Analysis of the Nagravision Video Scrambling Method

Markus G. Kuhn*

9 July 1998

Abstract

The Kudelski Nagravision pay-TV conditional access system is widely used in Germany (Premiere) and France (Canal+). It can be practically broken by image processing algorithms that rearrange the lines of a field based on statistical properties of typical TV images. With some knowledge about the limitations of the scrambling hardware one can reconstruct the scrambled TV image in real-time without knowledge of the cryptographic secret stored in the subscriber smartcard.

Draft Technical Report $Id: nagra.tex,v 1.6 1998-07-09 18:33:02+00 mgk25 Rel $

1 Introduction

Pay-TV broadcasters employ conditional access systems to ensure that only TV viewers who have payed a subscription fee and who have in return received a decoder box can watch the TV channel. The Nagravision [1] conditional access system for PAL television developed by Kudelski SA, Cheseaux, Switzerland, is used for instance by the pay-TV broadcaster *Premiere* in Germany. Like with other hybrid video scrambling systems such as *EuroCrypt* [2] or *VideoCrypt* [3], a digitally encrypted control word is sent over the radio interface to the decoder in order to control the descrambling of an analog TV signal. The control word is decrypted in a smartcard and converted into

*University of Cambridge, Computer Laboratory, Pembroke Street, Cambridge CB2 3QG, United Kingdom. Email: mkuhn@acm.org

the seed value for a random number generator. This random number generator then controls the image descrambling process for the next few seconds. *Nagravision* scrambles the video signal by permuting the lines within a field. It also inverts the audio spectrum by mixing it with a 12.8 kHz sine wave to make it unrecognizable. The audio signal can trivially be descrambled by just inverting its spectrum a second time; it is not protected by any cryptographic mechanisms.

Like with all hybrid scrambling systems, which digitally control the scrambling of a video signal that is transmitted in analog form, there are two different classes of techniques for descrambling the video signal without using a regular decoder or smartcard:

- Microelectronics testing equipment can be used to extract the decryption algorithm and secret key data from the smartcard and with this knowledge compatible pirate smartcards and decoders can be manufactured [4].
- Statistical properties of typical TV images can be used to partially reconstruct the original image. From a partially reconstructed image the random number seed value that controls the descrambler is then reconstructed and used to descramble the entire image in high quality. This technique makes it unnecessary to break the digital cryptography or smartcard security aspects of the system and it can be implemented without using any genuine decoder hardware.

2 Video Scrambling

The B,G/PAL TV standard [5, 6], which is used for instance in Germany, displays 25 frames per second. Each frame is displayed as a sequence of two interlaced fields so that the screen image is updated with a rate of 50 fields per second. With B,G/PAL 15625 lines are displayed per second which leaves 64 μs per line. Around 52 μs of this line interval contain active line content; the remaining time is the horizontal blanking interval, which consists of a 1.55 μs front porch, a 4.7 μs sync pulse and a 5.8 μs back porch. Of the $15625/25 = 625$ nominal lines that are transmitted per frame in 1/25 s, the frame line number intervals 23–310 and 336–623 each contain the $288 = 2^9 + 2^6$ visible lines of one field. In the following, we will only talk about the lines within a single visible field and we will use field line numbers ranging from 0 to 287 to refer to these lines. The remaining $625 - 2 \cdot 288 = 49$

