On Error Preserving Encryption Algorithms for Wireless Video Transmission

Ali Şaman Tosun, Wu-chi Feng
The Ohio State University
Department of Computer and Information Science
Columbus, OH 43210
{tosun,wuchi}@cis.ohio-state.edu

ABSTRACT
In this paper, we describe error preserving encryption mechanisms for transmission of video over wireless networks. One of the main problems with the secure transmission of data over wireless networks is that the bit errors that occur need to typically be resolved *before* decryption can begin. For video streaming applications, this is unacceptable due to the general requirement that video be presented to the user in a continuous manner with low latency. In this paper, we describe a systematic approach to understanding error preserving encryption algorithms. That is, encryption algorithms designed specifically for video to solve this problem. The main objective of this work is to ensure that the basic encryption of the stream can survive bit errors and that the errors are then passed to the application. We make use of the fact that video compression typically results in random byte distribution. Error preserving encryption algorithms are secure against ciphertext only attacks but vulnerable against known plaintext attacks. We limit this vulnerability by requiring a key exchange for each session.

Keywords
Video encryption, wireless video transmission

1. INTRODUCTION
With the continued advances in computing and networking technologies, the use of wireless devices for multimedia retrieval is becoming possible. The distribution of multimedia data, including video, to wireless devices presents significant hurdles to both high quality content delivery as well as the preservation of rights management to the data itself.

As the ability to capture and distribute multimedia data becomes much easier, digital rights management is becoming a key issue in how multimedia systems are used. A current example is that of the Napster song distribution network. One of the key points of contention from the music industry perspective is that to some degree they have lost control of the distribution of the content and that the blatant copying of these songs was an infringement on their ownership rights to the material. Digitals rights management systems are needed to prevent such problems for video distribution. Since video data has unique characteristics such as large size, time requirements to present it in a continusous manner, we need light-weight encryption, decryption algorithms.

When we mix digital rights management and the transmission of the data over wireless networks, several bad side-effects manifest themselves when transmitted over wireless networks. First, many of the standard encryption methodologies that exist today require that all bits received after the transmission of data are correct. This is due mainly to some of the inherent properties required in designing good cryptographic systems. As a result of this, one can expect significant overhead due to retransmissions and forward-error-correction in applying standard encryption techniques to multimedia data. Second, by requiring that all bit must be received correctly, the systems cannot take advantage of the fact that some of the multimedia applications are somewhat tolerable to network bit errors.

Recently, there have been several systems designed for reducing the complexity of encryption algorithms for compressed video data. By taking advantage of the special properties of video coding, several promising results were obtained. While these techniques help to minimize the amount of overhead required to secure a video stream, they do not work that well when used over wireless networks. The main reason for this is that they typically do not preserve bit errors from the wireless transmission, resulting in streams that cannot survive the decryption process without significant processing errors.

In this paper, we describe a theory with which on can build *error preserving video encryption algorithms*. The objective of this work is to understand permutation-complementation based encryption algorithms that take advantage of the fact that compressed multimedia data is already random in byte distributions. We present several formal properties of such systems and describe how to apply these in securing video data.

2. RELATED WORK

For brevity purposes, we assume readers are familiar with the MPEG video compression standards. Readers not familiar with MPEG are referred to references [1] and [3].

2.1 Video Security Algorithms

Historically the straight-forward application of *standard* encryption algorithms to *entire* video streams adds too much computational overhead because the all the data needs to be decrypted and is typically very large. Because MPEG decoding is a computationally intensive process, solutions were proposed to add minimal complexity for encryption and decryption.

The ZigZag-Permutation algorithm, which permutes the AC coefficients, is one of the earliest algorithms designed for MPEG encryption [11]. In this technique, the DCT coefficients are encoded in a different order than the standard MPEG video zig-zag order. Thus, in order to decrypt the sequence, an attacker must know the order in which the coefficients are encoded. Furthermore, by reordering the coefficients as they appear in the compressed bit-stream, this algorithm results in slightly lower compressions ratios due to the decrease in the average run-length size. This algorithm is very fast but has been shown to have serious security problems [10]. The main problem is the fact that only the AC values are permuted leaving all header information in plaintext, allowing for simple guessing algorithms to break the encryption.

