LARGE SCALE DISTRIBUTED WATERMARKING OF MULTICAST MEDIA THROUGH ENCRYPTION

Roland Parviainen, Peter Parnes
Department of Computer Science/Centre for Distance-spanning Technology
Luleå University of Technology, 971 87 Luleå, Sweden
Roland.Parviainen@cdt.luth.se, Peter.Parnes@cdt.luth.se

Abstract In this paper we describe a scheme in which each receiver of a multicast session receives a stream with a different, unique watermark, while still retaining the scalability of multicast. The watermarked streams can be used to trace those users who make unauthorized copies of a stream. The watermarking is enabled by encryption of two slightly different copies of the original stream with a large set of different keys.

Keywords: Watermarking, fingerprinting, multicast, multimedia

1. INTRODUCTION

IP Multicast [1] provides efficient many-to-many data distribution in an Internet environment. Senders send datagrams to a 'host group', a set of zero or more hosts identified by a single IP destination address. The datagrams are delivered to all members of the host group by the network infrastructure in an optimized way.

Multicast is very well suited to use for large scale media distribution because of the scalability: each network link in the network only has to transport one copy of each packet regardless of the number of receivers. The drawback is that receivers do not have to be authenticated and can easily eavesdrop on the traffic without being detected. A unicast solution, where we send one copy of the stream to each user, is easier to protect but is infeasible for large scale transmissions to 100,000 to 1000,000 users and above.

Authentication and confidentiality can be solved with the use of encryption, but there is still a problem with malicious users retransmitting the media data unencrypted to other users. One way to detect whom the illegal copy originated from is *fingerprinting,* embedding unique information, a watermark, into each copy of the media that identifies the user receiving the copy. This

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35413-2_36
R. Steinmetz et al. (eds.), *Communications and Multimedia Security Issues of the New Century*

information should be robust against any possible user manipulation, and in the remainder of this paper we will assume a robust watermarking scheme that can perform this fingerprinting exist. A fingerprinted stream might discourage illegal copying of the media, since the origin or the buyer of stream can be identified. This might be the only option for pure software solutions where tamper-resistant hardware is impossible.

The objectives of multicast and fingerprinting seem contradictory: multicast sends the same stream to everyone while to achieve the goals of fingerprinting every receiver should receive a different stream. In [2] Perkins, Brown and Crowcroft solve this problem by using active network elements that make sure all receivers get slightly different streams. In this paper we present a solution that does not suffer from the requirement of trusted active network elements. We propose a scheme were encryption is used to ensure different users receive fingerprinted streams. No trusted or active network elements are needed; all security is handled by the applications.

The remainder of this paper is structured as follows. In the next section we describe related work on multicast security. Then in section 3 we describe our approach in detail. In section 4 we describe possible attacks against the system. In section 5 an experimental implementation of this system is described. Sections 6 and 7 conclude this paper with limitations and conclusions.

2. BACKGROUND

2.1. MULTICAST SECURITY

Since IP multicast provides no authentication or confidentiality it is very easy to eavesdrop on, record and copy or retransmit a media stream completely anonymously. Basic encryption of multicast streams is not sufficient to protect important media streams since it is possible to retransmit either the content stream or the keys to untrusted users.

In [3] a 'virtual key' is marked instead of the plain-text. A certain minimum number of users need to collaborate to construct a key that works but identifies none of them. This does not prevent retransmission of media content and the bandwidth needed for the control messages may be too high.

Scalable content control schemes such as Nark [4] provides scalable authentication and encryption but need tamper-resistant smart-cards. Watermarking is not possible in todays limited smart-cards and have to be done off-card, but can be done using a system such as Chameleon [5].

Chameleon by Anderson and Manifacas is a similar scheme to ours, where a stream cipher is adapted to give slightly different output depending on a large unique key for each user. Our scheme can handle much more flexible watermarking algorithms; the two different watermarked packets do not have to have any common bits at all.

