A Video Scrambling Technique Based On Space Filling Curves

(Extended Abstract)

Yossi Matias and Adi Shamir

Department of Applied Mathematics, The Weizmann Institute of Science
Rehovot 76100, *Israel*

Abstract

In this paper we propose a video scrambling technique which scans a picture stored in a frame buffer along a pseudo-random space filling curve. We describe several efficient methods for generating cryptographically strong curves, and show that they actually decrease the bandwidth required to transmit the picture.

1. Introduction.

Video signals play a predominant role in modern society in diverse applications such as entertainment, telecommunications and military surveillance. In many applications it is essential to guarantee the privacy of this signal. However, the standard cryptographic techniques may be inadequate for three basic reasons :

1) The transmitted signal is analog.

2) The transmission rate is very high.

3) The allowable bandwidth is limited, and can be accommodated only when the grey levels of adjacent pixels are highly correlated.

Wyner [W1,W2] suggested a method of scrambling a discrete time analog sequence by using a large family of linear orthogonal invertible transformations which he described, that result in a negligible expansion of bandwidth. However, these transformations are not easy to compute, and in the case of video signals they do not exploit the two dimensional nature of the signal.

A video picture is usually transmitted by scanning its elements (pixels). The frame is scanned from top to bottom and from left to right; this is the conventional raster scan. Our approach is to effect a pixel permutation by changing the scanning order without changing the pixels' values. The scanning order is determined by the encryption key, and should be changed occasionally (but not necessarily every frame). Since we want to preserve the correlation between the grey levels of adjacent transmitted pixels, we propose to scan the picture with a connected curve i.e. a Space Filling Curve (SFC).

The encryption method is as follows :

a. Pseudo-randomly choose a Space Filling Curve.

b. Transmit an analog signal that represents the grey levels of successive pixels along this curve.

The corresponding decryption method is :

a. Choose the same SFC (by using a shared cryptographic key).

b. Fill a frame buffer with the received signal by following the SFC.

We can use the interframe redundancy and generalize our scheme to 3 dimensions. This is done by storing several consecutive frames in several frame buffers and finding a random SFC for the 3 dimensional collection of frame buffers. The use of a 3-dimensional SFC increases the number of possible SFCs by many orders of magnitude, increases the confusion of a would be attacker and further improves the signal's bandwidth, but increases the cost and complexity of the implementation.

Let G be the infinite graph whose vertex set consists of all points of the plane with integer coordinates and in which two vertices are connected if and only if the (Euclidean) distance between them is equal to 1. A *grid* graph is a vertex induced subgraph of G. Let v_x and v_y be the *coordinates* of the vertex v. A *rectangular grid* graph $R(m,n)$ is a grid graph whose vertex set is $\{v : 1 \leq v_x \leq m \text{ and } 1 \leq v_y \leq n\}$.

We shall consider the frame-buffer as a rectangular grid graph $R(m,n)$, whose vertices are the pixels. It follows that the Space Filling Curve is a Hamiltonian path in $R(n,m)$. The problem of finding Hamiltonian paths between specified pairs of vertices in rectangular grid graphs was examined by Itai et al [IPS], who showed how to construct one path. Our goal is to find many Hamiltonian paths, and we do not require specific endpoints. The ideal, of course, is to have an efficient algorithm which produces all Hamiltonian paths. However, a large family of Hamiltonian paths may be sufficient if we are careful about its cryptographic quality. To see what can go wrong, consider the following algorithm for generating Hamiltonian paths:

Algorithm A.

- Start at the upper left corner of the rectangle and move to the right.

  Repeat until all vertices are visited :

- Move straight ahead until visiting all unvisited vertices in this direction. At the last vertex decide (by the next pseudo-random bit) : either turn a 90° turn, or make a U-turn (towards unvisited vertices).

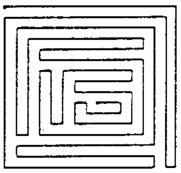

Figure 1. An example of a SFC generated by Algorithm A in a 16 *by* 16 *rectangular grid.*

Algorithm A is very efficient, it generates a large family of $\binom{n+m-2}{n-1}$ Hamiltonian paths, and it can be easily implemented. A typical scanning order produced by this algorithm is given in *Fig.* 1. However, this cryptosystem is easy to break with a ciphertext-only attack. Notice that we have long lines and turning points of two kinds: one is the 90° turn, and the other is the U-turn (180 degrees turn). Adjacent vertices from two lines which are connected by a U-turn are highly correlated. In order to locate the points where U-turns have been taken, it is sufficient to test for local symmetry points along the transmitted path **i.e.** for segments whose values are almost palindromes. Finding these points is quite easy, and thus breaking the cryptosystem is straightforward.

We conclude the introduction with a remark about the number of SFCs U_N in a square lattice of N points. By Orland et al. [OID] we have (for large N)

$$U_N \sim \begin{cases} 1.4715^N = 2^{0.5573N} & \text{in 2D square lattice} \\ 2.2073^N = 2^{1.1423N} & \text{in 3D cubic lattice.} \end{cases}$$

This demonstrates the advantage of using a 3-dimensional SFC. The confusion of a naive attacker is enlarged by a factor of

$$\frac{U_N^{(3D)}}{U_N^{(2D)}} = 1.5^N = 2^{0.585N} \quad (!!!)$$

which is more than 2^{150000} for $N = 512 \times 512$.

2. Tile and Merge

The method of producing Hamiltonian circuits discussed in this section is based on the concept of first *covering* all vertices by disjoint elementary circuits, and then *merging* these circuits into a single Hamiltonian circuit. Since our aim is to produce many Hamiltonian circuits we shall introduce randomness both in the *covering* and in the *merging* processes. While it is easy to produce all the possible *mergings* of a certain *cover* of a grid graph using the concept of spanning trees, the production of every possible *cover* (with a probability greater then zero) seems to be hard.