In [14], the authors propose a real-time, software encryption algorithm using selective coding of I frames. This approach encrypts the headers and the I-frames using the DES encryption algorithm. It has been shown that this technique has security problems due to the intracoded macroblocks present in unencrypted P and B frames [7]. Intracoded macroblocks are encoded independently of other frames and blocks and thus their presence in P and B frames can be decoded correctly without the I-frame. So even if I frames are encrypted, weonecan see parts of the image in P and B frames because of these I-blocks.

More recently, Qiao and Nahrstedt proposed a secure MPEG encryption algorithm which uses DES [9]. It encodes only half of the data in a stream using DES and has a 47 percent reduction in the number of XOR operations executed over DES. Thus, this approach can be performed in real-time.

2.2 Forward Error Correction

Forward Error Correction is a common technique used to decrease losses when data is transmitted over noisy channels. It adds some redundancy to the data stream so that errors can be detected and corrected. This comes at a cost of additional overhead for the transmission of the FEC bits. FEC codes such as BCH can be designed to correct a given number of bit errors. If a BCH code can detect up to c bit errors then it can correct $\lfloor c/2 \rfloor$ bit errors. If the number of bit errors in the data is more than c, the BCH code may not detect the error. Thus, there are no easy mechanisms that tells us whether the frame is in error. The overhead of BCH is linear in the number of bit errors it can correct. Bit errors left in data after decoding are left to the application.

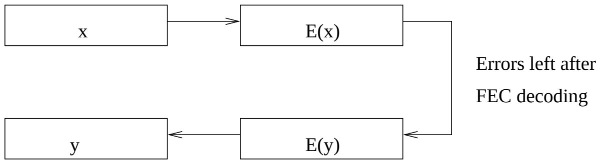

Figure 1: Encryption for wireless video transmission

Effect of single and double bit errors on MPEG player were addressed in [15]

2.3 Security Issues Relevant to Wireless Video Transmission

There are several properties that describe good cryptosystems. *The Error propagation property* [4] requires that cryptograms of close messages should be distributed over the whole cryptogram space. That is, messages that are similar should yield encryptions that are significantly different to prevent known plaintext attacks. A cryptographic transformation has the *avalanche effect* [4], if whenever a single input bit is complemented, on average one half of the output bits change. The output response of single-bit input change and double-bit input changes for cryptographic systems are investigated in [12]. A cryptographic transformation is *complete* [8] if each ciphertext bit depends on all the plaintext bits. These properties are bad for wireless video encryption because any bit errors left in encrypted streams after FEC decoding causes loss of data.

2.3.1 An Example of The Intolerance of Encrypted Streams To Bit Errors

To show how bit errors that might be introduced into the stream by the failure of FEC affect the encryption process. We created a text file and encrypted it using PGP. Then we introduced single bit errors into the encrypted file. For the decryption a key phrase is entered by the user. This phrase substitutes the key. When we introduced the bit errors we were not able to decrypt the file. pgp returned an *bad pass phrase error* and did not decrypt the file and no text was returned.

This is in fact a consequence of the error propagation, avalanche and completeness properties of security algorithms.

