SECURITY AND EFFICIENCY ANALYSIS OF PROGRESSIVE AUDIO SCRAMBLING IN COMPRESSED DOMAIN

Jiantao Zhou and Oscar C. Au

Department of Electrical and Computer Engineering
Hong Kong University of Science and Technology
Email: \{eejzhou, ecau\}@ust.hk

ABSTRACT

In this paper, we address the security and efficiency issues of two recently proposed audio scrambling schemes. We show that these two audio scrambling schemes are actually vulnerable against various attacks such as ciphertext-only attack, known-plaintext attack and chosen-plaintext attack. We also demonstrate that one of these two schemes is lack of efficiency in terms of generating the key stream using the dynamic password generator (DPG). Furthermore, we briefly discuss the ways to improve the security and efficiency of these two audio scrambling schemes.

Index Terms—Audio scrambling, security analysis

1. INTRODUCTION

With the advancement of the multimedia technologies and the increased availability of multimedia data, the need for secure transmission of multimedia signal has received significant attention in the last several years. A straightforward way to protect the media data is to apply the traditional cryptographic algorithms such as RC4 and AES to encrypt the whole bit stream. However, due to the large file size of the media data, this type of “blind” encryption scheme may not be efficient enough to meet the real-world requirements. More importantly, the media data typically exhibit well-defined structure which may be destroyed due to the “blind” encryption.

In order to design efficient as well as secure multimedia systems, there are a great number of multimedia encryption schemes proposed recently [1, 2, 3]. Unfortunately, intensive cryptanalysis results have been reported, showing that many of the existing multimedia encryption schemes are actually vulnerable against various attacks or inefficient in terms of computational complexity [4, 5].

More specifically, for audio data protection, audio scrambling is an efficient technique to break the intelligibility of the original audio with certain secret key such that only authorized users could have access to the original data [1]. In [1], Yan et al. proposed two progressive audio scrambling schemes directly applied in compressed domain. The first scheme is designed for protecting the raw PCM data, which is based on the Discrete Wavelet Transform (DWT). The basic idea is to first decompose the PCM data into several layers, and then perform XOR operations between the DWT coefficients with a key stream generated from a dynamic password generator (DPG). The progressive quality of the reconstructed audio could be achieved by controlling the number of layer keys assigned to the users. Yan et al. also presented another scrambling scheme for protecting the MP3 file [1]. The basic idea is to apply multiple rounds of XOR operations to the Huffman codewords according to a pre-defined key table.

In this paper, we address the security and efficiency issues of these two audio scrambling schemes in [1]. For the PCM audio scrambling scheme, we consider two cases of composing the key stream. We show that under both of these two cases the scrambling scheme is vulnerable against various attacks such as ciphertext-only attack, known-plaintext attack and chosen-plaintext attack. We also analyze the complexity of generating the key stream using DPG, and demonstrate that it would be much slower than a general pseudo-random number generator. Furthermore, for the MP3 audio scrambling scheme, we illustrate that we can efficiently recover the key table for scrambling using known-plaintext attack as well as chosen-plaintext attack. Based on the lessons drawn from the above analysis, we briefly discuss the ways to improve the security and efficiency of these two audio scrambling schemes.

The rest of this paper is organized as follows. In Section 2, we introduce these two audio scrambling schemes. The security and efficiency analysis of these two audio scrambling schemes are presented in Section 3 and Section 4, respectively. Section 5 describes the ways to improve these audio scrambling schemes. We conclude this paper in Section 6.

2. TWO AUDIO SCRAMBLING SCHEMES IN [1]

2.1. PCM audio scrambling based on DWT

The basic idea of this approach is to first decompose the PCM data into several layers using DWT, and then apply different keys to XOR the DWT coefficients for different layers. More specifically, the procedure of performing the PCM audio scrambling is as follows.

Step 1: Read the PCM samples.

Step 2: Perform DWT to these samples, and obtain the DWT coefficients $C_l$, for $1 \leq l \leq L$, where $C_l$ denotes the coefficients of layer $l$, and $L$ is the total number of layers.

Step 3: Initialize a counter $i = 1$.

Step 4: Attach layer $i$ with an ID$_i$ and a layer key KEY$_i$, where both ID$_i$ and KEY$_i$ are integers.

This work has been supported in part by the Innovation and Technology Commission of the Hong Kong Special Administrative Region, China (project no GHP/048/08).
Step 5: Input ID, and KEY, to a dynamic password generator to produce a key stream, denoted by Ei.

Step 6: Obtain the scrambled coefficients using

\[ C'_i = C_i \oplus E_i \]  

(1)

where C_i is assumed to be already binarized.

Step 7: Increment \( i = i + 1 \), and go to Step 4 until \( i > L \).