By using the tile-and-merge method we introduce efficient algorithms for producing exponentially many Hamiltonian circuits in a rectangular grid graph. A large subset of these Hamiltonian circuits can be encoded by only 1 bit per pixel. The algorithms can be easily extended to 3D.

2.1 The General Method

Let G be an undirected graph. Consider disjoint circuits $C_1, C_2, \ldots, C_k$ that cover all the vertices of G. Define two circuits C_i and C_j to be *neighbors* if there is an edge $e_1 = (v_1, v_1')$ in C_i , and an edge $e_2 = (v_2, v_2')$ $(= (v_2', v_2)$ $)$ in C_j such that there are edges $f_1 = (v_1, v_2)$ and $f_2 = (v_1', v_2')$ in G. Note that f_1 and f_2 complete e_1 and e_2 into a square. If C_i and C_j are neighbors then we can connect them into one circuit C_{ij} which covers exactly the same vertices as C_i and C_j by replacing e_1 and e_2 by f_1 and f_2. Denote this replacement as a *legal replacement* and denote (e_1, e_2) as a *legally replaced couple (LRC)* in (C_i, C_j). Note that between 2 circuits there may be several legal replacements but only one of them may be (randomly) chosen. Thus by a single legal replacement we reduce the number of circuits by 1. If we start with k circuits then after $k-1$ legal replacements we are left with a Hamiltonian circuit. This leads directly to the following definitions and algorithm for finding a random Hamiltonian circuit in a general grid graph.

Let G be a graph. Given k disjoint circuits $C_1, C_2, \ldots, C_k$ over G, define a graph $G_k = (V_k, E_k)$ where $V_k = \{C_1, C_2, \ldots, C_k\}$ and $E_k = \{(C_i, C_j) : C_i \text{ and } C_j \text{ are neighbors }\}$. For each $(C_i, C_j) \in E_k$ let L^k_{ij} be: $L^k_{ij} = \{(e_1, e_2) : e_1 \text{ and } e_2 \text{ are LRC in } (C_i, C_j)\}$. Note that in a grid graph, edges of a legally replaced couple are adjacent edges that are located in different circuits. Denote two circuits to be *external* if neither of them is surrounded by the other. The circuits $C_1, C_2, \ldots, C_k$ are called a *tile* of G. If the circuits are all external in pairs then we say that they are *external circuits* and we denote the tile as an *external tile.*

Algorithm SPAN

Given a grid graph G which has a Hamiltonian circuit:

1) Choose k disjoint external circuits $C_1, C_2, \ldots, C_k$ that **Cover** all the vertices of G s.t. G_k (as defined above) is connected.

2) Randomly choose a spanning tree for G_k.

3) For each edge (C_i, C_j) in the spanning tree randomly choose a legally replaced couple in L^k_{ij} and perform the legal replacement.

◇ ◇ ◇

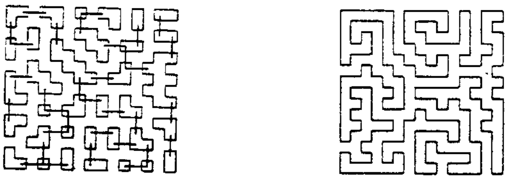

Figure 2. (a) An example of covering circuits and a spanning tree over them in a 16 by 16 rectangular grid. (b) The resulting Hamiltonian circuit.

THEOREM 1. The output circuit of algorithm **Span** is a Hamiltonian circuit in G.

Note that the fact that the initial tile is external is necessary for the correctness of the algorithm, since G_k may become disconnected during the execution of the algorithm when some of the circuits are internal.

Algorithm **Span** can be generalized so that it can find more Hamiltonian circuits after a *cover* was selected i.e. to enhance the randomness of the output circuit. This may be done by a slight modification of the algorithm with an insignificant degradation in efficiency. The main idea is to use an adaptive process that takes into account the new edges (in G_k) and LRCs which are created after each replacement of edges (in G).

If we could randomly choose all the combinations of elementary disjoint circuits that cover all the vertices in G then we would have a complete solution to the problem of finding all the Hamiltonian circuits, but this seems to be a hard problem. On the other hand there are some choices we can easily make. The most trivial choice is to make all C_i squares of 4 vertices (for the sake of simplicity we assume that the dimensions of the grid are even). The next natural choice is to make all C_i rectangles of 6 vertices. Note that in this case a 'legal replacement' as done in step 3 of **Span** can often be done in two ways; one of them should be randomly selected. A combination of the two choices results in an exponential number of tilings ($2^{\frac{mn}{5}}$).

2.2 Tile by Squares*

When G is tiled-by-squares only (the trivial choice), **Span** generates a very small subset of the set of all Hamiltonian circuits. Still, using the tile-by-square procedure **Span** constructs a set of Hamiltonian circuits with exponential cardinality. Moreover, the tile-by-squares procedure has a large advantage in coding the circuit. In addition, it can be conveniently analyzed and is the natural basis for extensions to more general graphs.

A simple and convenient method of encoding the scanning order is to add to each pixel the information of the next step in the scan. Thus, after arriving at a pixel, we can use this information and a Finite State Machine in order to know what is the next pixel in the scan. Obviously for any SFC we can locally use $\log_2 3 \approx 1.585$ bits per pixel (on the average) to represent *Forward, Left*

* In the beginning of our research Ron Rivest had communicated to us an idea of generating a Hamiltonian circuit in a $2n$ by $2n$ rectangular grid graph G in two steps:

1) Finding a spanning tree for the n by n rectangular grid graph, for which each vertex is the center of a square in G.
2) Surrounding the spanning tree using the edges of G.

Algorithm **Span** restricted to the tile-by-squares strategy produces exactly the same Hamiltonian circuits.

or *Right*. However, if we could save the information with only 1 bit per pixel then this could be significant in practical implementations.