In the watercasting scheme [2] the source sends multiple subtly different copies of each packets and routers at the nodes in the multicast distribution tree discard packets, such that the stream delivered to each receiver is unique. This approach has several problems: support for this protocol in routers is needed which is probably hard to achieve, the routers have to be trusted, and the source must send d copies of each media packet, where d is the depth of the multicast tree.

2.2. WATERMARKING

A simple watermarking method is to change the least significant bits in for example an audio clip or an image. For an audio clip, we could put our embedded message into to least significant bit of some or all samples. These methods are easily broken. More advanced methods often use spread spectrum techniques or transforms such as Fourier and DCT to make the watermarks more robust.

A watermarking method that fulfills some requirements for the difficulty in removing it is called *robust.* Some examples of robustness requirements for audio recordings from IFPI, the International Federation for the Phonographic Industry [6] are:

- The sonic quality of the sound recording should not change
- The marking information should be recoverable after a wide range of filtering and processing operations, including two successive D/A and A/D conversions, MPEG compression, etc.
- There should be no other way to remove or alter the embedded information without sufficient degradation of the sound quality as to render it unusable

Similar requirements can be made for still images, video and other media types. A good introduction to the subject is [7].

3. METHOD DESCRIPTION

The source sends two different copies of each media packet, each with a different watermark. Both copies are encrypted with two different, random encryption keys. The encrypted packets are then sent to all receivers using IP multicast. Any given receiver has access to the key of only one of the two encrypted packets of one media packet.

3.1. TRANSMISSION OF PACKETS

The source has access to k media packets: $P[1], P[2], \cdots, P[k]$ and an encryption algorithm E, such that $P = D(E(P, K), K)$. That is, $E(P, K)$

encrypts P with key k and $D(P, K)$ decrypts P. A watermarking algorithm W: $P_w = W(P, w)$, $w = U(P_w)$ and two watermarks, w_0 and w_1 are also needed. W embeds the watermark w in the cover object P, and U extracts the watermark from the marked object. A detection algorithm that detects if the watermark is still present can be used instead: $U(P_w, w) = B, B \in \{true, false\}$. The source needs $2k$ random encryption keys, $SK[1], SK[2], \cdots, SK[2k]$, to be able to transmit the media packets. A receiver r has access to k of these keys: either $RK_r[i] = SK[2i-1]$ or $RK_r[i] = SK[2i]$, $i = \{1, 2, \cdots, k\}$.

To transmit media packet i the source perform the following operations:

1 Read the media packet, $P[i]$
2 Create two watermarked packets, $V_0[i] = W(P[i], w_0)$ and $V_1[i] = W(P[i], w_1)$
3 Get two encryption keys: $SK[2i-1]$ and $SK[2i]$
4 Encrypt $V_0[i]$ and $V_1[i]$: $C_0[i] = E(V_0[i], SK[2i-1])$ and $C_1[i] = E(V_1[i], SK[2i])$
5 Transmit $C_0[i]$ and $C_1[i]$ together with i

At the receiver side, the client receives both packets and tries to decrypt them:

1 Receive two packets: $C_0[i]$ and $C_1[i]$
2 Get the decryption key for packet i: $RK_r[i]$
3 Try to decrypt both packets with key $RK_r[i]$
4 Only one packet will decrypt into a proper media packet: $V_{j_i}[i] = D(C_{j_i}[i], RK_r[i])$, $j_i \in \{0, 1\}$
5 Decode and render $V_{j_i}[i]$

For each media packet the receiver will be able to decode exactly one of the watermarked packets. Which of the two packets is decided by the keys the source has assigned to the receiver.

3.2. IDENTITY STRINGS

If the keys a receiver have access to is unique among all receivers and known by the source, a unique identity string for that user can be defined: $id_r = B_r[1], B_r[2], \ldots, B_r[k], B_r[i] \in \{0, 1\}$.

