Scientific Working Group on Digital Evidence

SWGDE Proposed Techniques for Advanced Data Recovery from Security Digital Video Recorders Containing H.264 Data

Disclaimer:
As a condition to the use of this document and the information contained therein, the SWGDE requests notification by e-mail before or contemporaneous to the introduction of this document, or any portion thereof, as a marked exhibit offered for or moved into evidence in any judicial, administrative, legislative or adjudicatory hearing or other proceeding (including discovery proceedings) in the United States or any Foreign country. Such notification shall include: 1) The formal name of the proceeding, including docket number or similar identifier; 2) the name and location of the body conducting the hearing or proceeding; 3) subsequent to the use of this document in a formal proceeding please notify SWGDE as to its use and outcome; 4) the name, mailing address (if available) and contact information of the party offering or moving the document into evidence. Notifications should be sent to secretary@swgde.org.

It is the reader's responsibility to ensure they have the most current version of this document. It is recommended that previous versions be archived.

Redistribution Policy:
SWGDE grants permission for redistribution and use of all publicly posted documents created by SWGDE, provided that the following conditions are met:

1. Redistribution of documents or parts of documents must retain the SWGDE cover page containing the disclaimer.
2. Neither the name of SWGDE nor the names of contributors may be used to endorse or promote products derived from its documents.
3. Any reference or quote from a SWGDE document must include the version number (or create date) of the document and mention if the document is in a draft status.

Requests for Modification:
SWGDE encourages stakeholder participation in the preparation of documents. Suggestions for modifications are welcome and must be forwarded to the Secretary in writing at secretary@swgde.org. The following information is required as a part of the response:

a) Submitter's name
b) Affiliation (agency/organization)
c) Address
d) Telephone number and email address
e) Document title and version number
f) Change from (note document section number)
g) Change to (provide suggested text where appropriate; comments not including suggested text will not be considered)
h) Basis for change

Intellectual Property:
Unauthorized use of the SWGDE logo or documents without written permission from SWGDE is a violation of our intellectual property rights.

Individuals may not misstate and/or over represent duties and responsibilities of SWGDE work. This includes claiming oneself as a contributing member without actively participating in SWGDE meetings; claiming oneself as an officer of SWGDE without serving as such; claiming sole authorship of a document; use the SWGDE logo on any material and/or curriculum vitae.

Any mention of specific products within SWGDE documents is for informational purposes only; it does not imply a recommendation or endorsement by SWGDE.

SWGDE Best Practices for the Recovery of Data from Security Digital Video Recorders

Table of Contents

Table of Figures

Note: Images in figures 1, 2, 3, 6, 7, 10, 11, 12 and Appendix B belong to the Forensic Audio Video Evidence Unit, Victoria Police Forensic Services Department. [1]

1. Purpose

The purpose of this document is to present advanced techniques for data recovery from security system digital video recorders (DVRs) when the data cannot be recovered using the guidelines provided in *SWGIT Section 24 Best Practices for the Retrieval of Digital Video* [2].

2. Scope

This document provides advanced recovery methods specific to security system DVRs storing video streams in the H.264 format. It is not intended for non-security DVRs such as those used for broadcast/cable television, gaming devices, and personal video recorders, nor has it been used to recover other video formats. Additionally, these methods should not be used by personnel not trained in the methods and techniques to which they refer. Traditional methods of recovery, such as those described in [2] or in traditional computer forensic capabilities should be attempted first. These advanced techniques should only be attempted after all other options have been exhausted.

3. Limitations

The documented technique has been shown to be successful in recovering video segments of data stored in the H.264 format only. Other video formats have not been tested.

The recovered data may require additional processing to portray the original recording properly, such as aspect ratio adjustments, frame rate adjustments, sequencing, and de-multiplexing.

Additional or alternate methods may be required to recover audio information and/or metadata. All video stored on the DVR may not be recovered through these methods.

4. Requirements

4.1 Case specific items

- Original DVR (if possible), or same make/model DVR, with power supply
- Forensic clone(s) and/or image(s) of the evidence hard drive(s)
- If data to be recovered is incident specific, request date/time of event and any known visual cues

4.2 Equipment

- Hardware write blocker
- Hex editor software (capable of viewing, editing, searching, carving, and concatenating)
- Video playback software (capable of decoding multiple video formats)
- (Optional) Software capable of extracting header data and/or assisting in demultiplexing

4.3 Technician/Examiner Qualifications

Personnel performing these advanced recovery techniques should meet technician/examiner level of training and experience as defined in the *SWGDE/SWGIT Guidelines & Recommendations for*

Training in Digital & Multimedia Evidence [3]. This should include experience in the use of a software hex editor and command line operations.

5. Recovery Methods

The flowchart in **Appendix A** provides an overview of the decision making process to follow in order to determine the best data recovery process a technician should utilize. The most difficult level of recovery, video frame recovery, is defined below.