3. A THEORY FOR UNDERSTANDING ERROR PRESERVING ENCRYPTION FUNCTIONS

As we described previously, there are a number of conflicting goals when transmitting video over wireless networks. We want to be able to provide digital rights management, but we have limited processing resources, bandwidth and use of FEC might render significant amount of data useless if FEC can not correct the bit errors. Using FEC that corrects a high percentage of bit errors, wastes bandwidth since such FEC has high overhead. Consider the following scenario using FEC that can correct 5 bits for wireless transmission of video. Some bit errors might be left in the data after FEC decoding. In such a situation, the number of bit errors will be 6,7 or 8 (slightly higher than FEC capability) with high probability. An ideal encryption function would be one which satisfies the condition that if $E(x)$ and $E(y)$ differ in $c \leq 8$ bits, then x and y differ in $d \leq c$ bits where *E(x)* denotes encryption of x. It can be shown that if the encryption function decreases some distances then it must increase some other distances. So one class of functions which satisfy the above condition is the class which preserves the distance. We concentrate on encryption functions with the property that if *E(x)* and *E(y)* differ in i bits then x and y also differ in i bits. This process is given in Figure 1. With errors possibly left after FEC decoding $E(y)$ is given to the decryption algorithm. Note that $E(y)$ is $E(x)$ with errors. If $E(x)$ and $E(y)$ differ in k bits, we want x and y to differ in k bits. Using typical encryption algorithms, x and y will differ in about half the bits. In this section we identify all functions which preserve bit errors and provide methods to generate such functions efficiently for video encryption for transmission over wireless networks.

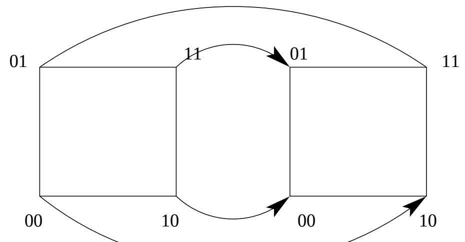

Figure 2: Example of an error preserving function

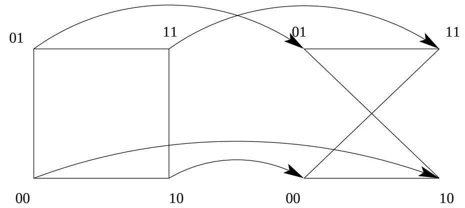

Figure 3: Example of a function which is not error preserving

To produce a framework for how to do this, we will first describe several properties and lemmas that are needed. We then describe how to apply these to video streams. Let *f* be a function of the form $f : \{0,1\}^n \rightarrow \{0,1\}^n$ which means that *f* maps binary strings of length n to binary strings of length n. Let $d(x, y)$ be the hamming distance between x and y. By definition of hamming distance $d(x, y)$ is the number of bits in which x and y differ. We call a function of the above form *error-preserving* if $d(x, y) = d(f(x), f(y))\ \forall x, y$, in other words if x and y differ in k bits then *f(x)* and *f(y)* also differ in k bits. To help visualize such functions we introduce neighbor graphs. Neighbor graphs form a geometrical shape and help understand error preserving functions. We can construct a *neighbor graph NG* of $\{0,1\}^n$ by connecting x and y if *d(x,y)=1*. As an example consider binary strings of length 2. There are 4 such strings $00, 01, 10, 11$. 00 and 01 differ in 1 bit, connect them with an edge, Similarly connect 00 and 10, 01 and 11, 10 and 11. The resulting shape is a square. Similarly construct a neighbor graph *NG(f)* by connecting *f(x)* and *f(y)* if *d(f(x),f(y))=1*. Connect 2 nodes if $f(x)$ and $f(y)$ differ in 1 bit. Neighbor graphs NG and $NG(f)$ form n-dimensional hypercube. For n=2, they are squares, for n=3, they are cubes. A function *f* is error preserving if it preserves the shape of the hypercube. Error preserving functions map vertices of NG to $NG(f)$ and when doing that shape of the n-dimensional hypercube is preserved. If x and y form an edge in NG, then $f(x)$ and $f(y)$ should also form an edge in $NG(f)$. Figure 2 gives an example of a function which is error preserving and Figure 3 gives an example of a function which is not error preserving. In Figure 3 the function does not preserve the shape of the square. In Figures 2 and 3 left part shows NG and right part shown $NG(f)$. In algebra, a transformation that preserve shape is called an *isometry*. We use the concept of isometry because we use geometry for some of the proofs. So the number of error-preserving functions is equal to the number of isometries of n-dimensional hypercube. If a function is error preserving it will be an isometry and if it is not then there must be an edge which violates error preserving property and the function won't be an isometry. We refer to readers not familiar with algebra to [5]

3.1 Theoretical Results on Error Preserving Encryption Functions

In this subsection we prove the following lemmas. Readers that do not care to read the proofs throughly can move to the following section.