Step 8: Perform IDWT to the resulting C'_i, for 1 \( \leq i \leq L \), to construct the scrambled PCM samples.

Step 9: Write the file appropriately such that the file structure is preserved.

For more details please refer to [1].

2.2. MP3 audio scrambling

The MP3 audio scrambling algorithm consists of \( L \) rounds of scrambling operations. For each round 1 \( \leq l \leq L \), a sampling rate \( s_i(l) = a' \) is defined to determine the indexes of the audio data to be scrambled, where \( a \in Z \) is a constant. The procedure of performing the \( l \)th round of scrambling operation is as follows:

Step 1: Read the MP3 file, and let \( C = c_1c_2 \cdots c_a \) be the audio data to be scrambled.

Step 2: Create a key table \( K_T = \{key(i,j)\} \times q \) which is generated using a so-called Arnold matrix [1]. For simplicity, it is assumed that \( p = q \). More specifically,

\[ K_T = \mod (A \cdot R, p) \]  

(2)

where \( A \) is the Arnold matrix and \( R = \{r_{i,j}\} \times q \) with \( r_{i,j} = \text{rand(\text{range})} \) being a random number within a certain range.

Step 3: Initialize a counter \( t = 1 \).

Step 4: Calculate two quantities

\[ j_t = \mod (t, p) + 1 \]

\[ k_t = \mod (k_{t-1}, p) + 1 \]  

(3)

with initial value \( k_0 = 0 \).

Step 5: Scramble the audio data \( c_t \) as

\[ c'_t = c_t \oplus key(j_t, k_t) \]  

(4)

where \( key(j_t, k_t) \) is the entry in the key table \( K_T \) indexed by \( j_t \) and \( k_t \).

Step 6: Update the counter \( t = t + s_i(l) \).

Step 7: Repeat Step 4-6 until \( t > n \).

The keys used in different rounds of scrambling will be assigned to the corresponding authorized users to gain access to different layers. For more details please refer to [1].

3. SECURITY AND EFFICIENCY ANALYSIS OF THE PCM AUDIO SCRAMBLING SCHEME

In this section, we analyze the security and efficiency of the PCM audio scrambling scheme in [1]. It should be noted that the DPG in this scheme is used in a way like a pseudo-random number generator, and the generated key stream is directly XORed with the layer coefficients. A crucial point that was not stated clearly in [1] is how to encrypt \( C_i \) if its length \( |C_i| \) is larger than the key stream length \( |E_i| \). According to [1], \( |E_i| = 16 \) bits, for all the layers. However, \( |C_i| \) depends on the audio data length and the DWT parameters, and is generally larger than 16 bits for many applications such as music files distribution. In the following, we consider two cases of composing the key stream when \( |C_i| > |E_i| \).

Case 1: Repetitive use of \( E_i \). Namely, the actual key stream for the XOR operations is \( E_iE_iE_i \cdots \)  

Case 2: Request the DPG to generate more key stream (passwords).

In the following, we will show that under Case 1, the scrambling scheme is even vulnerable against the ciphertext-only attack, which is the weakest attack type among all the possible attacks. We then demonstrate the feasibility of using known-plaintext attack and chosen-plaintext attack to break the scrambling scheme under Case 2.

3.1. Ciphertext-only attack with Case 1

In this attack scenario, the attacker can only have access to the encrypted bit stream, and the objective is to derive the key stream or recover the plaintext directly. In the case that \( |C_i| > |E_i| \) and Case 1 holds, there would exist different parts of the plaintext that will be encrypted in an identical way. This leads to severe security weaknesses to the scrambling system. The attacker could combine the ciphertext parts encrypted with the same key stream that equals the combination of two plaintext messages and is independent of the key. The underlying statistics of the plaintext, e.g., the repetitive pattern of music signal due to chorus, might then be used to derive both the plaintext and the key stream [9]. This situation is similar to the case of stream cipher in which short period of the key stream is a significant drawback [9].

3.2. Known-plaintext attack with Case 2

In this attack scenario, the attacker is assumed to have several pairs of encrypted audio data and the corresponding original audio data. In practice, it is also feasible since the attacker may find the original audio data from the Internet or the audio files he/she already has according to the description or the meta data attached with the encrypted audio file. In addition, we assume that the encrypted audio data is always available to the attacker, which is quite reasonable according to Kerckhoff’s principle [8].

Different from Case 1, the key stream now is not of periodic pattern. However, after getting the plaintext/ciphertext pair, from (1), we can see that the key stream used to encrypt the \( r \)th layer coefficient \( C_r \) could be recovered as

\[ E_i = C'_i \oplus C_i \]  

(5)

where \( C'_i \) denotes the encrypted layer coefficient.