5.1 Video Recovery, Targeted by Time and Date of Incident

The procedure described here is recommended when the requested examination is to recover footage from an incident at a specific time and date. In each case, the examiner will begin by attaching the previously prepared forensic clone to a write blocker, and loading the data contained on the clone into the chosen hex editing software.

5.2 Identify H.264 data

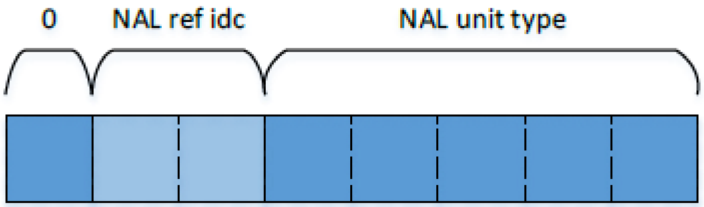
As stated in the limitations, this method is best suited for H.264 data. With slight modifications, this method may be applied to other video formats, but this is beyond the scope of this document.

H.264 data typically consists of a *byte stream* encasing a *Network Abstraction Layer unit* (NAL unit). The compressed data is contained within these NAL units. The first byte of a NAL unit is used to identify the kind of data contained inside.

Each NAL unit is preceded in the byte stream by a *zero byte* followed by a *start code prefix*. A zero byte is simply the byte 0x00 and a start code prefix is a sequence of three bytes, 0x000001. Therefore, each NAL unit should begin with the five byte sequence 0x00000001*ab*, where *ab* denotes a valid first byte for a NAL unit.

To read the first byte of a NAL unit consists of three parts, described in Section 7.3.1 of [4]. The first bit is called the *forbidden zero bit* and is always 0 for valid H.264 NAL units. The following two bits are the *NAL ref idc*. These bits can take any value (depending on the NAL unit type) so are of little diagnostic value. Higher values tend to indicate that the NAL unit is of higher importance. The final five bits contain the *NAL unit type*. The NAL unit type can take many values as detailed in Table 7-1 of [4], but the most likely values are summarized in the table below.

Figure 1. Format of first byte of a NAL unit



A simple way to quickly assess whether a NAL unit is valid and determine its contents which works for the majority of NAL units encountered in security DVR footage is to consider the two octets separately. Therefore, if the first byte is written 0x*ab*, then the first octet is *a* and the second is *b*. If the octet *a* is greater than seven, then it is not an H.264 NAL unit (as the forbidden zero bit is not zero). In most cases, the octet *a* will be an even value: 2, 4, 6 or less commonly, 0. If this is the case, then the octet *b* is equal to the NAL unit type and the contents for most NAL units can be determined by referring to the table below.

Figure 2. Common NAL unit

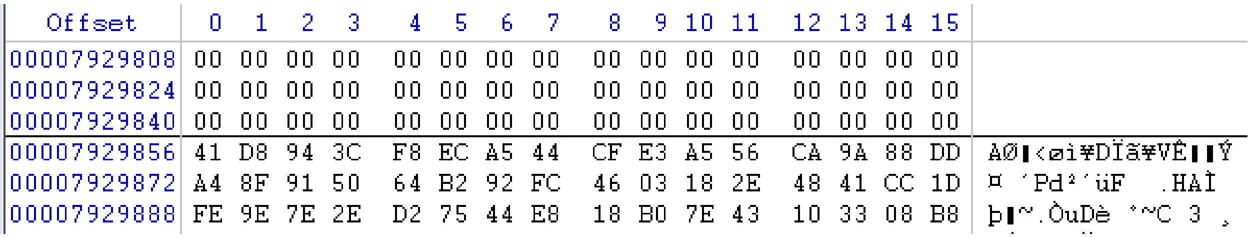
NAL unit type	Content
1	Coded slide of non-IDR picture (usually a P-frame)
5	Coded slide of an IDR picture (usually an I-frame)
6	Supplemental enhancement information
7	Sequence parameter set
8	Picture parameter set

If the video data is of type H.264, it should contain numerous valid NAL units. This can be determined using a hex editor. This is done by locating the video data, searching it for occurrences of 0x00000001 and then checking the following byte to ensure that it is a valid NAL unit.

5.2.1 Locating the start of the video data

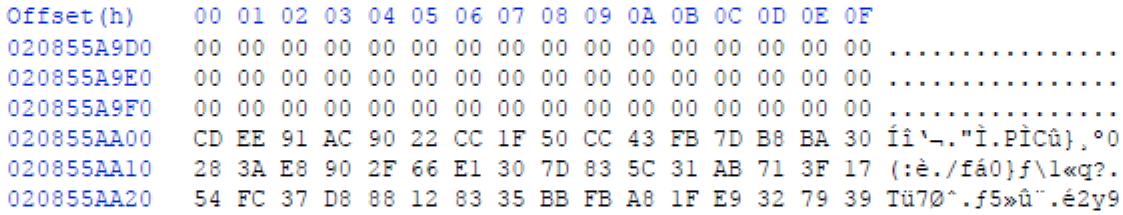
The video data should be compressed. Compressed data should be identifiable as predominantly having high entropy / low redundancy (that is, the data is unpredictable) as evident in the following example taken from a Swann brand security digital video recorder. Note that the NAL units may not be evident at the beginning of the video data, but may begin further into the drive.