CLAIM 1. The SFCs that are generated by algorithm **Span** using tile-by-squares can be (constructively) encoded by only 1 bit per pixel.

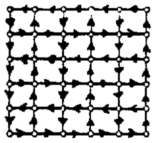

Figure 3. A 6 by 6 Manhattan orientation rectangular grid graph

Let G^M be a directed grid graph with a Manhattan orientation as in *Fig.* 3.

CLAIM 2. The set of Hamiltonian circuits that can be generated by algorithm **Span** restricted to tile-by-squares is exactly the set of Hamiltonian circuits in G^M.

Using Kasteleyn's exact enumeration [Kas] we have

COROLLARY 1. The number of Hamiltonian circuits that can be constructed by **Span** using tile-by-squares is $2^{0.4206334N}$.

Considering the cryptographic strength of the scheme we have

COROLLARY 2. Given a picture of black&white (horizontal or vertical) strips of one pixel width scrambled using a SFC, we can easily reconstruct the SFC that was used.

The SFC which is constructed by a tile-by-squares can be exposed by a single black&white pattern which is simple and therefore has a reasonable probability to occur (at least locally) in real transmissions. Thus, although the tile-by-squares technique, when used in algorithm **Span**, is economically attractive, it provides a much weaker scheme than the more general techniques which tile the plane with elementary circuits of various sizes and shapes.

3. The Ham+ Algorithm

In this section we describe an efficient algorithm which can produce all the Hamiltonian circuits of the rectangular grid $R(m,n)$ plus 'almost Hamiltonian' circuits that miss up to $\frac{n}{3}$ vertices in the $(m-1)$'th row. Every Hamiltonian circuit has a positive probability to be generated in each run, and in our application the almost Hamiltonian circuits are just as useful as the Hamiltonian circuits.

3.1 General Description

Algorithm HAM+ constructs the Hamiltonian circuit by scanning the rows of the array from top to bottom. We shall enumerate the rows from top to bottom and the columns from left to right. After the $(i-1)$'th iteration the rectangle $R(i-1,n)$ is completely covered with disjoint paths whose endpoints are all in the i'th row. In the i'th iteration the i'th row is filled by advancing the endpoints. No endpoint is left inside $R(i,n)$ and no path becomes a circuit. This is accomplished by letting the endpoints of each path identify each other as *partners*. In the final two rows we tie the paths together in order to get a global circuit, but in the process we may miss some of the vertices in the $(m-1)$'th row. We do not know how to complete the process without this additional degree of freedom.

When stationed at a vertex, we choose the edges connected to it in the circuit by choosing two of its neighbors. The idea is to pseudo-randomly choose between all possible choices in a way that would not prevent us from successfully generating the promised circuit. However, it is not guaranteed that all circuits have the same probability to be generated because when stationed at a vertex in the i'th row, the probability of an edge to be chosen should be dependent on the $(i-1)$ rows; this fact is not taken into consideration in the algorithm.

A full description of algorithm Ham+ (along with its non-obvious proof of correctness) will be given in the full paper.

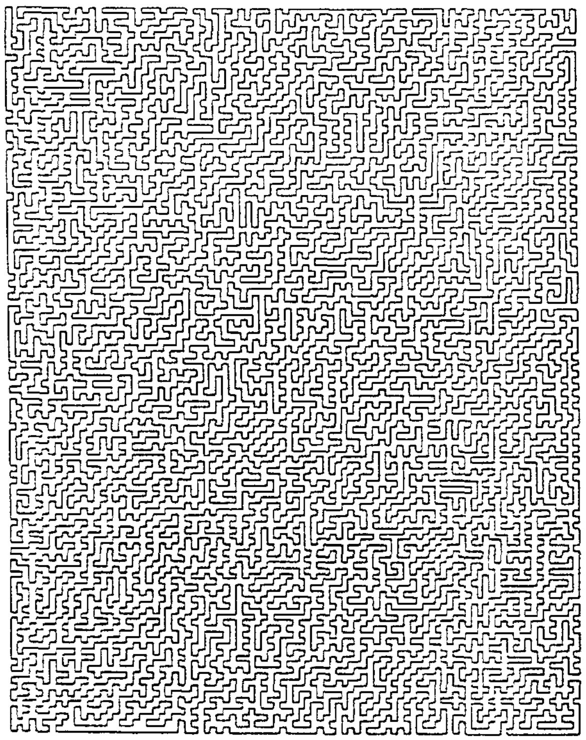

Figure 4. A circuit that was generated by Ham+
(note the 6 missing vertices at the last but one row).

3.2 Discussion

The number of missed vertices is $N_{missed} \leq \lfloor \frac{n}{3} \rfloor$. It is possible to reduce the number of missed vertices if we begin to tie the disjoint paths in the $(m-2)$'th or earlier rows. An interesting question that is left open is what is the minimal lookahead required to generate exactly the set of all Hamiltonian circuits?

In the application of HAM+ for video scrambling, the problem of missed vertices can be overcome by several techniques. For instance by transmitting the rectangle $R(m-2, n)$. The relative number of missed vertices $\frac{N_{missed}}{nm}$ is less then $\frac{1}{3m}$ (which is 0.065% for $m = 512$). Therefore the influence of the missed vertices on the overall performance is negligible.

The complexity of the algorithm is $O(mn)$, since each pixel is considered only once. The algorithm is efficient, it can be easily implemented and it can generate all Hamiltonian circuits plus extra circuits that can be used as well.

4. Autocorrelations of a Random 'Typical' Picture's Signals

We wish to examine the autocorrelation function $\varrho(k)$ of the scrambled signal and compare it with the autocorrelation $\rho(R)$ of the original signal. Let $p_k(R)$ be the probability that two points in the SFC that are k seperated have the Euclidean distance R in the grid and let E_k be the expectation with respect to $p_k(R)$. For a random picture of an isotropic stationary model (where the two-dimensional autocorrelation function depends only on the Euclidean distance between points) we have

$$\varrho(k) = E_k(\rho(R)) \tag{0.1}$$

for $k \ll \{$ the number of pixels in a line $\}$.