From the description of the DPG, we can find that for given ID, and layer key, the key stream generated from the DPG would be identical. In other words, different audio files will be encrypted using the same key stream. Therefore, the key stream recovered from (5) could be re-used to decrypt the consequent communications.
### 3.3. Chosen-plaintext attack with Case 2

In this attack scenario, the attacker is assumed to have temporary access to the encoder, and hence, can input arbitrary audio data and obtain the corresponding encrypted file. The attacker can therefore design a file consisting of all-zero layer coefficients. More specifically, \( C_i \) would be a zero sequence with long enough length. From (1), we can immediately get

\[
E_i = C_i
\]

#### 3.4. Efficiency Analysis

A key component in the PCM audio scrambling scheme is the DPG, which generates the necessary key stream for the XOR operations. However, it should be noted that the DPG is essentially an asymmetric encryption system, which is generally much slower than a symmetric system such as RC4 and AES. Let us in the following analyze the complexity of generating the key stream using the DPG.

In fact, the DPG in [1] is only a part of the original authentication system originally proposed in [6]. Let \( ID_i \) be the ID for layer \( i \), \( p_1 \) and \( p_2 \) are two large primes serving as the secret for layer \( i \). A series of passwords \( PW_{1,i} \) could then be generated in the following manner

\[
\begin{align*}
(PW_{1,i})^2 &= ID_i' \mod n \\
(PW_{2,i})^2 &= PW_{1,i} \mod n \\
&\vdots \\
(PW_{m,i})^2 &= PW_{m-1,i} \mod n
\end{align*}
\]

where
- \( ID_i' = ID_i \cdot k \), with \( k \) being either 1, 2, -2, or -1. This guarantees that \( ID_i' \) is a quadratic residue (QR) to both \( p_1 \) and \( p_2 \);
- \( n = p_1 \cdot p_2 \);
- \( m \) is a pre-defined constant originally used to limit the total number of login time.

In order to efficiently calculate \( PW_{1,i} \) from (7), we need to know the factorization \( n = p_1 \cdot p_2 \), which is also the most important advantage of an authorized user over an unauthorized one. Knowing this factorization, we can first calculate the quadratic roots

\[
\begin{align*}
(m_{p_1})^2 &= ID_i' \mod p_1 \\
(m_{p_2})^2 &= ID_i' \mod p_2
\end{align*}
\]

whose explicit solutions are

\[
\begin{align*}
m_{p_1} &= ID_i^{(p_1+4)/3} \mod p_1 \\
m_{p_2} &= ID_i^{(p_2+4)/3} \mod p_2
\end{align*}
\]

Therefore, the four possible solutions for \( PW_{1,i} \) are

\[
\begin{align*}
s_1 &= (y_{p_1} \cdot p_1 \cdot m_{p_2} + y_{p_2} \cdot p_2 \cdot m_{p_1}) \\
s_2 &= n - s_1 \\
s_3 &= (y_{p_1} \cdot p_1 \cdot m_{p_2} - y_{p_2} \cdot p_2 \cdot m_{p_1}) \\
s_4 &= n - s_3
\end{align*}
\]

where \( y_{p_1} \) and \( y_{p_2} \) are the Bézout numbers for \( p_1 \) and \( p_2 \), respectively. \( PW_{1,i} \) will then be determined as the unique root that is QR for both \( p_1 \) and \( p_2 \).

In (9) where modular exponentiation is involved, the complexity is \( O(\log(p_1 + p_2)) \) modular multiplications by using the right-to-left binary method [11]. It should also be noted that there are some other more efficient methods for modular exponentiation at the cost of more memory usage, such as the one in [10], where the algorithm needs \( O(h/ \log h) \) modular multiplications, where \( h \) is the length of the exponent, requiring an \( O(h^2) \) pre-computed look-up table size with small constant of proportionality.

In (10), the major computations come from the task of finding \( y_{p_1} \) and \( y_{p_2} \), which are the Bézout numbers for \( p_1 \) and \( p_2 \), respectively. This could be done using the extended Euclidean algorithm with complexity \( O(\log^3(\max\{p_1, p_2\})) \) modular multiplications [11].

Therefore, the total complexity of generating \( PW_{1,i} \) is of order \( O(\log^3(\max\{p_1, p_2\}) + \log(p_1 + p_2)) \) modular multiplications, where \( p_1 \) and \( p_2 \) are at least 200 digits to ensure the security according to Rivest [11]. Furthermore, from (7), we can see that there exists strong dependence between \( PW_{1,i} \), for different \( t \), making it difficult for parallel operations. This indicates that the DPG would be much slower than a general pseudo-random number generator which could typically produce hundreds of megabytes per second [8].

