Using datagram based multimedia streams as a cover channel for hidden transmission

Ahmed Patel, Nikita Schmidt, Mikhail Bessonov

Computer Networks and Distributed Systems Research Group

Department of Computer Science, University College Dublin

Belfield, Dublin 4, Ireland

Tel.: +353 1 7062476 Fax: +353 1 2697262

Email: {apatel, cetus, mikeb}@net-cs.ucd.ie

Abstract

The paper describes a scheme for hidden transmission of secret information using established nonreliable packet-based multimedia stream as a cover channel. Solutions are proposed to address packet loss and synchronization problems with respect to covert communication in an open network. Use of both secret and public key cryptography is discussed. The technology described is affordable for mass user.

Keywords

Steganography, stream delivery, covert channel, subliminal channel

1 INTRODUCTION

While cryptography provides for message exchange with minimal risk of messages being decoded by a third party, steganography (Wayner 1996) is aimed at preventing the third party from discovering even the fact of existence of secret communication. It achieves this by spreading the bits carrying a message across a bulk of 'noisy' data (that is, data that have random bit patterns in which single bits can be changed without causing noticeable harm). As presented by Tuomas Aura (Aura 1995), the basic principles of steganography are:

- Messages are hidden using a public algorithm and a secret key.
- Only a holder of the correct key can detect, extract or prove to a third party the existence of the hidden messages. Nobody else should be able to find any statistically significant evidence of their existence.
- Even if the enemy knows or is able to select the contents of the hidden messages, this should be no advantage in detecting other hidden messages.

- The algorithm must be cryptographically strong. That is, it is theoretically impossible or computationally infeasible to detect hidden messages.

The public awareness of steganography and its techniques has increased substantially in the recent years. Software currently available enables a user to embed an encoded message in a graphic file of one of the popular formats. Due to the fact that only the lowest significant bits of pixels or Discrete Cosine Transform (DCT) coefficients are effected (and normally only a small fraction of those), the image degradation is hardly detectable by eye. This technique can as well be applied to other types of multimedia information (such as, for example, digitized audio files).

However, in order to be able to transmit messages of a large size a huge amount of covering image data is necessary. A new class of networking applications has become available for common use on the Internet recently. These are stream audio and video playback and conferencing packages, the most wellknown examples being CU-SeeMe and Internet Phone. These applications perform real-time transmission of digitized output from microphone and/or video camera which often lasts for hours. Enormous volume of network traffic provides an excellent cover channel for steganographic purposes.

A low delay between recording and playback is essential for conversational communications such as teleconferencing or telephony. This time constraint renders retransmission of lost packets impractical. Consequently, some portions of the data may be lost in transit. This problem makes it necessary to include error recovery elements in the steganographic scheme without using retransmission.

Another problem arising is the detection of the beginning of a secret message in a data stream. More than one secret messages can be inserted in a continuous data stream with arbitrary intervals between the messages. The approach discussed in this paper deals with the distinction between types of frames used in video streams. As far as audio streams are concerned, the sampling instant transferred by the Internet Real Time Protocol (RFC 1889 (1996), RFC 1890 (1996)) or another real-time protocol can be used.

Traditional steganography applications often presume the use of symmetric keys. The technique described in this paper was designed to be employable by a typical Internet user, who has no access to secure means for secret key exchange. The use of asymmetric keys is discussed with respect to the impact it has on steganographic strength of the transmission.

2 GENERAL MODEL OF OPERATION

We presume that the sender starts with the following:

- secret message, preferably compressed and/or encrypted;

Figure 1 The Encoder.

- the recipient's secret key;
- a cover audio/video stream to the recipient established.

The algorithm defines bit selection function, which takes the recipient's key and a packet to be transmitted as arguments and returns a set of bit positions in the packet. These selected bits are replaced with the bits of the secret message. To facilitate proper decoding by the recipient this bit selection function must, of course, return the same value even after the packet has been modified as described above. An example of such a function may be found in Tuomas Aura's paper (Aura 1995).

The sender performs the following steps, which are discussed in details further in the paper:

1. Select a suitable error control code and encodes the secret message with this code.
2. Reorder the bits of the encoded message to suppress the negative influence of error burst, possibly using the recipient's key.
3. Wait until appropriate synchronization point.
4. Generate a preamble for the secret message and inserts it into the stream, using only the bits identified by bit selection function.
5. Insert the encoded message using the bit selection function described above, as it is performed in most steganographic applications.

The operations are illustrated in Figure 1. The recipient performs reverse transformations.