This identity string can be derived by the source from both the keys given to the receiver and the stream the receiver decrypted. From the keys the source sent to the receiver:

$$B_r[i] = \begin{cases} 0, & \text{if } RK_r[i] = SK[2i-1] \\ 1, & \text{if } RK_r[i] = SK[2i] \end{cases}$$

From the decrypted stream for the user:

$$B_r[i] = \begin{cases} 0, & \text{if } U(V_{j_i}[i]) = w_0 \\ 1, & \text{if } U(V_{j_i}[i]) = w_1 \\ undefined, & \text{if neither } C_0[i] \text{ nor } C_1[i] \text{ was received or decrypted} \end{cases}$$

If the receiver do not receive all packets, due to for example packet loss or that the receiver tuned in late, the identity strings will not match completely. If n is large enough, the partial identity string will still be long enough to be unique among all receivers although some bits are undefined.

3.3. BANDWIDTH USAGE

Since 2 copies have to be sent for each media packet, the bandwidth usage is doubled both for the source and the receivers. This can be decreased by some optimizations.

At any given time, only one of the two watermarked packets is actually useful for a single receiver since the other packet can not be decrypted. If we send the two copies on different multicast groups the receivers can hop between the groups by joining and leaving them as the group the correct packet is transmitted on changes. In this approach we not only have to send the keys to each receiver but also which stream to receive; one extra bit for each key is needed. Unfortunately the join/leave latency for IP multicast is currently too large for this approach. Also, if more than one receiver is on the same network segment most of saving is lost.

As pointed out in [2] another optimization would be to only watermark 1 in every x packet, thus reducing the bandwidth to $(1 + 1/x)$ times the bandwidth of the original stream. Unfortunately a malicious user could remove these watermarked packets and retransmit the resulting degraded stream if x is large. We must therefore make sure that the degradation is large enough to discourage removal of the watermarked packets. One example of this is to only add watermarks to the I frames of an MPEG video stream or only watermark the last 10 minutes of a movie.

3.4. KEY DISTRIBUTION

The receiver keys can be treated as a long term key distributed by out-of-band means when the users registers, either as a downloadable file (protected by e.g. SSL/TLS [8]) or delivered to the user on a floppy or cdrom. All these solutions have problems when revocation of access is considered. The keys can also be continuously streamed to the users. The bandwidth per user needed for this is small, but it is still a serious scalability problem.

The amount of keys that each receiver needs depends on the required security (see section 4.1). The total size of the keys for one receiver is then

$keys \cdot keysize$. Using 10000 keys and a key size of 128 bits thus requires 160 kbyte per user. The source only has to store a bitmask of length $keys$ for each user and $2 \cdot keys \cdot keysize$ bytes of key material. A cryptographically secure random number generator can also generate the bitmasks instead to further reduce storage needs at the source.

3.5. KEY SIZE

It can be argued that attackers with sufficient funds to break, for example, 56 bit keys also have enough funds to get keys in other easier ways, for example using false identities while registering, and thus we do not need any stronger keys. On the other hand, getting access to computing power does not necessarily require monetary funds; a distributed attack is also possible.

It is not enough to break one of the keys since it only gives the attacker the option to change one of the watermarked packets in the stream. An attacker has to break a sufficient amount of keys to get enough packets to create an unidentifiable watermarked stream.

4. ATTACKS

We assume that it is not possible to either remove the watermark or break the encryption in reasonable time. We also assume that the attacker can not steal the non-watermarked stream from the source by breaking into the server.

If the encryption algorithm is broken an attacker can choose the final watermarked stream and make traitor tracing impossible, but if the encryption algorithm is chosen with care and with large enough key size and the keys are generated properly this can be avoided.

It is possible to attack another receiver's computer and steal the stream or keys from there, thus indicating someone else as the pirate. These kinds of attacks are out of scope for this paper.