### 4. Security Analysis of the MP3 Audio Scrambling

In this section, we evaluate the security of the MP3 audio scrambling scheme in [1]. Since the key table \( K_T \) contains all the secret information, it would be sufficient to recover \( K_T \) in order to break the whole system. We use three types of attacks to evaluate the security of this scrambling scheme.

#### 4.1. Ciphertext-only attack

In this attack scenario, the only information that the attacker can have access to is the encrypted MP3 file. From (3), we find that the indexes \( j_t \) and \( k_t \) take the value 2, 3, \ldots, \( p_1 \), 2, \ldots, 1, 2, \ldots, \( p_1 \), 2, \ldots, \( p_1 \), 2, \ldots, respectively, with the increasing \( t \geq 1 \). Hence, it can be easily seen that both \( j_t \) and \( k_t \) are obtained from a periodic sequence with period \( p \). Equivalently, the key stream used for scrambling \( key(j_t, k_t) \) is also periodic with period \( p \), as the key table \( K_T \) is static. On the other hand, \( p \) is a small integer. A reasonable value of \( p \) is 16 according to [1].

In the case that the length of the MP3 data is longer than the period of the key stream, there would exist different parts of the plaintext that will be encrypted in an identical way. The attacker could combine the ciphertext parts encrypted with the same key stream that equals the combination of two plaintext messages and is independent of the key. The underlying statistics of the plaintext, e.g., the repetitive pattern of music
signal due to chorus, might then be used to derive both the plaintext and the key stream [9].

4.2. Known-plaintext attack

In this attack scenario, the attacker has access to the encrypted MP3 file and the original MP3 data. Since from [1], there is no operation to make the key table $K_F$ content-adaptive, we can therefore see that there exist different files encrypted using the same $K_F$. Suppose the attacker somehow obtains a famous music album, which is quite reasonable since he/she could get it from some online free resources or simply copy from his/her friends. Since the MP3 scrambling scheme is mainly designed for online music distribution, the online music store adopting this scrambling scheme very likely has this album. The attacker can then search in the online music store, and download the encrypted music album. In this way, the attacker can have access to both the plaintext and the corresponding ciphertext, which makes the known-plaintext attack very feasible in practice.

Let us then consider the $i$th round of scrambling. As the parameters $p$, $q$, and $a$ are publicly accessible, we know the indexes of the keys $(j_t, k_t)$ that are used for scrambling. Therefore, the attacker can recover the keys $key(j_t, k_t)$ using

$$key(j_t, k_t) = c_t \oplus c_t'$$

where $c_t$ is the original audio data and $c_t'$ is the encrypted audio data. Following the same fashion, we can recover all the keys that are used in all the rounds of scrambling.

4.3. Chosen-plaintext attack

In this attack scenario, the attacker has the additional freedom to input any MP3 file into the scrambling system, and get the corresponding output. This is feasible when the attacker somehow could obtain the temporary control of the system e.g., by using some malicious codes. He can then design a MP3 file consisting of all zero Huffman codewords with long enough length. The keys $key(j_t, k_t)$ can be recovered as

$$key(j_t, k_t) = c_t'$$

5. WAYS TO IMPROVE THESE TWO AUDIO SCRAMBLING SCHEMES

In this section, we briefly discuss the ways to improve the security and efficiency of these two scrambling schemes. For the PCM scrambling scheme, the key problems are how to avoid using the same key stream more than once, and how to generate the key stream efficiently. To this end, we introduce a hashing based approach to address these two problems. At the registration stage, each user will be assigned a unique secret key $H$. Upon receiving the request of the user, the server will generate random session keys $D_i$ for each layer. After that, the server will generate a layer seed by using the hashing function $S_t = hash(H, D_i)$, where the hashing function could be MD5 or SHA-1, which could be efficiently implemented. For each layer, a high speed pseudo-random number generator using $S_t$ as the seed will be adopted to generate long enough key stream to XOR with the DWT layer coefficients. Finally, the encrypted DWT coefficients will be subject to the IDWT to produce the encrypted PCM file. Upon receiving the encrypted PCM file together with the session keys $D_i$, which are publicly accessible, the user can perform similar steps to retrieve the original PCM data. For the MP3 audio scrambling scheme, we can treat the audio data encrypted in difficult rounds as the layer coefficients. Therefore, a similar improved strategy could be applied to improve the security and efficiency. Due to the page limit, we leave the detailed discussion and experimental results to our forthcoming paper.

6. CONCLUSIONS

In this paper, the security and efficiency of two recently proposed progressive audio scrambling schemes have been analyzed. In addition, brief discussions on how to improve the security and efficiency have also been presented.

7. REFERENCES