SECURE EMBEDDING OF SPREAD SPECTRUM WATERMARKS USING LOOK-UP-TABLES

Mehmet Utku Celik, Aweke Negash Lemma, Stefan Katzenbeisser and Michiel van der Veen

Information and System Security Group
Philips Research Europe
Eindhoven, The Netherlands 5656 AE

ABSTRACT

In an electronic content distribution system, it is preferable to embed forensic tracking watermarks at the client-side to limit bandwidth usage and server complexity. Embedding in these untrusted clients, however, requires secure embedding methods that do not leak unmarked contents or the watermarking secrets. In this work, we propose a look-up-table (LUT) based cipher, similar to Anderson's Chameleon cipher, for securely embedding spread-spectrum watermarks, which are noise robust and detectable without the original content. We also develop fast detection mechanisms that make the watermark detection feasible for tracking systems with large number of clients. Our fast detection algorithm improves detection speed six orders of magnitude in a typical system.

Index Terms— Forensic tracking, fingerprinting, copyright protection, stream-cipher

1. INTRODUCTION

In the last decade, we have experienced a clear trend toward electronic distribution of audio-visual content. The threat of copyright infringement, however, continues to be a significant problem for record companies and movie studios. An effective deterrent against illegal re-distribution of copyrighted content is forensic tracking watermarking (a.k.a. fingerprinting), where each distributed copy of the content carries a unique watermark (fingerprint) that links that particular copy to the consumer who receives it. Upon discovery of an unauthorized copy (e.g. on a peer-to-peer network), the watermark present in the copy is probed to reveal the identity of the consumer.

Today, electronic content distribution systems embed forensic tracking watermarks primarily at the distribution server. Uniqueness of each copy, however, prevents the use of network-level bandwidth saving mechanisms such as multi-casting and caching. The bandwidth requirement rapidly becomes prohibitive for mass-scale distribution. Therefore, it is preferable to embed the watermark at the client—not only to limit the bandwidth usage, but also to reduce server complexity and to enhance scalability. A single unmarked master copy is sent to all clients in encrypted form eliminating the bandwidth bottleneck. Each client device embeds its unique watermark to the master content. Embedding in these untrusted clients, however, require *secure embedding* methods that do not leak unmarked contents or the watermarking secrets.

In *secure embedding*, the content is sent to the client in encrypted form along with a client-specific decryption key. The decryption process and watermarking process are securely intertwined so that decryption results in a different—uniquely watermarked—content copy personalized for the client. Security of the embedding is based on the design of the particular encryption and decryption methods. In literature, basically four different approaches for secure embedding can be found. In [1], Emmanuel et al. encrypt each video frame by masking it with a noise sequence. Decrypting the video frame using a unique decryption key including a watermark automatically superimposes the corresponding watermark on the content. The scheme achieves perfect security if a new masking sequence is generated for each frame, at the cost of transmitting a very long decryption key. Another approach, called stream switching, divides the content stream into various segments (e.g. chapters in a movie) and prepares multiple versions of each segment by embedding a different watermark and encrypting with a different key [2, 3]. Each client is given a unique series of decryption keys that enables him to decrypt one version of each segment. The selection of decryption key series encodes the forensic tracking payload in the whole content. The scheme is susceptible to collusion attacks, where a group of attackers can implicate an innocent client by pooling together their decryption keys. Collusion-secure codes [4] can counter the attack, but they require large number of content segments. In [5], Kundur et al. encrypt signs of all significant (mid-frequency) DCT coefficients in an image and give each user necessary keys to decrypt only a subset of these coefficients. The coefficients that remain scrambled form a forensic mark (fingerprint). While the scheme is efficient in its partial encryption of compressed data, the lack of efficient detection mechanisms and collusion security measures severely limit its potential use.