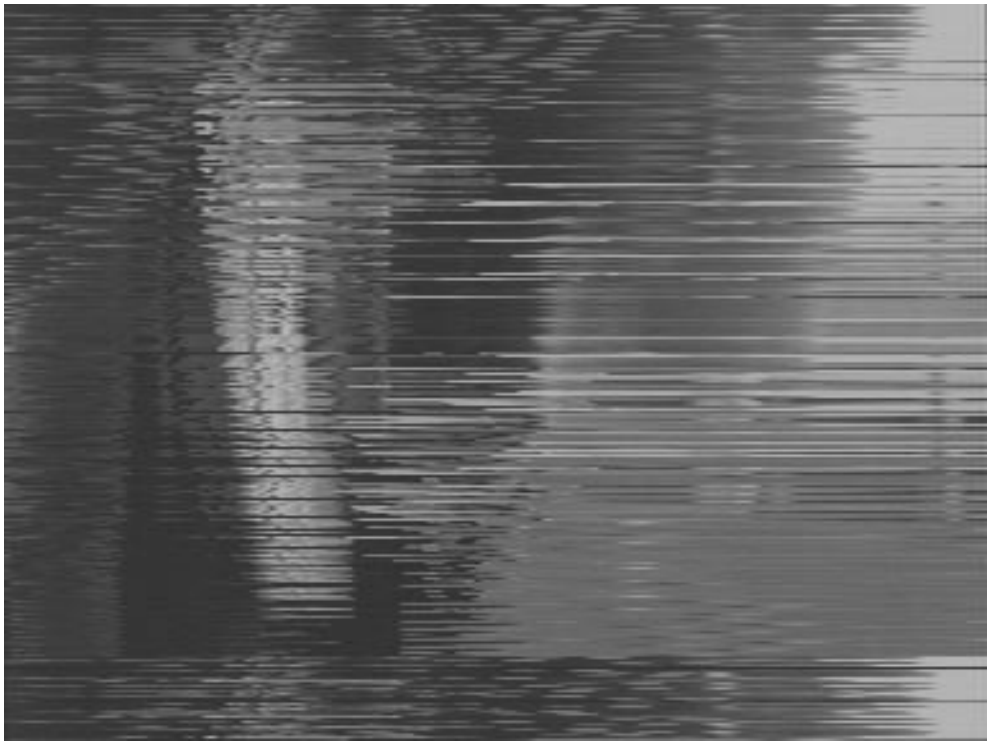

Figure 1: Example of a *Nagravision* scrambled image field.

lines of a frame do not contain a visible image signal and form the vertical blanking intervals. These are used for the vertical sync pulse and for digitally transmitted data such as videotext and control signals for conditional access systems.

Nagravision scrambles the image by permuting lines within a field. In addition, the boundaries between fields are shifted by 32 lines between the scrambled and descrambled image as Fig. 1 shows. This means that in the scrambled signal the last 32 lines (field lines 256–287) of one field and the first 256 lines (field lines 0–255) of the following field together form a group of 288 lines that are permuted and then used together to form a single field in the descrambled signal. This line scrambling can be described by a permutation function $p : \{0, \ldots, 287\} \to \{0, \ldots, 287\}$ which says that field line i in the descrambled signal appears as field line $p(i) - 32$ in the scrambled signal for all $0 \leq i \leq 287$, where a negative field line number $p(i) - 32 < 0$ refers to a field line number $288 + (p(i) - 32)$ in the previous field. The descrambling permutation p^{-1} is just the inverse function of p such that line $j - 32$ in the scrambled field can be found as line $p^{-1}(j)$ in the descrambled signal. The 4.43 MHz color burst signal in the back porch is not permuted.

Nagravision permutes the 288 lines that will form a field in the descrambled signal by buffering 32 lines in RAM and by writing and reading lines into and out of this buffer in a pseudo-random order. We shall refer to these 32 buffer lines as $B_0, \ldots, B_{31}$. While field line number i is received, the content of buffer $B_{v(i)}$ is sent out as the descrambled signal, and then buffer $B_{v(i)}$ will be immediately overwritten with the signal of the incoming line. The buffer selection function v has the form

$$v(i) = \begin{cases} S(u(i)), & \text{for} \quad 0 \leq i \leq 255 \\ i - 256, & \text{for } 256 \leq i \leq 287 \end{cases} .$$

Using v, we can write the descrambling permutation function as

$$p(i) = \max\{j \mid 0 \leq j < i + 32 \wedge v((j - 32) \bmod 288) = v(i)\}$$

or equivalently

$$p^{-1}(i) = \min\{j \mid i - 32 < j \leq 287 \wedge v(j) = v((i - 32) \bmod 288)\}.$$

$S : \{0, \ldots, 255\} \rightarrow \{0, \ldots, 31\}$ is a substitution table stored in non-volatile memory in the descrambler. It differs between the broadcasters and presumably can be updated over the radio interface from time to time, but it does not change frequently.