Removal of the watermark might be possible if a robust watermarking algorithm is not chosen. The existence of such an algorithm is not considered in this paper.

Several users can collaborate and combine their streams to try to prevent watermark detection, either by choosing packets from different streams or by merging packets with for example bit voting. The two watermarks w_0 and w_1 and the watermarking algorithm should be chosen so that at least one of the watermarks is still present and recoverable after the bit voting.

Getting access to several streams and selecting packets from different streams is an easy and powerful attack. In the next section we analyze this attack further.

4.1. TRAITOR TRACING

If p users collaborate, at least k/p of the original bits from one of the streams will always remain. We must make sure that this fraction is large enough to ensure a high probability of detection of the collaborators.

We assume we have one identy string for the illegal stream id_I and n identy strings for the legal watermarked streams: $id_{L[1]}, id_{L[2]}, \ldots, id_{L[n]}$. We want to identify the identy string of at least one of the streams that was used to create the illegal stream. We can calculate a measure on how good a watermarked stream match the illegal stream by adding the XOR value of bits from the identity strings:

$$M(L[i], I) = \sum_{j=1}^{k} B_{L[i]}[j] \oplus B_I[j]$$

We define the set S_{nc} as the values $M(L[i], I)$ where i is not one of the collaborators, that is $L[i]$ was not used to create the illegal stream. S_c is the values $M(L[i], I)$ where i one of the collaborators. The values in S_{nc} have a binomial distribution $X_{nc} = Bin\left(k, \frac{1}{2}\right)$, and the distribution of the values in S_c is $X_c = Bin\left(k\left(1 - \frac{1}{p}\right), \frac{1}{2}\right) + \frac{k}{p}$. These distributions can be approximated with a normal distribution,

$$X_{nc} = N\left(\frac{k}{2}, \sqrt{\frac{k}{4}}\right)$$

and

$$X_c = N\left(\frac{k}{2}\left(1 + \frac{1}{p}\right), \sqrt{\frac{k}{4}\left(1 - \frac{1}{p}\right)}\right).$$

As the number of collaborators, p, increases X_c converges towards X_{nc} as expected. If p is relatively small, the probability that all values in S_c are less than $E[X_c]$ is very high. For example, if $n = 100000, p = 10, k = 10000$ the distributions are $X_{nc} = N(5000, 50)$ and $X_c = N(5500, 47.43)$. The probability that at least one of the values in S_c is greater than $E[X_c]$ is $A = 1 - P(X_c < E[X_c])^{100000} = 1 - P(X_c < 5500)^{100000} = 0.76 \cdot 10^{-18}$ The probability that at least one of the values in S_{nc} is greater that $E[X_c]$ is $B = 1 - P(X_{nc} < E[X_c])^{10} = 1 - \left(\frac{1}{2}\right)^{10} = 0.9990$. That is, there is a high probability that the identy string with the highest value of $M(L[i], I)$ is one of the collaborators. For $p = 50$ on the other hand, we have virtually no chance to detect any of the collaborators.

5. IMPLEMENTATION

To test the feasibility of our scheme it was implemented in an existing Java application system[9] for audio transmission over multicast which uses the MPEG-1[10] audio compression standard. The system consists of a server application and a client application. The server read MPEG-1 audio data from disk and send it to a multicast address using RTP[11], the client receives this data and decodes it. The MPEG decoding is not done in Java but in native code.

Blowfish was chosen as the encryption algorithm, and was provided by the reference implementation of the Java^{TM} Cryptography Extension (JCE) 1.2.1 from Sun. Since the implementation is not supposed to be used in a production environment a very weak watermarking algorithm was chosen for simplicity: a few otherwise unused bits in the MPEG audio frame header are used for the watermark.

Prior to the transmission, the server generate $2n$ random keys and keys for p users, where the values n and p are given. These keys are stored in files until they are used. If the stream is longer than the value of n, the keys 'wrap around': the keys for packet i is $SK[1 + (2i - 1 \mod n)]$ and $SK[1 + (2i \mod n)]$.