In [6], Anderson and Manifavas present a stream-cipher that supports the use of multiple decryption keys, which decrypt the same cipher-text to slightly different (watermarked) plain-texts. During encryption, a short-term key and a secure index generator are used to generate a sequence of indices, which are used to select entries from a look-up-table (LUT). Four selected entries are XORed together with the plaintext to form a word of the ciphertext. The decryption process is identical to encryption except for the use of a decryption LUT, which is obtained by injecting bit "errors" in some entries of the encryption LUT. Decryption superimposes these "errors" onto the content, thus leaves a unique fingerprint. Recently, Adelbach et al. [7] proposed a generalization of the Chameleon cipher, called *Fingercasting*. In their scheme, the LUT entries are randomly chosen elements from $\mathbb{Z}_p$ and the XOR operation is replaced by a modular addition in $\mathbb{Z}_p$. As a result, the scheme can embed spread-spectrum watermarks. Authors provide rigorous security proofs for the confidentiality of the resulting cipher.

In this paper, we utilize a secure embedding mechanism which is similar to the Chameleon [6] and Fingercasting [7]. Instead of XOR or modular operations, we use regular additions on real numbers. As a result, our watermark can be detected even after subsequent noise addition or lossy compression (unlike Chameleon) and in the absence of the original content (unlike Fingercasting) (Sec. 2). We further observe the high computational complexity of the detection of [7] and develop alternate detection mechanisms, which are up to six orders of magnitude faster (Sec. 3).

2. SECURE EMBEDDING USING LUTS

Our secure embedding mechanism for spread-spectrum watermarks is based on an encryption framework which is similar to the Chameleon cipher [6]. The server encrypts the content by adding a pseudo-random blinding sequence which is obtained by selecting entries from a long-term encryption look-up table (LUT) based on a secure pseudo-random sequence generator. Each client decrypts the content using his/her personalized decryption LUT, which is a superimposition of the encryption LUT and a personalized watermark LUT. (See Fig. 1). As a result, the decryption process effectively superimposes a spread-spectrum watermark sequence onto the decrypted content. The personalized nature of the watermark sequence makes it possible to trace the decrypted content back to an individual client. In the following we describe the key generation, encryption and decryption operations in greater detail.

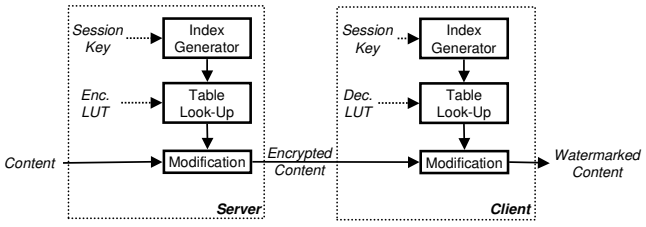

Fig. 1. Encryption and corresponding joint decryption and watermarking procedures.

2.1. Key generation

The server constructs a long-term encryption key $\mathbf{E}$, which has the form of a LUT of size L. The entries of the table, $\mathbf{E}[i]$, are chosen independently and randomly according to a Gaussian probability distribution ($\mathbf{E}[i] \sim \mathcal{N}(0, \sigma_E)$). The table $\mathbf{E}$ is considered to be the master encryption key and it is common for all N clients. For each client $k \in \{0, 1, \ldots, N-1\}$, the server chooses a personalized watermark LUT $\mathbf{W}_k$ of size L. The entries of $\mathbf{W}_k$ are chosen randomly according to a desired probability distribution ($\mathbf{W}_k[i] \sim \mathcal{N}(0, \sigma_W)$ with $\sigma_W < \sigma_E$). The server constructs a personalized decryption LUT, denoted by $\mathbf{D}_k$ by computing

$$\mathbf{D}_k[i] = -\mathbf{E}[i] + \mathbf{W}_k[i], \tag{1}$$

for $0 \leq i \leq L-1$. Finally, $\mathbf{D}_k$ is transmitted to each client over a secure channel. Note that both $\mathbf{E}$ and $\mathbf{D}_k$ are long-term encryption/decryption keys, which do not need to be updated by each and every transaction. In addition to this long-term encryption key, the server transmits to authorized users a session key K which is unique for each content.