It was shown in [Ren] that

$$p_k(R) = A_k \left(\frac{R}{k^\nu}\right)^\theta \exp\{-\left(\frac{R}{k^\nu}\right)^\delta\} \tag{0.2}$$

where $\nu = \frac{3}{4}, \theta = 1.93 \pm 0.27, \delta = 4.6 \pm 0.06$ in two dimensions and $\nu = 0.59, \theta = 0.67 \pm 0.34, \delta = 2.6 \pm 0.06$ in three dimensions; A_k is a normalizing factor. Recall the form of the autocorrelation function of a (simplified) isotropic model (a first-order Markov process) [NL] $\rho(R) = e^{-\alpha R}$. Together with Eq. (0.1) and (0.2) it implies

$$\varrho(k) = A_k \sum_R \left(\frac{R}{k^\nu}\right)^\theta \exp\{-\left(\frac{R}{k^\nu}\right)^\delta - \alpha R\}. \tag{0.3}$$

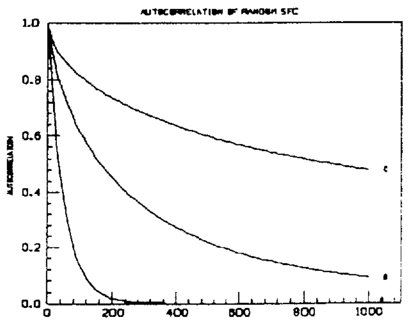

Figure 5.

In *Fig.* 5 we show the autocorrelation functions of a raster scan signal (A), a random two-dimensional SFC scan signal (B) and a random three-dimensional SFC scan signal (C). The higher autocorrelations in the SFC scan signals make it possible to reduce the bandwidth required to transmit the encrypted signal compared to the bandwidth required to transmit the original signal (or alternatively, for a given bandwidth, to gain a picture of better quality when its signal is scrambled before transmission). Unlike most other analog encryption schemes, our scheme actually compresses the signal. A related compression algorithm (which cannot be used as a cryptosystem) was recently proposed in [LZ].

5. Demonstrative Simulations

Faithful to the saying "seeing is believing" we applied our scheme to two pictures that are considered typical examples: A detailed *Landscape* picture and a *Head & Shoulder* picture. In order to simulate the effect of a low pass filter of $p\%$ on a transmitted video signal we applied a Fourier Transform on the (one dimensional) signal, zeroed all $(100 - p)\%$ high frequency coefficients, and performed an inverse Fourier Transform of the result.

Each picture was transformed into a signal using four scans: a (conventional) Raster Scan, two scans along different SFCs generated by **Ham+**, and a Random Scan in which the pixels in each row were randomly permuted. Each signal was filtered with low pass filters of $p\%$ with $p = 80, 60, 40, 20, 10, 5, 1$.

In *Fig. 8.a* we can see how the video signals of the Landscape picture, obtained by the four scans, are displayed on a screen when a descrambler is not used. The plain signal appears at the upper row of the figure. The signals that were obtained by the SFCs scans appear at the second and third rows of the figure. The signal that was obtained by the Random scan appears at the lower row of the figure.

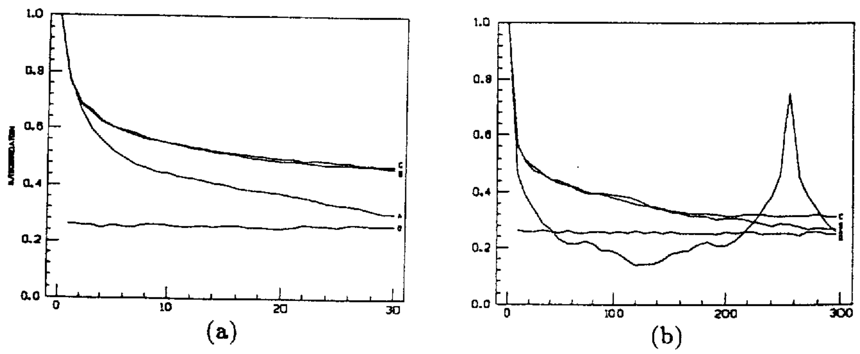

Figure 6. The autocorrelations of the Raster Scan signal (A), the SFCs' signals (B and C) and of the Random Scan signal in the Landscape picture in the range 0–30 (a) and 0–300 (b).

Fig. 6 shows the behavior of the autocorrelation functions in the ranges 0–30 and 0–300. The Raster Scan's function behaves as expected – it rises to a second peak at 256 due to the line-to-line correlation; refer to Franks [Fr]. The SFCs' functions continue the descent, demonstrating the loss of the line-to-line correlations; this loss occurs while on the other hand, there is an enhancement of correlation in the range 0-230 (in comparison with the Raster Scan's function). The Random Scan's function looks like a δ-function.

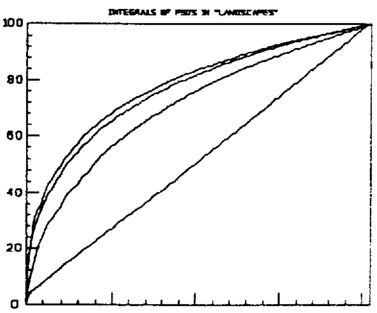

Figure 7. The integrals of the PSDs of the 4 signals.

The bandwidths of the 4 scans are demonstrated by showing the integrals I_ps of the PSDs. An I_p shows the amount of power that remains in a signal after a low pass filtering is applied; therefore, the higher the I_p, the better is the scan. *Fig.* 7 shows all 4 I_ps. The integral of the Random Scan's PSD is, as expected, a straight line. The Raster Scan's PSD is much better; it is the curve just above the Random Scan's line. However, it is not as good as the PSDs of the SFCs which are the two top curves.