When we enable the watermarking scheme in the server the CPU usage increases from 1.0% to 12%[1]. The large increase is because the server now has to encrypt every packet twice. For the client application the increase is from 6.7% to 18%. The bandwidth usage is the same for both a client and a server, independent of the number of users. As expected, the bandwidth[2] usage increases by a factor of roughly two, from 133.3 kbyte/s to 270.4 kbyte/s. That the factor is 2.03 and not 2 is due to a small extra overhead for padding when doing encryption. The cost of key distribution is not included.

6. LIMITATIONS

Our scheme relies on the existence of a robust watermarking scheme. No perfectly robust watermarking algorithm has been proposed so far, and it is not clear such a algorithm is possible. Without any optimizations, we need at least twice the bandwidth of the original media stream which might be too much for some applications. We have not considered the problem of revoking access for a receiver; with the current scheme this would require that new keys have to be distributed to all receivers again.

Although only the use of two watermarks are described in this paper we are not limited to this. Instead of sending two different packets for each media packet it is possible to send any number of copies, although this increases the bandwidth substantially. The watermarks w_0 and w_1 do not have be constant

and can change at any time as long as $w_0 \neq w_1$ and the source keeps track of them.

7. CONCLUSION

We have described a scheme for scalable fingerprinting of multicasted media. Contrary to other proposed schemes no active network components or tamper-resistant smart-cards are needed. By increasing the bandwidth with a constant factor we can ensure that every receiver get a unique fingerprinted stream. The watermarks that make up the fingerprints are not fixed to a certain number of bits or format but can be of any format the watermarking algorithm requires for robustness.

Like other traitor tracing schemes based on watermarks, it has only limited collusion resistance. The most promising attack is to get hold of several streams and mix them together so the source can no longer be identified. The main countermeasure is to increase the number of watermarks in one stream.

Notes

1. Measured with perfmon.exe on a Pentium III 550 MHz computer running Windows NT4. The bit-rate of the media stream was 128 kbit/s and each key was 40 bits.
2. This includes RTP headers but not IP and UDP headers

References

[1] S. Deering. *Multicast Routing in a Datagram Internetwork.* PhD thesis, Stanford University, 1991.

[2] I. Brown, C. Perkins, and J. Crowcroft. Watercasting: Distributed watermarking of multicast media. In *Proceedings of the First International Workshop on Networked Group Communication*, 1999. Springer-Verlag Lecture Notes in Computer Science, 1736.

[3] B. Chor, A. Fiat, and M. Naor. Tracing Traitors. In *Advances in Cryptology---CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 257--270. Springer-Verlag, 1994.

[4] B. Briscoe and I. Fairman. Nark: Receiver-based multicast non-repudiation and key management. In *ACM Conference on Electronic Commerce*, 1999.

[5] R.J Anderson and C. Manifavas. Chameleon -- A New Kind of Stream Cipher. In *Fourth Workshop on Fast Software Encryption*, 1997.

[6] London W1R 5PJ International Federation of the Phonographic Industry, 54 Regent Street. Request for proposals - Embedded signalling systems issue 1.0., June 1997.

[7] S. Katzenbeisser and F.A.P. Petitcolas, editors. *Information Hiding: Techniques for Steganography and Digital Watermarking.* Artech House, 2000.

[8] T. Dierks and C. Allen. The TLS protocol, 1999. IETF RFC2246.

[9] R. Parviainen. Multicast Interactive Radio. In *Proceedings of the Practical Application of Java*, pages 137--153, 1999.

[10] MPEG Group. ISO/IEC International Standard 11172; coding of moving pictures and associated audio for digital storage media up to about 1,5 mbit/s, 1993.

[11] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A transport protocol for real-time applications, 1996. IETF RFC1889.