Lemma 1: The Number of error preserving functions $\{0,1\}^n \rightarrow \{0,1\}^n$ is $n! * 2^n$

Proof: We want to find functions which map an n-dimensional hypercube onto another n-dimensional hypercube so that bit errors are preserved. Determining the mapping for 0^n and its neighbors determines the n-dimensional hypercube. There are 2^n possibilities for 0^n since any given hypercube has 2^n vertices. 0^n has n neighbors. We can map them in *n!* ways. Therefore there are $n! * 2^n$ error preserving functions.

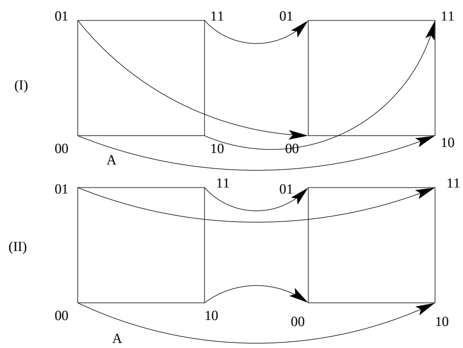

Figure 4: Sample isometries of square

As an example, consider the simplest case where n = 2. In this case, the hypercube is a square. We can map 00 to any one of the 4 vertices. Lets say we map 00 to 10 (labeled A in graph). 00 has neighbors 01 and 10. We have 2 choices for mapping its 2 neighbors. Either we map 01 to 00 and 10 to 11 (case I) or we map 01 to 11 and 10 to 00 (case II). It is easier to visualize using Figure 4. Depending on how we map the neighbors we get the isometry *rotation to left by 90* or *reflection along y axis.*

Lemma 2: All error-preserving functions can be generated using permutation and complementation. Permutation permutes the positions of bits and complementation complements a subset of the bits.

Proof: Permuting the positions of bits is an error-preserving function. Complementing a subset of the bits is also an error-preserving function. Now consider all the functions which can be formed by permutation followed by complementation of a subset of the bits. We can permute bits in $n!$ ways. We can select a subset of bits in $\{0,1\}^n$ in 2^n different ways. Therefore there are $n! * 2^n$ functions that can be formed using permutation followed by complementation. To show that all error-preserving functions are generated this way, we need to show that generated functions are unique. Assume an error-preserving function generated by P_1C_1 where P_1 is a permutation and C_1 is a complementation of a subset of bits, assume that the same function is also generated by P_2C_2. We have to show $P_1 = P_2$ and $C_1 = C_2$. Clearly if $P_1 = P_2$ then $C_1 = C_2$ and if $C_1 = C_2$ then $P_1 = P_2$. So assume $P_1 \neq P_2$ and $C_1 \neq C_2$. Since P_1 is a permutation P_1 maps 0^n to 0^n. Since P_2 is a permutation P_2 maps 0^n to 0^n. Since $P_1C_1 = P_2C_2$, C_1 and C_2 complement same bits. Therefore $C_1 = C_2$ which implies $P_1 = P_2$. This is a contradiction. So Permutation complementation generates a total of $n! * 2^n$ error-preserving functions and it generates each error preserving function only once. Therefore it generates all the error-preserving functions using lemma 1.

Lemma 3: $(P_1C_1)\circ(P_2C_2) = (P_3C_3)$ where $\circ$ denotes function composition.

Proof: Let $f = (P_1C_1)$ and $g = (P_2C_2)$. Then $(P_1C_1)\circ(P_2C_2) = f \circ g$. From $(f \circ g)(x) = g(f(x))$ using the fact that g is error preserving we have $d(g(f(x)), g(f(y))) = d(f(x), f(y))$. And using the fact that f is error preserving we have $d(f(x), f(y)) = d(x, y)$. Therefore $d(g(f(x)), g(f(y))) = d(x, y)$ and $(P_1C_1)\circ(P_2C_2)$ is error preserving and it can be written as (P_3C_3) by lemma 2.