The difference between the SFCs' I_ps and the Raster Scan's I_p is especially emphasized in the range of about 5–25% of the spectrum. This is the range where we expect the most significant improvements of performance of the scrambled signal in comparison with the original signal; i.e. we get a picture of better quality if we scramble it before transmission via a low-pass channel.

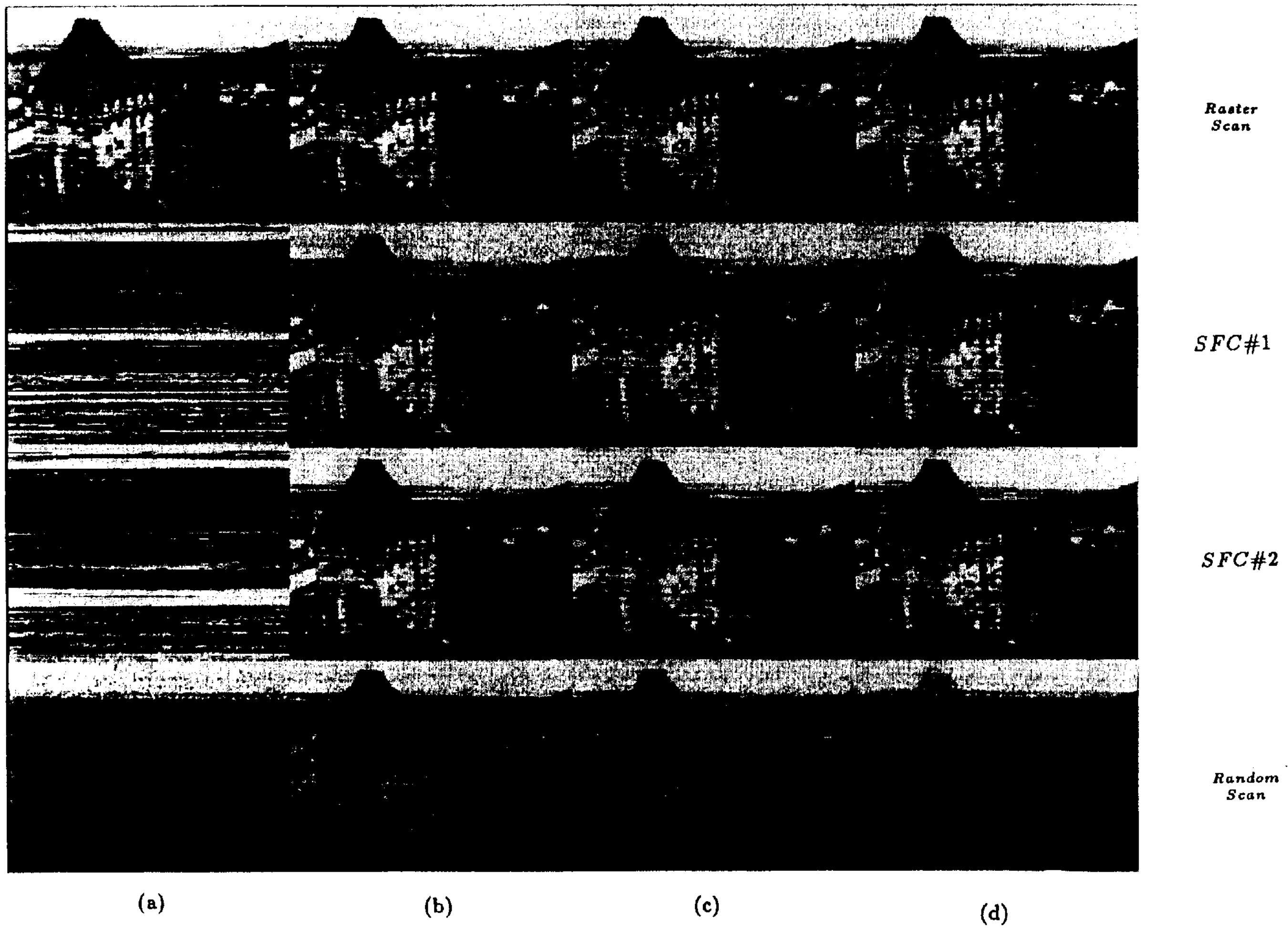

Figure 8. The 4 signals (a), the output of the 80%, 60% *and* 40% *filterings of the Landscape picture's signals (by the 4 scans) in (b), (c) and (d), respectively.*