2.2. Encryption

We denote the original content by a vector $\mathbf{x}$ of length M with elements $x_0, x_1, \ldots, x_{M-1} \in \mathbb{R}$. To encrypt content, the server first uses an index generator (which can be a pseudo-random number generator or a stream cipher operating in OFB mode) and the session key K to generate indices t_{ij} to its master lookup table, where $0 \leq i \leq M-1$, $0 \leq j \leq S-1$ and $0 \leq t_{ij} \leq L-1$. Finally, the server distorts each feature x_i of the content by adding S entries of the lookup table, thereby yielding the encrypted content $\mathbf{c}$, where

$$c_i = x_i + \sum_{j=0}^{S-1} \mathbf{E}[t_{ij}]. \tag{2}$$

2.3. Decryption & Watermarking

To decrypt the content, the client uses the index generator and its session key K to reconstruct the indices t_{ij} used by the server in the encryption step. By inverting the encryption process with its own decryption table, it can obtain watermarked content $\mathbf{y}$:

$$\begin{aligned} y_i &= c_i + \sum_{j=0}^{S-1} \mathbf{D}[t_{ij}] \\ &= x_i + \sum_{j=0}^{S-1} \mathbf{W}[t_{ij}]. \end{aligned} \tag{3}$$

3. WATERMARK DETECTION

Forensic tracking watermark detection may be performed off-line, possibly in the presence of abundant computational resources and time. In many practical applications, however, detection fast and low-cost detection is crucial. In this section, first we present typical system parameters, which we later use to compare the complexity of different detection algorithms.

3.1. Parameters for a typical forensic tracking system

In our hypothetical forensic tracking system for video distribution, we assume a spread-spectrum watermark embedded into one coefficient (e.g. DC) in every 8×8 DCT block of a video sequence. In standard-definition video (e.g. PAL 576×720 pixels @50 fields/sec.), there are approximately $160,000$ 8×8 blocks per second. Assuming detection over a 2 minute video clip, the total number of watermarked coefficients is close to 20 million ($M = 2.0 \times 10^7$). The LUTs in our system have 1 million entries ($L = 1.0 \times 10^6$) and four look-ups are performed for each coefficient ($S = 4$). We further assume 100 million clients in our distribution area ($N = 1.0 \times 10^8$).

3.2. Basic detection method

During watermark detection, the server uses the session key K and the index generator to reconstruct the indices t_{ij} used in the encryption step. Using watermark LUT $\mathbf{W}_k$ for each client, it computes a watermark sequence $\mathbf{w}_k = w_{k,0}, w_{k,1}, \ldots, w_{k,M-1}$ by

Table 1. Watermark detection complexity, **A)** when watermark sequence is correlated with the estimated sequence, **B)** when watermark LUT is correlated with the estimated LUT, **C)** when circular correlations are performed using FFTs ($\Gamma = 2\lceil N/L \rceil + 1$).

		One-time ops				Per client ops				Total ops			
		Idx	L-up	Add	Mult	Idx	L-up	Add	Mult	Idx	L-up	Add	Mult
A	Construct watermark	-	-	-	-	SM	SM	$SM - M$	-	SMN	SMN	$SMN - MN$	-
A	Correlate watermark	-	-	-	-	-	-	M	M	-	-	MN	MN
A	Total	-	-	-	-	SM	SM	SM	M	SMN	SMN	SMN	MN
B	Construct LUT	SM	SM	SM	-	-	-	-	-	SM	SM	SM	-
B	Correlate LUT	-	-	-	-	-	-	L	L	-	-	LN	LN
B	Total	SM	SM	SM	-	-	-	L	L	SM	SM	$SM + LN$	LN
C	Construct LUT	SM	SM	SM	-	-	-	-	-	SM	SM	SM	-
C	Correlate LUT	-	-	$\Gamma L \log L$	$\Gamma L \log L$	-	-	-	-	-	-	$\Gamma L \log L$	$\Gamma L \log L$
C	Total	SM	SM	$SM + \Gamma L \log L$	$\Gamma L \log L$	-	-	-	-	SM	SM	$SM + \Gamma L \log L$	$\Gamma L \log L$

Table 2. Detection complexity for different detection methods ($S = 4$, $N = 1.0 \times 10^8$, $M = 2.0 \times 10^7$, $L = 1.0 \times 10^6$).