The function $u : \{0, \ldots, 255\} \rightarrow \{0, \ldots, 255\}$ depends on two frequently changing parameters $r \in \{0, \ldots, 255\}$ and $s \in \{0, \ldots, 127\}$. It has the form

$$u(i) = (r + it) \bmod 256 \qquad \text{with} \qquad t = 2s + 1.$$

The 15-bit seed value (r, s) is determined from the decrypted control word returned by the smartcard[1].

Since $t = 2s + 1$ has no common factor with 256, the function u is for all combinations of r and t a permutation on $\{0, \ldots, 255\}$. With $2^{15} = 32768$ possible combinations of r and t, there are 32768 different functions u and depending on the structure of S there are up to as many different functions v and p possible with a fixed substitution table S.

To learn more about the relationship between members of the set V_S of all functions v for a fixed S, we first have a look at the structure of the set U of all 2^{15} functions u. We define the following four relations on U:

[1] Does (r, s) change per field or just every few seconds? How is it calculated from the 64-bit control word returned from the smartcard?

- u is an *Eastern neighbor* of u' or u' is a *Western neighbor* of u iff $u'(i) = u((i+1) \bmod 256)$ for all $0 \leq i \leq 255$.
- u is a *Northern neighbor* of u' or u' is a *Southern neighbor* of u iff there exists a $j \in \{1, 3, \ldots, 255\}$ such that $u'(i) = u(ij(j+2)^{-1} \bmod 256)$ for all i.

Every function $u \in U$ has exactly one Northern, Eastern, Southern, and Western neighbor. These can be obtained by adding ± 1 to either parameter r or s of u. If we build a directed graph G_U using U as the set of nodes, where there exists an edge from any node to its four neighbors, then the resulting graph can be laid out as a 256×128 orthogonal grid on the surface of a torus using (r, s) as coordinates. G_U is characterized by the fact that in every cycle on it, the number of Eastern edges minus the number of Western edges is a multiple of 256 and the number of Northern edges minus the number of Southern edges is a multiple of 128.

With equivalently defined neighborhood relations, we can also build a corresponding graph G_{V_S} using the set V_S of functions v as the nodes. G_{V_S} can potentially have fewer nodes than G_U, since it is possible with certain pathological substitution tables S (such as $S(i) = 0$ for all i) that for two functions $u, u' \in U$ with $u \neq u'$ we have $S(u(i)) = S(u'(i))$ for all i, which means both u and u' in G_U correspond to a single node in G_{V_S}. It is also possible that for some S some nodes in G_{V_S} have more than one Northern, Southern, Eastern or Western neighbor. But in general for a randomly picked S, we expect G_{V_S} and G_U to be isomorphic.

The nature of the scrambling method restricts the permutation p by the condition $p(i) < i + 32$, which is equivalent to $p^{-1}(i) > i - 32$, and by the condition that the sequence $p(0), \ldots, p(287)$ can be split up into 32 monotonically increasing subsequences. Each of these subsequences corresponds to the sequence of lines that were stored in one of the 32 buffers, that is for any $0 \leq i < j \leq 287$ with $v(i) = v(j)$ we have $p(i) < p(j)$. This leaves $32^{288-32} = 2^{1280}$ possible permutations p if S is not known, compared to only 2^{15} possible permutations if S is known.

3 Attack techniques

The following techniques are based on the observation that in a typical TV image C, the correlation of two pixels drops quickly as the distance between these pixels increases. This means for instance that for two pixel luminosities

$C_{x,y}$ and $C_{x',y'}$, the average absolute difference $E(|C_{x,y} - C_{x',y'}|)$ is smaller or alternatively the normalized correlation $E(C_{x,y}C_{x',y'})/\sqrt{E(C_{x,y})E(C_{x',y'})}$ is larger if the two pixels are direct neighbors than if they are many lines apart. The permuted lines can be sorted back into an arrangement close to their original order just by rearranging them in a way that maximizes the similarity (correlation) between neighbor lines.