3 FIGHTING THE PACKET LOSS

There are two basic ways to treat packet loss: ARQ (Automatic Repeat Request) and FEC (Forward Error Correction). ARQ means that the sender

retransmits the packets reported missing by the recipient. This technique is easily feasible in cases of two way communications, e.g. teleconferencing. It may be also achieved in case of one way communication in which the stream protocol used supports reporting of transfer statistics to the sender. Although the potential of ARQ with steganographic channels is significant, it is beyond the scope of the paper.

FEC allows to recover the content of a message even if a number of packets was lost in transit (and never retransmitted) by means of redundant coding. The easiest case of FEC is repeating every portion of information several times. No coding guarantees the delivery, but higher the redundancy rate is, higher it makes the probability of successful recovery.

There are many algorithms for redundant coding used in communications (Peterson and Weldon 1972). The selection of the coding method is determined by the error distribution typical for the particular channel. Specific coding algorithms are invented to achieve optimal performance with particular error distributions. The following properties of the stream used for cover channel are important in our scheme:

- With packet switched networks (like Internet) errors usually happen in bursts, i.e. the probability of packet $N + 1$ being lost is considerably higher than average if the packet N is lost.
- The lower layer protocols used in stream delivery, e.g. the User Datagram Protocol (RFC 768 (1980)), can detect packet loss or distortion, so it is always known which bits of a secret message are undetermined. This makes a contrast to traditional bit channels where no bit received can be trusted.

Block error control and correction codes can be classified by the Hamming distance and the number of bits recovered. We do not need the recovering capability since we know the positions of missing bits. For that reason the error correction codes contain excessive information which is not required in the proposed scheme. We use just error control codes.

Since the positions of the bits lost in transfer are known, an attempt can be made to recover the missing bits by supplying all the possible combinations of 1 and 0 in missing bits until the code no longer detects an error. If the code used has a Hamming distance of d , we are guaranteed to recover successfully up to $d - 1$ missing bits in a code block (Wayner 1996).

The selection of error detecting code is determined by (a) the length of the secret message, (b) the probability of packet loss in transfer and (c) the desired probability of successful delivery of the whole secret message. There is a trade-off between the probability of successful delivery of a message of a given length and the amount of overhead which is required. This overhead effectively lowers data throughput of the secret message.

The summary of encoding methods, maximum message sizes and the resulting overheads is presented in Table 1. The calculations are made for the

Table 1 The effect of error detecting codes

<i>Max. size (KB)</i>	<i>Overhead (times)</i>	<i>Coding method</i>
2	3.10	BCH (31,10) code w/o odd-weight code words
3	3.33	Cyclic (30,9) code
4	3.63	Cyclic (29,8) code
6	4.00	Cyclic (28,7) code
15	4.33	Calabi & Myrvaagnes (26,6)
36	4.80	MacDonald (24,5) code
123	5.00	MacDonald (30,6) code w/o parity digit
306	5.60	MacDonald (28,5) code
444	6.75	Cyclic (27,4) code
626	8.00	Cyclic (8,1) code
1234	8.33	Calabi & Myrvaagnes (25,3) w. overall parity
6261	9.00	Cyclic (9,1) code
62613	10.00	Cyclic (10,1) code
626166	11.00	Cyclic (11,1) code

packet loss probability of 10% (stream delivery becomes impossible if it is higher, as found by experience of using the Internet) and assumed probability of successful delivery of the secret message being 95% (as would be desired by user).

The fact that packets are always lost as a whole makes it necessary to shuffle the bits somehow after applying error control coding so that the bits belonging to the same code word get distributed among several packets. The simplest approach is putting first bits of a number of code words to the first packet, second bits of those words to the second packet and so on. Then the next bunch of code words is taken and the process is repeated. Another approach is generating pseudorandom permutations of bits in sequences of code words using the secret key (Luby and Rackoff 1988).

4 FINDING THE MESSAGE

Since audio/video streams can be particularly long and can carry more than one secret message, it is necessary to somehow mark the beginning of each secret message in a stream. That is achieved through a mechanism of preambles. A preamble is a short data structure steganographically inserted in a data stream. The recipient scans incoming packages looking for a preamble. When it finds one, it can determine where the secret message starts. A preamble is therefore a sort of synchronization signal. Preambles must satisfy the following requirements:

- Since preambles must be undetectable by an eavesdropper, secret password must be necessary to recognize a preamble.
- It must be recognised as such by the recipient, who knows the key, e.g. by means of checking for a predefined bit sequence (so called ‘magic number’).
- It must provide the recipient with important information about the secret message, such as the number of the packet containing the beginning of the message, message length, and encoding.
- It must contain random bits to avoid compromising the secret key.