For simplicity we will not use $\circ$ for the rest of the paper.

Lemma 4: $(P_1C_1)^{-1}(P_2C_2) = (P_3C_3)$

Proof: For simplicity, in this proof we use the fact that isometries of an n-dimensional hypercube form a group. Therefore inverses exist. We have $(P_1C_1)(P_2C_2) = (P_3C_3)$ from lemma 3. Multiplying on the left by $(P_1C_1)^{-1}$ we get $(P_2C_2) = (P_1C_1)^{-1}(P_3C_3)$ which is in the form above.

Lemma 5: $(P_1C_1)(P_2C_2)^{-1}(P_3C_3) = (P_4C_4)$

Proof: Immediate using the fact that isometries form a group, lemma 4 and lemma 3.

Lemma 6: Number of permutations which map none of the bits to their original position is given by $D_n = n!(1 - 1/(1!) + 1/(2!) - 1/(3!) + ... + (-1)^n 1/(n!))$.

Proof: Such permutations are called derangements in combinatorics. We refer the readers not familiar to combinatorics to [2].

Lemma 7: Using the random Derangement-Complementation (D-C) algorithm. The probability that k users will all have distinct (D-C) is

$$\prod_{j=1}^{k}\left(\frac{D_n 2^n - j}{D_n 2^n}\right)$$

which is larger than

$$2^{k\log\left(\frac{D_n 2^n - k}{D_n 2^n}\right)}$$

4. APPLICATIONS OF ERROR PRESERVING ENCRYPTION FUNCTIONS

In this section ,we describe some applications of the theorems given in previous section and describe why they are important for multimedia transmission over wireless networks.

Let's first discuss the use of error-preserving functions in encryption. Such functions won't satisfy error propagation property, avalanche effect and completeness property. These properties mainly guard against known plaintext attack. That is an attacker knows a message *t*, its encrypted version *E(t)* and is trying to decode another encrypted message *E(u)*. Such an attack to error preserving functions can be done as follows. The attacker finds how many bits differ in *E(t)* and *E(u)*, let's say *a* bits differ. He then knows that the plaintext *t* and *u* differ in *a* bits. He can then exhaustively search for those bits. The number of texts which satisfy it are given as combination *C(n,a)* where *n* is the length of the message.

Considering limited bandwidth, processing resources and power and overhead of FEC to guarantee very low bit-error-rate, we are not in a position to support security at the same level as wired networks. Clearly encryption schemes that satisfy error propagation property, avalanche effect and completeness property are not suitable for wireless video transmission. Our approach to this problem is as follows. For each video transmission session we use a different (P-C) pair to minimize the chances of known plaintext attack. It is also possible to change the (P-C) pair regularly to guard against such attacks.

Another important issue is why a (P-C) based encryption is suitable for video transmission. Let's first concentrate on the permutation part to motivate the issue. In general, permutation is not a secure encryption method because its use heavily depends on what is being encrypted. Consider a text message encrypted using permutation of size 128 bytes (bytes are permuted). There are 128! ways to order 128 objects. But in text we have some letters that appear many times in a block of size 128 and frequency of vowels is much higher than consonants. There are many such properties of text which makes the attach much easier (most words are 3-8 characters long, words are delimited with space). But video data is compressed and does not exhibit as many hints that can be used for an attack. Theoretically permutation is secure if data sources are stationary and memoryless [6]. Compressed video data exhibits more random behavior than uncompressed video (otherwise there is some correlation and it can be compressed even further. The crucial question at this point is how close compressed video sources are to being stationary and memoryless. To answer this question we used some MPEG files which has been heavily used for research purposes. We investigated the distribution of bytes and distribution of double-bytes. If these are close to uniform then the data source is close to being stationary and memoryless. We are not aware of any tests that measure how close a distribution is to being memoryless and stationary. We used the MPEG files flower.mpg and bus.mpg but the same can be observed with all MPEG files. Byte and double-byte distribution for these files are given in figure 5 and 6 respectively. As can be seen from the figures, compressed video data is close to being stationary and memoryless.