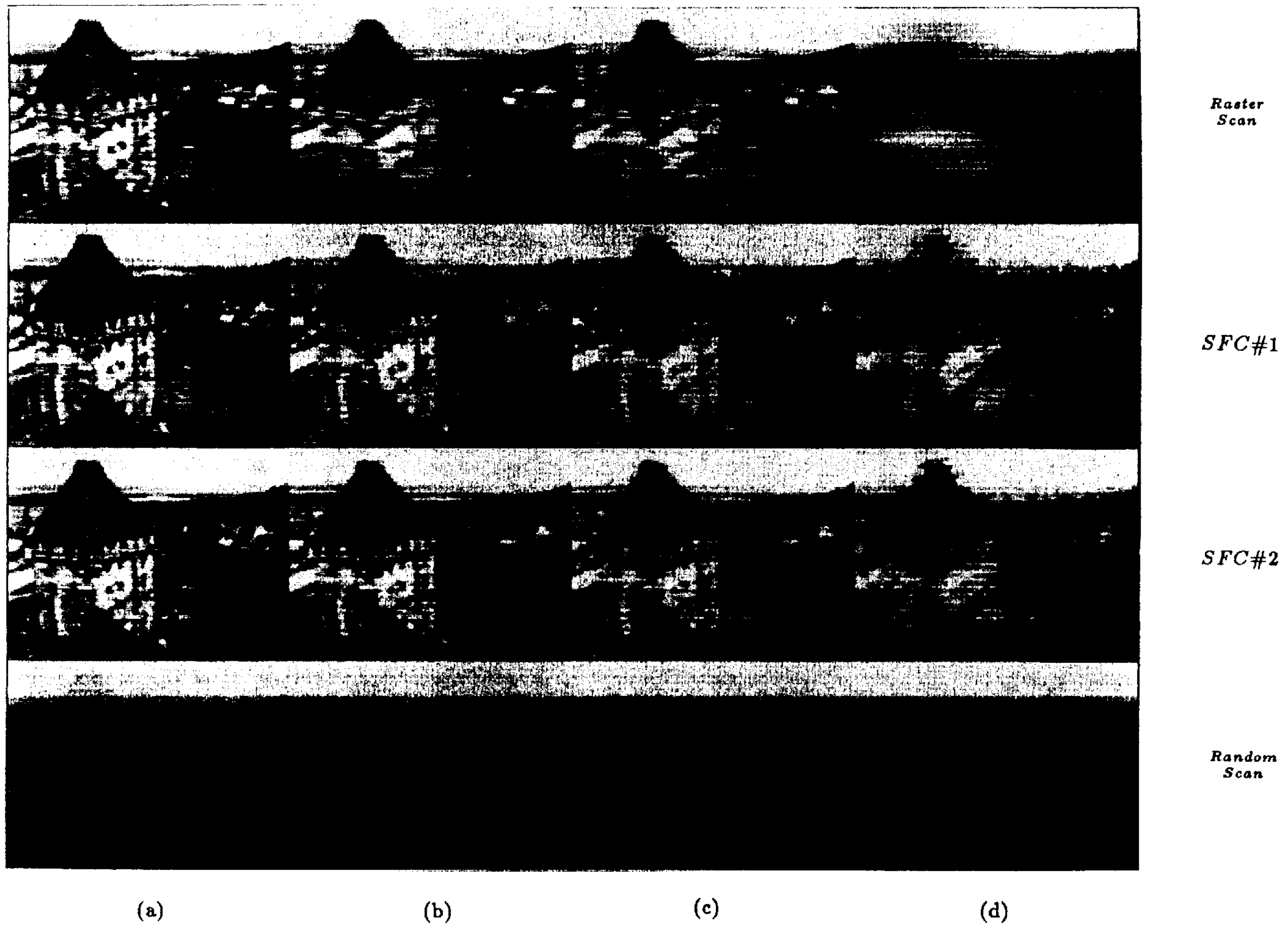

Figure 9: The output of the 20%, 10%, 5% and 1% filterings of the Landscape picture's signals (by the 4 scans) in (a), (b), (c) and (d), respectively.