Detection Method	Idx	L-up	Add	Mult	Total
Sequence correlation	8.0×10^{15}	8.0×10^{15}	8.0×10^{15}	2.0×10^{15}	2.6×10^{16}
LUT correlation	8.0×10^{7}	8.0×10^{7}	$\sim 1.0 \times 10^{14}$	1.0×10^{14}	$\sim 2.0 \times 10^{14}$
Circular LUT correlation	8.0×10^{7}	8.0×10^{7}	$\sim 4.1 \times 10^{9}$	$\sim 4.0 \times 10^{9}$	$\sim 8.3 \times 10^{9}$

$$w_{k,i} = \sum_{j=0}^{S-1} \mathbf{W}_k[t_{ij}]. \tag{4}$$

The server correlates $\mathbf{w}_k$ with an estimated watermark $\mathbf{z}$ obtained from the "suspect" (possibly watermarked) content. The correlation value is compared with a suitably chosen threshold to decide if the watermark for that particular client is present in the suspect content (Fig. 2). The process is repeated either for all clients or until a positive match is found. We evaluate the complexity of the detection

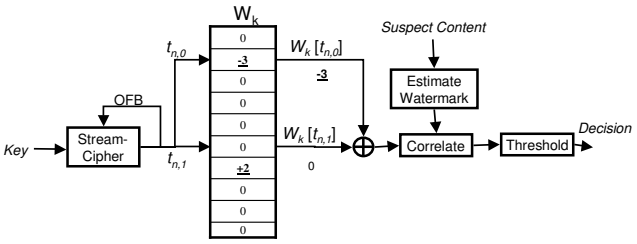

Fig. 2. Detection procedure ($S = 2$) in which watermark sequence is reconstructed and correlated with the watermark estimate derived from the received signal.

process in terms of four basic operations: index generation (*Idx*), table look-up (*L-up*), addition (*Add*), and multiplication (*Mult*). Assuming detection is repeated for all clients, the detection complexity increases linearly with the number of coefficients in the content and the number of clients (**A** in Table. 1). When above parameters are used, the detection process requires roughly 2.6×10^{16} operations (Table 2), which would take more than 300 days on a machine capable of 1G ops/sec (billion operations per second). This complexity is prohibitive in many practical applications and faster methods are necessary.

3.3. An alternate detection strategy: Correlating LUTs

When we examine the watermark detection process, we observe that the correlation value is a linear combination of elements in the watermark sequence, which in turn are linear combinations of watermark LUT entries. That is,

$$\begin{aligned} \langle \mathbf{z}, \mathbf{w} \rangle &= \textstyle\sum_{i=0}^{M-1} z_i w_i \\ &= \textstyle\sum_{i=0}^{M-1} z_i \left(\sum_{j=0}^{S-1} \mathbf{W}[t_{ij}] \right) \\ &= \textstyle\sum_{i=0}^{M-1} \sum_{j=0}^{S-1} z_i \mathbf{W}[t_{ij}]. \end{aligned} \tag{5}$$

We further observe that obtaining the watermark sequence $\mathbf{w}$ from the watermark LUT $\mathbf{W}$ for each client creates significant complexity. Fortunately, the linearity of the correlation allows us to replace this process, which needs to be done for each client, with a process performed once on the suspected content.