Attacking the (P-C) algorithm by generating all the (P-C) pairs is not practical due to the total number of such functions. Since MPEG has a well-defined structure one possible attack that comes to mind is try to eliminate some (P-C) pair if the MPEG player fails on a prefix of that pair. To see if such attacks are feasible or not we encrypted flower.mpg and bus.mpg except for the header (note that in our scheme attacker does not have access to header) and played it using the mpeg player. We used a block size of 256 bytes. Given a encrypted MPEG file, the MPEG player gives core dumps, bus errors or floating point exceptions, We selected 10,000 random (P-C) pairs and recorded the bytes where MPEG player gave an error. The results are in Figure 7. The x-axis shows the byte number of the error and y-axis shows the number of experiments in which the player failed before reaching that byte. As can be seen from the figure almost all the time the first encrypted block is parsed without detecting any errors, thus, removing bad prefixes does not necessarily help. To have intuitive idea of why this is the case, consider an arbitrary binary huffman tree and try to decode any string of length n. Whenever you reach a leaf, you start decoding from root so you only recognize an error at the end if the process does not end at a leaf node.

The proposed encryption algorithm is as follows. First a connection is established between video server and client and the (P-C) to be used and initial header and several extra bytes of the MPEG stream are transmitted using some other encryption algorithm such as DES. The reason for this is that the initial segment of the MPEG stream has some fields that can be guessed easily. The size and width of the video for example. The size of this initial video segment will be roughly 160 bytes. Another reason is that for establishing a connection, a standard set of steps needs to be followed and data sent during these steps can be guessed easily if permutation is used because some fixed message type needs to be used.

In the remainder of this section we describe the applications of the lemmas given in previous section.

Lemma 1 gives the expression for the number of error preserving functions. This is important because, an attacker can enumerate all such functions to find the function used. $n! * 2^n$ is a fast growing function and if n is chosen property, such an attack is not feasible. One important point is how can an attacker enumerate only error preserving functions, is there a simple way to do it? We propose a way to enumerate such functions in lemma 2.

Lemma 2 shows that all the error preserving functions can be enumerated using a permutation followed by complementation of a subset of bits. We need a way for doing this so that we can select such a (P-C) pair before transmission. Lemma 2 also gives a practical way to implement such functions using a computer. What we need to do is generate a random permutation, generate a random subset and permute and complement. Since operations on bits are costly to manipulate on a computer, for video, we can define these operations on bytes. Implementing permutation can be easily and efficiently done using table look-up. Implementing complementation of bytes is easy since many programming languages offer such operators.

Double encryption [13] is proposed as a way to improve security of a block encryption algorithm. It is based on encrypting twice with 2 different keys. Decryption is the reverse. Lemma 3 shows that for error preserving encryption double encryption does not help since there is a (P-C) that is equivalent to it.

Lemma 4 can be used for secure wireless video multicast in an ad-hoc network where users can be expelled from the group. Assume each node forms a group with its children. Construct a multicast tree, and do the multicast as follows. Root encrypts the video using permutation-complementation (P-C) algorithm and sends it to its subtrees. Each children by Lemma 5 uses the (P-C) equivalent to decrypting its parent's encryption and encrypting with its (P-C) to encrypt the message and sends it to its children. Note that in this scheme, when a user is expelled only the parent of that user needs to change the (P-C) and send it to its children.

Triple encryption [13] is another way to improve the security of a block encryption algorithm. Here we focus on version with 3 different keys, similar argument is true for different versions of triple encryption. Encryption is done by encrypting with first key, decrypting with second key and encrypting with third key. Decryption is done by decrypting with third key, encrypting with second key and decrypting with third key. Lemma 5 shows that triple encryption is equivalent to a single encryption and therefore does not improve security.