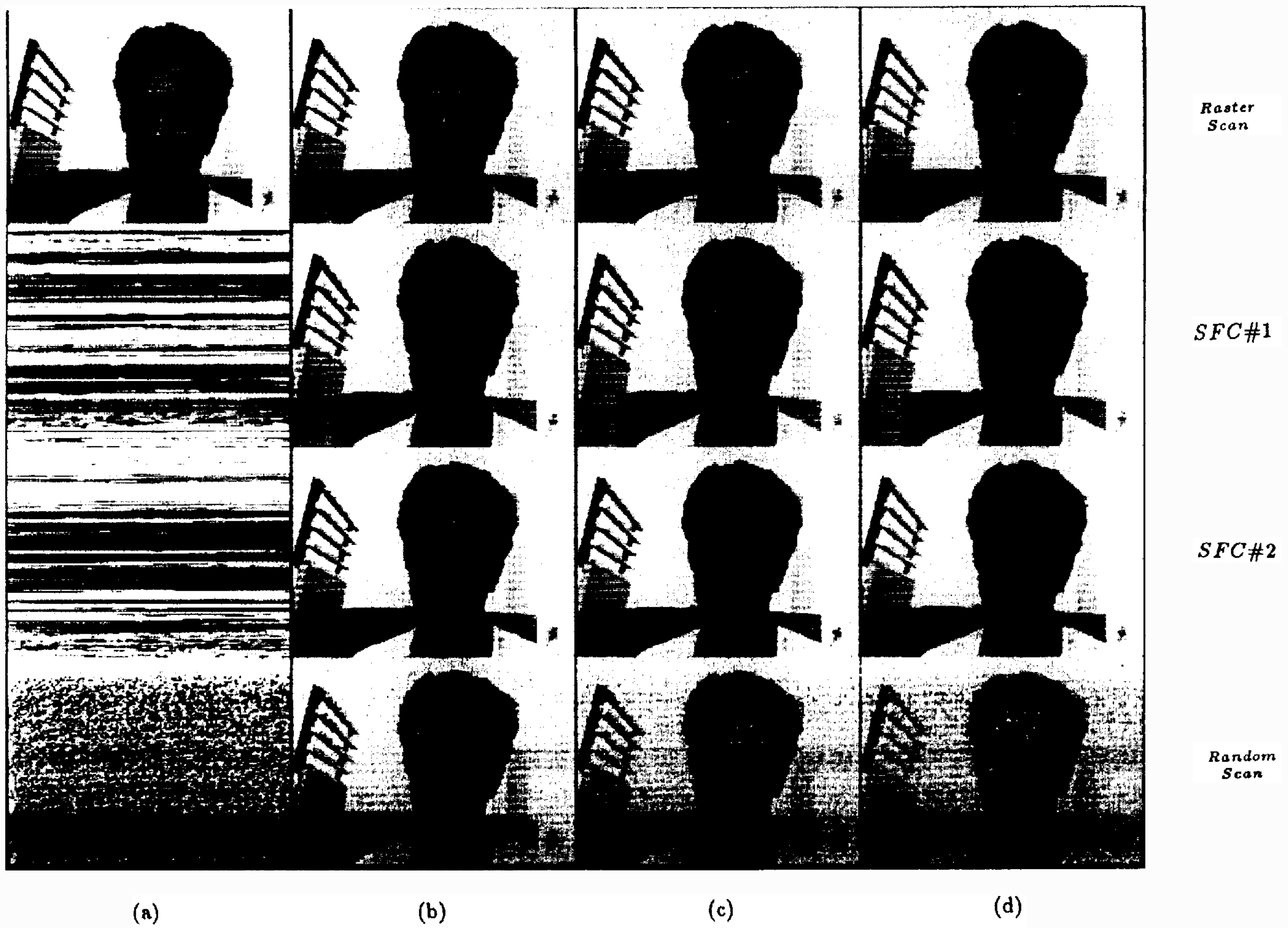

Figure 10. The 4 signals (a), the output of the 80%, 60% and 40% filterings of the Head & Shoulder picture's signals (by the 4 scans) in (b), (c) and (d), respectively.

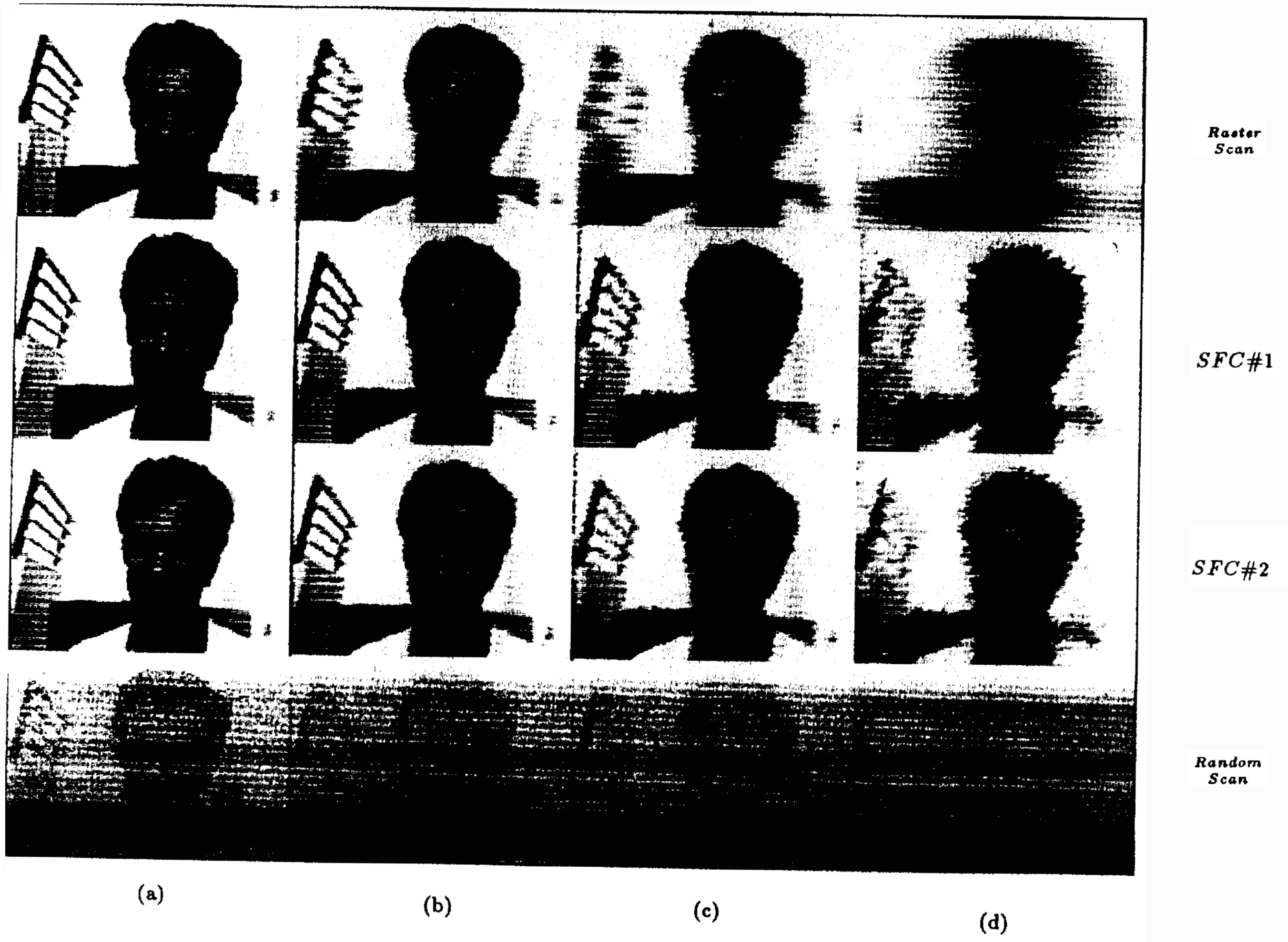

Figure 11. The output of the 20%, 10%, 5% and 1% filterings of the Head & Shoulder picture's signals (by the 4 scans) in (a), (b), (c) and (d), respectively.

In *Fig.* 10.a we can see how the video signals of the Head & Shoulder picture, obtained by the four scans, are displayed on a screen when a descrambler is not used. The filtered Head & Shoulder pictures are shown in *Fig.* 10 b,c,d and in *Fig.* 11.

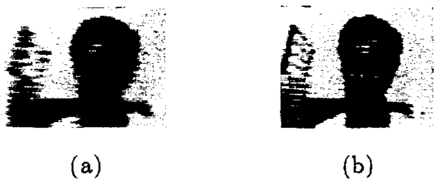

Figure 12. 32 pixels by 32 pixels 5% filtered Head & Shoulder pictures. (a) A Raster Scan's picture. (b) A SFC Scan's picture.