In this new detection algorithm, first we construct a suspect LUT $\mathbf{Z}$ from the sequence of watermark estimates $\mathbf{z}$, which are obtained from the suspect content. To construct $\mathbf{Z}$, we start with a LUT of size L filled with zeros. Subsequently, we reproduce the index sequence t_{ij} and for each $0 \le i < M$, we add the element z_i to all positions t_{ij} of the LUT $\mathbf{Z}$ with $0 \le j < S$. Once all elements of the sequence are processed, the resulting LUT $\mathbf{Z}$ is correlated with the LUT $\mathbf{W}_k$ of each client to obtain the detection result $\langle \mathbf{Z}, \mathbf{W}_k \rangle$ (Fig. 3). Correlating the LUTs is equivalent to correlating the watermark sequence $\mathbf{w}$ with the suspected content $\mathbf{z}$, as by Eqn. 5 we have

$$\begin{aligned} \langle \mathbf{Z}, \mathbf{W} \rangle &= \textstyle\sum_{l=0}^{L-1} \mathbf{Z}[l] \mathbf{W}[l] \\ &= \textstyle\sum_{l=0}^{L-1} \left(\sum_{i,j|t_{i,j}=l} z_i \right) \mathbf{W}[l] \\ &= \textstyle\sum_{l=0}^{L-1} \sum_{i,j|t_{i,j}=l} z_i \mathbf{W}[t_{ij}] \\ &= \textstyle\sum_{i=0}^{M-1} \sum_{j=0}^{S-1} z_i \mathbf{W}[t_{ij}]. \end{aligned} \tag{6}$$

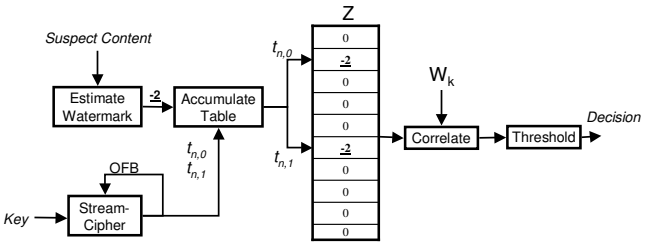

Fig. 3. Alternate detection procedure. Watermark estimates are accumulated in an empty LUT at positions indicated by t_{ij}. This LUT is then correlated with the watermark LUT for detection.

Note that the LUT construction step of the new algorithm is performed only once, considerably speeding up the detection (see row **B** in Table. 1). Comparing rows **A** and **B**, we further see that complexity of the new correlation step is proportional to the LUT size (L) instead of content size (M). While L and M are system parameters that can be chosen freely, in all cases it is preferable to choose $L \ll M$ to limit the bandwidth overhead due to LUT transmission. For the typical parameters above, the alternate detection method is more than two orders of magnitude faster than the basic detection method. However, detection still takes about 3 days on a single 1G ops/sec machine (Table 2).

3.4. An alternate LUT selection strategy: Circular Shifts

So far, we have assumed that the watermark LUTs for different clients are selected independently. While this approach is favorable in terms of security, as it creates the most uncertainty for attackers that may access LUTs from multiple clients, it requires a separate correlation step for each client thereby considerably slowing the detection. We propose an alternate LUT selection mechanism in order to speed up detection. Particularly, each client LUT is obtained by circularly shifting a reference LUT ($\mathbf{R}$) by a fixed offset, i.e. $\mathbf{W}_k = \mathit{CircShift}(\mathbf{R}, k)$. The correlation values between two sequences for all possible circular shifts can be computed efficiently in a single step using the Fast Fourier Transform (FFT). In our case, we use this property to compute the correlation between the suspect LUT $\mathbf{Z}$ and all client LUTs, which are circularly shifted versions of the reference LUT $\mathbf{R}$. Let $\mathbf{\Phi}$ be

$$\mathbf{\Phi} = \mathcal{IFFT}\left(\mathcal{FFT}(\mathbf{R}).* conj(\mathcal{FFT}(\mathbf{Z}))\right), \qquad (7)$$

where $.*$ is the element-wise multiplication, $conj$ is the complex conjugate and $\mathcal{FFT}, \mathcal{IFFT}$ are forward and inverse FFTs. Each element of $\mathbf{\Phi}$ corresponds to the correlation between $\mathbf{Z}$ and a circularly shifted version of $\mathbf{R}$, where the index of each element in $\mathbf{\Phi}$ indicates the shift amount:

$$\begin{aligned} \mathbf{\Phi}[k] &= \langle \mathit{CircShift}(\mathbf{R}, k), \mathbf{Z} \rangle \qquad (8) \\ &= \langle \mathbf{W}_k, \mathbf{Z} \rangle . \qquad (9) \end{aligned}$$