Some of the permutations such as the ones which keep majority of the bits in their original positions are vulnerable. We eliminate some of the permutations from the (P-C) algorithm to improve security. We limit ourselves to the permutations which map none of the bits to its original position. Such permutations are called derangements. Lemma 6 gives the expression for number of derangements. Number of derangements converges to $n!/e$ as n tends to infinity. So, we still have a large set of error preserving functions to work with. This is good since it makes brute-force attack impractical.

Video-on-demand servers may server hundreds of users, for such systems we want to be able to generate (P-C) pairs with minimum probability of collision. In order to achieve this, for each new user we can randomly select a derangement, a subset of bits for complementation and send it to user. Lemma 7 gives the probability that no collision occurs for k users. For n= 10 and k = 1000 second expression gives 0.99999994520261 which means that probability of collision is very low. For k = 10000 the expression is 0.99999452027058. Note that second expression in lemma is given for ease of computation, it is a lower bound. Actual value will be higher. And in practice, value of n will be much higher than 10.

Running times of permutation, P-C for block of 256 bytes with operations limited to bytes is given in Table 1. Selected files are commonly used in research community. As shown in the table, the permutation based algorithm is about 10 times faster than *gpg*, and (P-C) algorithm is slightly slower than permutation and about 9-10 times faster than *gpg*. Algorithm in [9] will be about 2 times faster than gpg since it encrypts about half the data, but it is not an error preserving algorithm. These results were taken using a 400 MHz PC with 256 MB memory running the Linux operating system. Our purpose here is to show the ratio of running times of the two programs. Using a mobile device the ratio should be similar.

5. CONCLUSION

In this paper, we described a theory for which to build error preserving video encryption algorithms. For wireless video transmission error preserving video encryption algorithms are needed to prevent loss of data. Our scheme takes advantage of the fact that most multimedia applications are somewhat tolerable to bit errors. We have shown that all error preserving encryption algorithms can be implemented by permuting bits and complementing a subset of the bits, double and triple encryption does not improve the security of the system. We have given examples of applications where such encryption is desirable. Error preserving encryption algorithms have the additional advantage that they can be implemented very efficiently on a computer and are typically 9-10 times faster than gpg. This is important since wireless devices have limited processing resources and power. Since video decoding uses most of the processing power, efficient mechanisms are needed for encryption. Our results show that, error preserving encryption algorithms have many desirable properties for wireless video transmission.

6. REFERENCES

[1] Coding of moving pictures and associated audio. In *Committee Draft Standard ISO11172:ISO/MPEG 90/176.*

[2] Richard A. Brualdi. *Introductory Combinatorics.* Prentice-Hall, 1999.

[3] Le Gall D. Mpeg: A video compression standard for multimedia applications. *Communications of the ACM*, 34(4):46–58, April 1991.

[4] G. Feistel. Cryptography and computer privacy. *Scientific American*, 5(228):15–23, 1973.

[5] John B. Fraleigh. *A First Course in Abstract Algebra.* Addison-Wesley, 1982.

[6] A. Sgarro G. Longo, M. Marchi, editor. *Geometries, Codes and Cryptography.* Springer-Verlag, 1990.

[7] Agi I. and Gong L. An empirical study of secure mpeg video transmissions. In *NDSS'96*, pages 137–144, 1996.

[8] J. B. Kam and G. I. Davida. Structured design of substitution-permutation encryption networks. *IEEE Transactions on Computers*, 28(4).

[9] Qiao L. and Nahrstedt K. A new algorithm for mpeg video encryption. In *CISST'97.*

[10] Qiao L. and Nahrstedt K. Comparison of mpeg encryption algorithms. *International Journal on Computer & Graphics*, 22(3), 1998.

[11] Tang L. Methods for encryting and decrypting mpeg video data efficiently. In *ACM Multimedia'96.*

[12] J.P. Pieprzyk. Error propagation property and application in cryptography. *IEE Proceedings Part E*, 4(136):262–270, 1989.