Like the secret message itself, the preamble is inserted only into those bits of the stream which are pointed to by the bit selection function mentioned above.

Since the detection of a preamble is critical for the message delivery, it is proposed that the preamble be repeated several times in the cover stream. The presence of random bits combined with the use of bit selection function makes such repeating of the preamble to be steganographically safe.

If the decryption algorithm is slow, it makes sense to provide an interval between a preamble and the start of the actual message, i.e. the preamble should reference a packet in the stream which will come considerably later during transfer.

To avoid extra load imposed on the receiver by decoding and testing every stream packet, it is preferred that only the ‘natural’ synchronization points in the stream to be used to carry preambles. Such natural points are, for instance, I-frames in MPEG-1 (ISO/IEC 1993), MPEG-2 (ISO/IEC 1995) and H.261 (ITU-T 1993) encoded video streams. With audio, there are no well defined frames or similar entities, but the packets of the stream typically store timing information which can be used. The first packet recorded every second is usually a decent choice, because it gives enough time to check for preambles without forcing unreasonably long intervals between secret messages.

5 USE OF ASYMMETRIC KEYS

The technique discussed so far presumes the use of secret keys. It is well known that the exchange of secret keys by secure means is often costly and inconvenient, so here an attempt is made to extend the scheme to use public key cryptography. The proposed approach is to pass a secret key encrypted with the recipient’s public key; after that secret messages can be transmitted using this secret key. Of course, in this case the bits selected to carry the secret key will be known to anybody (in fact, there is no need for bit selection function). Thus such a transmission is more vulnerable to discovery by statistical analysis.

However, (a) the message containing encrypted key is rather short and (b) the bits of this message have almost uniform distribution. Since the lowest significant bits of the cover channel are assumed to be random as well as having

almost uniform distribution, this method is presumed to be statistically strong for practical use. Although the use of asymmetrical keys is not as strong as the use of symmetrical secret keys, it is still suitable for not mission critical applications.

6 POSSIBLE VULNERABILITIES

The method depends heavily on the randomness of the bits used as the cover. Inaccurate selection of the cover stream makes hidden data easily detectible. An example of definitely wrong cover stream is a stream which many repetitions of exactly the same piece, e.g. a video obtained by playing the same short film many times. Obviously, one should not use the same cover for more than one message, as one should not use the streams, unmodified originals of which can be obtained by anybody else. As far as video samples are concerned, the ones with a lot of motion and mixed colours are preferred. Generally speaking, the masking source must be 'noisy' in its own right. A perfect background for hiding data in audio transmission would be something like a kitchen with bacon and eggs being toasted, which guarantees the presence of strong noise spread all over the frequency spectrum.

It is essential that the cover stream contains some really random patterns. For example, transmission of noncompressed audio samples will suffice if the least significant bit of each sample is random. In practice, consumer (as opposed to professional) equipment normally provides the necessary random noise level, as shown by the following experiment.

Several seconds of silence (without any sound source plugged into the sound-card) were recorded at an average input gain level with 16 bits per sample resolution, 8000 samples per second. The samples recorded were then processed by a correlation analysis:

$$C(\delta) = \frac{\sum_{i=1}^{M-N} (x_i - \bar{x})(x_{i+\delta} - \bar{x})}{\sum_{i=1}^{M-N} (x_i - \bar{x})^2}, \quad \delta = 1, \dots, N,$$

where M is the number of samples and N was taken to be 16 000, thus measuring the correlation within intervals of up to two seconds.

The same analysis was also performed on samples which inherited a limited amount of low order bits from the original ones and had zeros in the high order bits. The results, shown in Figures 2 and 3, demonstrate a strong intersample correlation in case when all 16 bits were preserved, which is still noticeable with 10 bits. When only 6 bits were left, the correlation practically disappeared. Notice that while the behaviour of full samples is quite different among the two cards considered (periodical oscillation in the first case and a slow drift in the second), the behaviour of the 6 least significant bits is quite similar and has a white noise-like spectrum.

Figure 2 Covariance graphs for soundcard noise (sound card 1).

Figure 3 Covariance graphs for soundcard noise (sound card 2).

Even with truly random noise present in the masking data, there is still a theoretical possibility of discovering the fact of hidden transmission. The problem is general and it is present with most of the steganography applications. Covariance analysis like the one shown above is only one of many similar tools available. There can always exist hidden dependencies between lower bits of the samples which can probably be detected by a more sophisticated analysis.

