th international multimedia modelling conference; 2005. p. 76–83.