Thus, L correlations can be computed simultaneously with only three FFT operations, each real FFT taking approximately $L \log L$ real additions and $L \log L$ real multiplications. The complexity of the correlation process reduces from L^2, as required by regular correlations, down to $3L \log L$ additions and multiplications.

As the reader may have noticed, the watermark construction proposed above only supports a maximum of L possible shifts, thus L unique watermark LUTs or clients. However, in a typical system (as the example system given above) the number of clients N far exceeds the LUT size L. In this case, we divide the users in $\Lambda = \frac{N}{L}$ groups and assign each group an independent reference LUT, $\mathbf{R}_\lambda$, where $\lambda \in \{0, 1, \ldots, \Lambda - 1\}$. The watermark LUT for a client will then be $\mathbf{W}_k = \mathit{CircShift}(\mathbf{R}_{\lfloor k/L \rfloor}, (k - k \lfloor k/L \rfloor))$. When multiple reference patterns are used for LUT selection, the suspected content has to be correlated with every reference LUT $\mathbf{R}_\lambda$. Each computation requires three real L-point FFT operations. As $\mathcal{FFT}(\mathbf{Z})$ only needs to be performed once, correlation with Λ reference patterns requires $\Gamma = 2\Lambda + 1$ FFTs (**C** in Table. 1). For the typical parameters above, we require hundred reference patterns ($\Lambda = 100$). Corresponding detection takes less than 9 seconds on a single 1G ops/sec machine (last row of Table 2). This is six orders of magnitude smaller than the basic detection system.

4. CONCLUSIONS

In this paper, we have proposed a method for securely embedding spread-spectrum watermarks on the client-side in an electronic content distribution system. We have shown that the watermark detection process can be sped-up by six orders of magnitude for a typical system, if the watermark LUTs for different clients are derived from a limited set of reference LUTs and correlation is performed directly on LUTs. Currently, we are trying to quantify the security of the system against various attack models, including collusion attacks where correlated watermark LUTs may adversely affect system security.

5. REFERENCES

[1] S. Emmanuel and M. Kankanhalli, "Copyright protection for MPEG-2 compressed broadcast video," in *ICME 2001. IEEE Int. Conf. on Multimedia and Expo.*, 2001, pp. 206–209.

[2] R. Parviainen and P. Parnes, "Large scale distributed watermarking of multicast media through encryption," in *Proc. of IFIP TC6 and TC11*, 2001, pp. 149–158.

[3] H. Jin and J. Lotspiech, "Attacks and forensic analysis for multimedia content protection," in *Proc. of ICME. IEEE Int. Conf. on Multimedia and Expo*, 2005.

[4] D. Boneh and J. Shaw, "Collusion-secure fingerprinting for digital data," *IEEE Trans. Info. Theory*, vol. 44, no. 5, pp. 1897–1905, Sept. 1998.

[5] D. Kundur and K. Karthik, "Video fingerprinting and encryption principles for digital rights management," *Proceedings of the IEEE*, vol. 92, no. 6, pp. 918–932, 2004.

[6] R. J. Anderson and C. Manifavas, "Chameleon - a new kind of stream cipher," in *FSE '97: Proc. of the 4th Int. Workshop on Fast Software Encryption*. London, UK: Springer-Verlag, 1997, pp. 107–113.

[7] A. Adelsbach, U. Huber, and A.-R. Sadeghi, "Fingercasting—joint fingerprinting and decryption of broadcast messages," in *11th Australasian Conf. on Info. Security and Privacy*, 2006.