Essentially, both the side engaged in the covert data transmission (sender) and the side seeking to be able to intercept such transmission (eavesdropper) have some mathematical models of the cover data and the payload. The eavesdropper suspects a hidden transmission if the stream does not fit into his model of the cover data. If the sender has the same (or more restrictive) model as the eavesdropper, it is always possible to hide the fact of covert transmission. The sender's advantage in this race is the ability to optimise its model for the particular cover data source used, having as much statistical material as needed. The eavesdropper is forced to create models describing more general masking data sources. However, the sender can never know for sure what kind of models are used by potential eavesdroppers.

When the cover data are not random, an eavesdropper can be almost 100% sure that a hidden transmission is being attempted if the data do not fit his model. But with sufficiently random data there is always a possibility that a particular pattern can happen under real circumstances even though it does not fit the model and vice versa, so that a fact of covert communication can only be detected to a certain degree of probability. And the more random the data, the less the degree. The use of only a small portion of masking bits, with bit selection function depending on a secret key, makes detection radically harder even with a superior cover channel model.

7 CONCLUSION

The scheme described in this paper is applicable to transmit covert data over nonreliable multimedia stream connections. The required technology and equipment to implement this scheme are available to mass user. The size of the secret messages is not limited by the size of the underlying cover files (as it is with static image files, e.g. JPEG) to be transmitted. The excessive overhead caused by FEC for transmission of long messages can be eliminated by employing the ARQ scheme which is outside the scope of the paper but worthy of research. The advantage of FEC, however, is that it is relatively simple, cheap and can be used in unidirectional (including multicast) environments.

REFERENCES

- Wayner, P. (1996) *Disappearing Cryptography: Being and Nothingness on the Net*. AP Professional (Academic Press), Chestnut Hill, Massachusetts.

- Aura, T. (1995) Invisible communication, in *Proceedings of the HUT Seminar on Security '95*, Espoo, Finland.
- Luby, M. and Rackoff, C. (1988) How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, **17**(2), 373–386.
- Peterson, W.W. and Weldon, E.J. (1972) *Error Correcting Codes*. The MIT Press.
- Schulzrinne, H., Casner, S., Frederick, R. and Jacobson, V. (1996) *RTP: A Transport Protocol for Real-Time Applications*. RFC 1889.
- Schulzrinne, H. (1996) *RTP Profile for Audio and Video Conferences with Minimal Control*. RFC 1890.
- Postel, J. (1980) *User Datagram Protocol*. RFC 768.
- ISO/IEC 11172 (1993) *Information Technology: coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s*.
- ISO/IEC 13818 (1995) *Generic coding of moving pictures and associated audio information*.
- ITU-T H.261 (1993) *Recommendation H.261 - Video codec for audiovisual services at $p \times 64$ kbit/s*.

8 BIOGRAPHIES

Ahmed Patel

Ahmed Patel received his M.Sc. and Ph.D. in Computer Science from Trinity College Dublin in 1977 and 1984, respectively. From 1978–82 he was responsible for developing the Irish Universities Data Network and from 1982–85 he developed the EuroKom computer conferencing and electronic mail service used by R&D projects in Europe. At University College Dublin he is a lecturer in computer science and head of the Computer Networks and Distributed Systems Research Group. He is involved in various multinational R&D projects in ESPRIT, RACE, INFOSEC, AIM, COST and the Irish National Telecommunications programmes. His main research interests include network management, security, protocols, performance evaluation, intelligent networks, CSCW and open distributed processing systems. He has published many technical papers and co-authored two books on computer network security and one book on group communications. He is a member of the Editorial Advisory Board of the Computer Communications and Collaborative Computing journals.

Nikita Schmidt

Nikita Schmidt received his M.Sc. in Mathematics from St.Petersburg State University in 1994. He currently makes his Ph.D. research in the area of Com-

puter Networks and Distributed Systems at University College Dublin. He participates in the UNIVERSE research project (EU Télématix for Libraries Programme), which aims at providing transparent concurrent access to multiple library databases. His research interests include distributed networking and stream and multimedia delivery over heterogeneous networks.

Mikhail Bessonov

Mikhail Bessonov received his M.Sc. in Mathematics from St.Petersburg State University in 1993. He currently makes his Ph.D. research in the area of Computer Networks and Distributed Systems at University College Dublin. He is active in the GAIA (ACTS) research project, which deals with electronic brokerage and delivery of multimedia content. His research interests include distributed object architectures, stream multimedia over Internet, and distributed information search and retrieval.