3.1 Reconstructing the Permutation

Let $C_{x,y}$ be the luminosity or even the whole three-dimensional color vector of the pixel (x,y) in the scrambled image. Then the matrix $K \in I\!R^{288\times 288}$ shall be the correlation matrix for a field, defined as

$$K_{i,j} = \frac{\sum_k C_{k,i}C_{k,j}}{\sqrt{\sum_k |C_{k,i}|^2 \cdot \sum_k |C_{k,j}|^2}}.$$

$K_{i,j}$ is a measure for the similarity of lines i and j. Obviously $K_{i,j} = K_{j,i}$ and $K_{i,i} = 1$, therefore we only have to determine $K_{i,j}$ for all $i < j$. Exchanging two lines i and j in the image C corresponds in K to swapping the contents of the lines i and j, plus swapping the columns i and j. The goal of rearranging the lines of C to form the original image corresponds to permuting lines and columns in K to bring the largest values as close as possible to the diagonal, so that we maximize the value of a profit function such as

$$G(K) = \sum_{i=0}^{286} K_{i,i+1}.$$

This corresponds to finding the permutation matrix P that maximizes the value $G(PKP^{\mathsf{T}})$. P relates to the permutation p that we want to reconstruct by $P_{p(i),i} = 1$ for all i and all other $P_{i,j}$ are zero.

In an alternative formulation of the same problem, we look at an undirected graph G_K with nodes N_i $(0 \le i \le 287)$, of which each corresponds to a field line $i - 32$ in the scrambled image. The edge connecting N_i and N_j in this graph has the value $K_{i,j}$ for all i and j. We then look for a Hamiltonian path (i.e., a path that visits all nodes exactly once) of the form $N_{p(0)}, \ldots, N_{p(287)}$, that fulfils the previously stated conditions for p and whose sum of edge labels is maximal. Finding such a path is a variant of the Traveling Salesman Problem [7, 8], which unfortunately is known to be NP-complete, although there exist a number of useful approximation algorithms.

3.2 Reconstructing the Substitution Table

One possible way of determining S is to reverse engineer a Nagravision decoder and read the entire table out of its non-volatile memory. Since this procedure might be illegal in some regions, alternative approaches are of interest. A logic analyzer can be used to record the sequence of accesses to the line buffers B_i, which results in a large collection of functions $v \in V_S$. If opening the decoder is also not acceptable for legal reasons, we can use a PC video adapter to perform a chosen cipher image attack in which we send to the decoder a test image that contains genuine encrypted control word information in the vertical blank interval and that uses a redundant binary code to mark every line with its field line number. We record the descrambled test image and when we read the line number markers in there, we get the permutation p^{-1}. If we do not have access to a Nagravision decoder or do not want to access one for legal reasons, we can also attempt to use one of the Traveling Salesman approximation algorithms to determine samples of p^{-1} from the correlation matrix of scrambled TV images alone as described in the previous section.

In both cases, we have to transform the observed permutations p^{-1} into buffer access functions v before we can extract S. This can be accomplished with the following simple algorithm, provided that p is error-free: We set $b_i := i$ for all $0 \leq i \leq 31$. Then for each of the first 256 lines $p^{-1}(j)$ with $0 \leq j \leq 255$ that the decoder outputs, we find the i for which $b_i = p^{-1}(j)$ and we set both $v(j) := i$ and $b_i := j + 32$. As a final check, we verify that after these 256 steps we have $b_i = p^{-1}(256 + i)$ for all $0 \leq i \leq 31$.

In this way, we collect a number of members of V_S. We search in this collection for neighbors and connect the available v functions to a subset of the graph G_{V_S}. For plausibility tests, we can try to arrange the available nodes of G_{V_S} to the torus surface grid that characterizes the topology of G_{V_S}. As soon as we have found a Southern neighbor v' for a node v, we know that v' has the parameter $t = j$ and v has the parameter $t = j + 2$, where j is the search parameter in the definition of Southern neighborhood. We can now easily determine a function $v_{t=1}$ that has the seed parameter $t = 1$ by using $v_{t=1}(i) = v(i(j + 2)^{-1} \bmod 256) = v'(ij^{-1} \bmod 256)$. We now have a potentially rotated substitution table $\tilde{S}$ in the form $v_{t=1}(i) = S(u_{t=1}(i)) = S((r + i) \bmod 256) = \tilde{S}(i)$.