[13] Bruce Schneier. *Applied Cryptography.* John Wiley and Sons, 1996.

[14] Maples T.B. and Spanos G.A. Performance study of a selective encryption scheme for the security of networked, real-time video. In *ICCC'95*, 1995.

[15] Ali Saman Tosun and Wu chi Feng. The effect of bit errors on mpeg video quality for wireless networks. In *ISAS'99.*

File	*Size*	*perm. (sec.)*	*P-C (sec)*	*gpg (sec.)*
bus.mpg	718464	0.0622	0.0793	0.7666
flower.mpg	719510	0.0683	0.0848	0.7675
twister.mpg	401371	0.0378	0.0452	0.4751
tenis.mpg	1246001	0.1181	0.1277	1.2686

Table 1: Timings for gpg and permutation and P-C

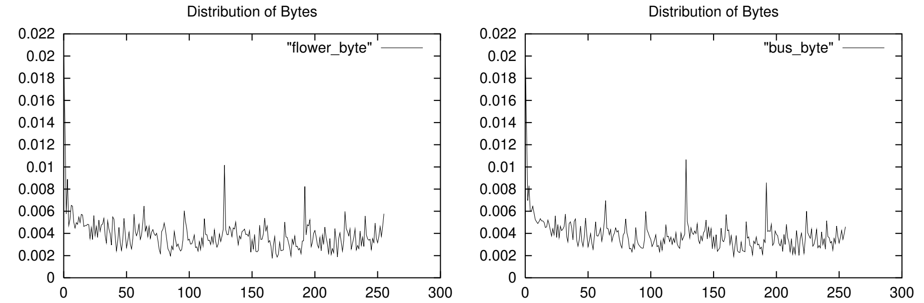

Figure 5: Byte distribution for flower.mpg and Bus.mpg

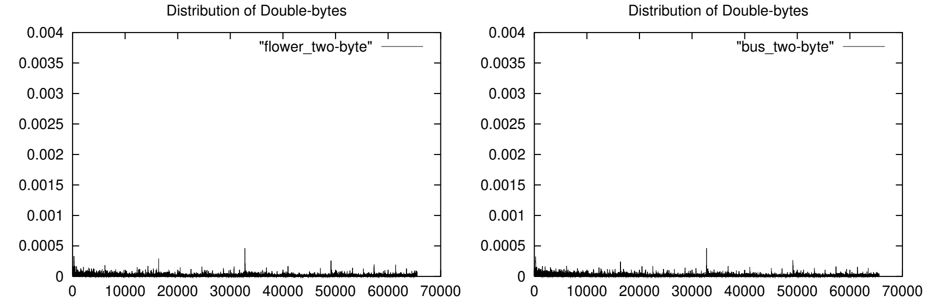

Figure 6: Double Byte distribution for flower.mpg and Bus.mpg

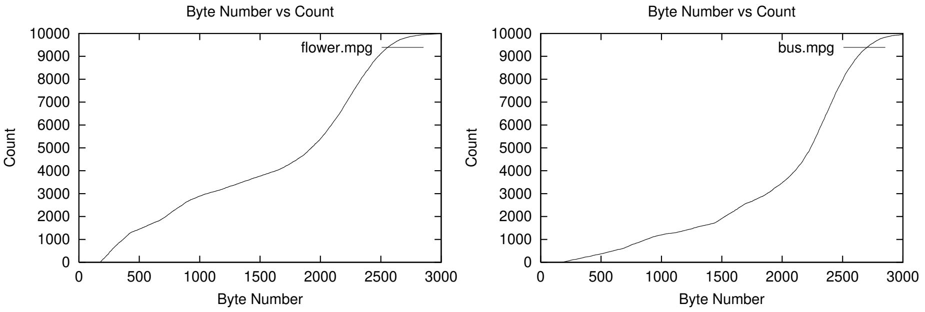

Figure 7: Byte no vs number of experiments which failed before that byte