In order to create the effect of looking-from-a-distance we reduced the 5% filtering Head & Shoulder pictures of the Raster Scan and an SFC Scan to 32 pixels by 32 pixels pictures *(Fig. 12)*. Naturally, local disturbances are diminished. We see that the Raster Scan's picture is not recovered; we observe no improvement in the quality due to the reduction in size. On the other hand, the SFC's picture seem to have lost very little information.

Figure 13. 32 pixels by 32 pixels 1% filtered Head & Shoulder pictures. (a) A Raster Scan's picture. (b) A SFC Scan's picture.

The improvement of the signal by its scrambling is also demonstrated in *Fig.* 13 – the 1% filtering Head & Shoulder pictures of the Raster Scan and of an SFC scan, reduced to 32 pixels by 32 pixels pictures. In the Raster Scan's picture we can only guess that the shadow was originally a Head & Shoulder picture, while in the SFC's picture we can still observe some details.

Finally we give in *Fig.* 14 a closer look at a typical ciphertext produced by the SFC encryption scheme, and challenge the reader to guess what is the corresponding picture.

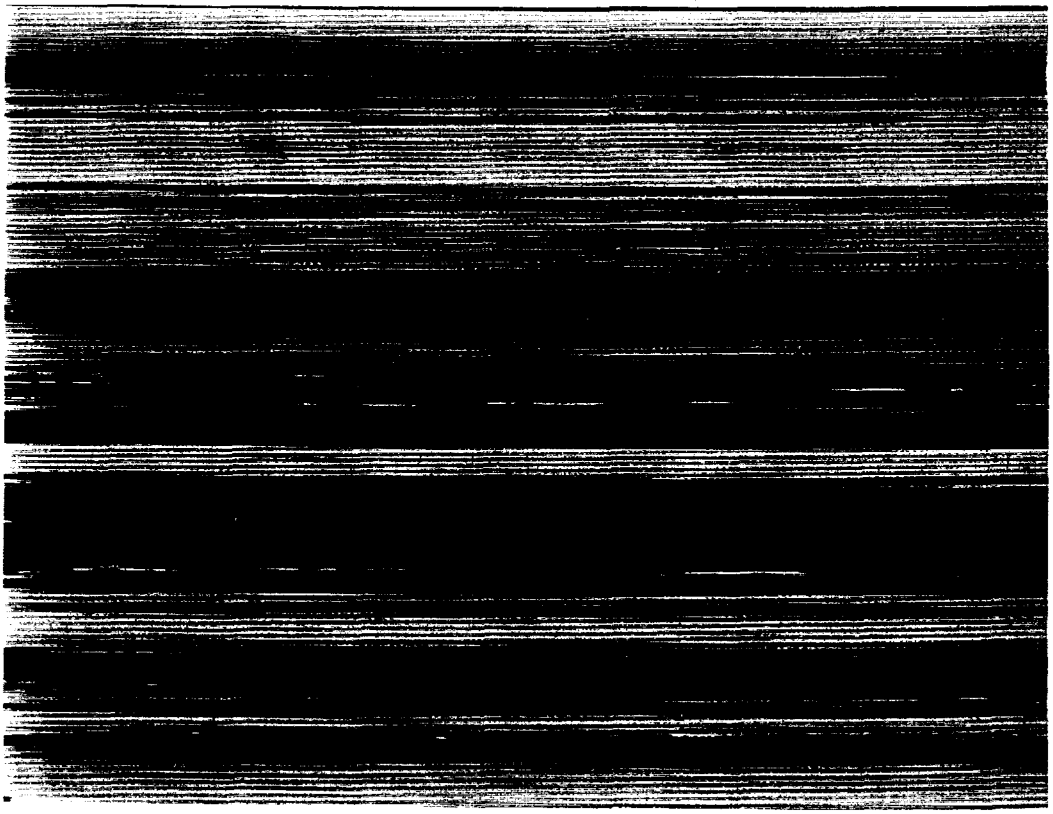

Figure 14.

References

[Fr] – L.E. Franks, *"A model for the random video process"*. Bell Systems Tech. J., 45 pp 609-630 (1966).

[IPS] – A. Itai, C.H. Papadimitriou and J.L. Szwarcfiter, *"Hamilton paths in grid graphs"*. Siam J. Comput. (1982).

[Kast] – P.W. Kasteleyn, *"Graph theory and crystal physics"*. in "Graph theory and theoretical physics", ed. F. Harary, Academic Press, London, 1967.

– *"A soluble self-avoiding walk problem."* Physica, 29, 1329-1337, 1963.

[LZ] – A. Lempel and J. Ziv, *"Compression of two-dimensional data"*. IEEE Trans. Inform. Theory, vol. IT-32, no. 1, pp. 2-8, Jan. 1986.

[NL] – A.N. Netravali and J.O. Limb, *"Picture Coding: A Review"*. Proc. IEEE, vol. 68, no. 3, March 1980.

[OID] – H. Orland, C. Itzykson and C. de Dominicis, *"An evaluation of the number of Hamiltonian paths"*. J. Physique Lett. 46, L-353-L-357, April 1985.

[Ren] – S. Render, *"Distribution functions in the interior of polymer chains"*. J. Phys. A: Math. Gen. 13, 3525-3541, 1980.

[W1] – A.D. Wyner, *"An analog scrambling scheme which does not expand bandwidth, Part 1 : Discrete time"*. IEEE Trans. Inform. Theory, vol. IT-25, pp.261-274, May 1979

[W2] – A.D. Wyner, *"An analog scrambling scheme which does not expand bandwidth, Part 2 : Continuous time"*. IEEE Trans. Inform. Theory, vol. IT-25, pp.415-425, July 1979