We are not concerned about the cyclic shift caused by the unknown parameter r, because we just have to rename this unknown r to be our $\tilde{r} = 0$ and then $\tilde{S}$ can be used for a more efficient determination of the permutation as described in the next section. In this search, we are not interested in what r is used

inside the decoder, as long as our $\tilde{r}$ just generates the same functions u and v using a correspondingly rotated table $\tilde{S}$.

3.3 Realtime Determination of the Permutation Based on a Known Substitution Table

Once we know S either by extracting it from a *Nagravision* decoder or by determining it as outlined in the previous section, we can reverse the scrambling rather efficiently. A simple approach as implemented for instance in [9] is to perform a brute force search over all 2^{15} possible (r, t) tuples. For every possible (r, t) pair, the value of a penalty function is estimated by measuring the difference $|C_{x,p(y)} - C_{x,p(y+1)}|$ between a small number of randomly selected selected pixel pairs in the scrambled image that would under the tested (r, t) become neighbor pixels in the descrambled image. This can be implemented very efficiently since the permutation has to be performed only for the few test pixels and not for the entire image. We search for the (r, t) pair, for which the penalty function

$$H = \sum_{i=1}^{n} |C_{x_i,p(y_i)} - C_{x_i,p(y_i+1)}|$$

is minimal. The $(p(y_i), p(y_i + 1))$ pairs are precalculated for all 2^{15} (r, t) pairs for increased efficiency. Once this (r, t) pair has been identified, the corresponding permutation function is used to rearrange all lines in realtime.

A potentially much more efficient approach could be a binary subdivision search instead of a linear search over all 2^{15} possible (r, t) tuples. To implement this, we need a preparatory phase in which for a given substitution table S we build a binary decision tree. Each node in this decision tree lists a number of test pixel coordinates (x_i, y_i) and we branch to the left or the right subtree depending on whether H for these test pixels is above or below a threshold. Each leave of this tree is labeled with the (r, t) tuple that shall be used as the most likely candidate. A carefully built decision tree should be roughly balanced such that the maximum depth is not much over 15.

4 Remark

This paper is work in progress and quite likely contains errors and omissions. I started writing it in order to better understand the mathematical

properties of the *Nagravision* scrambling method and the algorithms used in the various currently available Nagravision pirate decoders that have been designed by anonymous developers. I also wrote it to collect and discuss possibly useful ideas and insights towards more efficient and more flexible attacks. An interesting problem is to find ways of determining the substitution table without access to any *Nagravision* decoder, since this allows the attack to automatically recover from potential encrypted uploads of new substitution tables. Since the Nagravision system is anyway scheduled to be replaced by a DVB conditional access system, I do not think that publishing my thoughts on the topic can do any economic harm, but I hope they might be of some educational benefit for anyone interested in pay-TV scrambling systems. Suggestions are very welcome.

Special thanks to Fabian Petitcolas for comments on the paper and to Ralph Metzler for providing frame grabber images for experiments.

References

[1] Andre Kudelski. Method for scrambling and unscrambling a video signal. United States Patent 5375168, 20 December 1994.

[2] Access control system for the MAC/packet family: EUROCRYPT. European Standard EN 50094, CENELEC, December 1992.

[3] Michael Cohen and Jonathan Hashkes. A system for controlling access to broadcast transmissions. European Patent Application 0 428 252 A2, 22 May 1991.

[4] Ross J. Anderson and Markus G. Kuhn. Tamper resistance – a cautionary note. In *The Second USENIX Workshop on Electronic Commerce Proceedings*, pages 1–11, Oakland, California, 18–21 November 1996.

[5] Television systems. ITU-R Recommendation BT.470, International Telecommunication Union, Geneva.

[6] Jim Slater. *Modern Television Systems – to HDTV and beyond.* Pitman, London, 1991.

[7] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms.* MIT Press, 1990.

[8] Eugene L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. *The Traveling Salesman Problem.* John Wiley & Sons, 1985.

[9] Gaston. NagraTV 2.00. Internet `http://eurosat.com/salp/gaston/`, June 1998. Linux open source software in C.

[10] John McCormac. *European Scrambling Systems 5 – The Black Book*. Waterford University Press, 1996. ISBN 1-873556-22-5.