Figure 3. Beginning of data area for Bunker Hill DVR



```
   Offset     0  1  2  3   4  5  6  7   8  9 10 11  12 13 14 15
00007929808 00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00007929824 00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00007929840 00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
00007929856 41 D8 94 3C  F8 EC A5 44  CF E3 A5 56  CA 9A 88 DD  AØ▌<øì¥DÏã¥VÊ▌▌Ý
00007929872 A4 8F 91 50  64 B2 92 FC  46 03 18 2E  48 41 CC 1D  ¤ ‘Pd²’üF  .HAÌ
00007929888 FE 9E 7E 2E  D2 75 44 E8  18 B0 7E 43  10 33 08 B8  þ▌~.ÒuDè °~C 3 ¸
```

Figure 4. Beginning of data area for Swann DVR



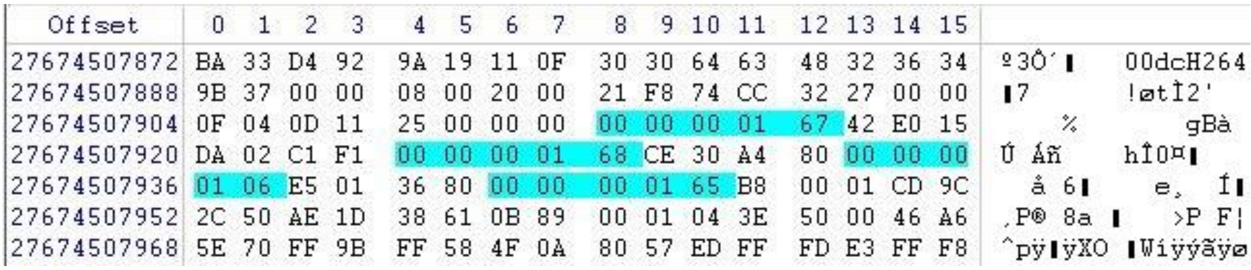
```
Offset(h)   00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
020855A9D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
020855A9E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
020855A9F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
020855AA00  CD EE 91 AC 90 22 CC 1F 50 CC 43 FB 7D B8 BA 30 Íî'¬."Ì.PÌCû},°0
020855AA10  28 3A E8 90 2F 66 E1 30 7D 83 5C 31 AB 71 3F 17 (:è./fá0}ƒ\1«q?.
020855AA20  54 FC 37 D8 88 12 83 35 BB FB A8 1F E9 32 79 39 Tü7Ø^.ƒ5»û¨.é2y9
```

5.2.2 Locate an I-frame and parameter sets

H.264 data found in security DVR footage typically consists of I-frames (intra-frame encoded frames) and P-frames (predictive encoded frames). B-frames would be unusual for a system which is recording "on the fly" as they would require the system to look ahead. I-frames are the most useful for diagnostic purposes, as they do not depend on any other frames.

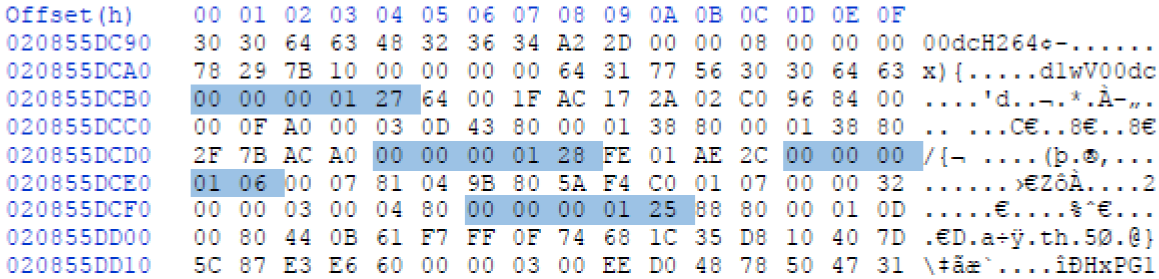
Each I-frame is usually encoded in a single NAL unit of type 5. In order to decode frames, both sequence and picture parameter sets are required (types 7 and 8). These are usually small NAL units located just prior to an I-frame. If a section of byte stream data is extracted from the first parameter set prior to an I-frame through to the end of the I-frame then tools like ffmpeg should be able to interpret and play or transcode this data. Again, this can be located by searching through the video data for the bytes 0x00000001 until the following byte indicates a NAL unit containing an IDR picture or a parameter set (see Figure 5 and Figure 6 below).

Figure 5. Video data with beginning of parameter sets and I-frame highlighted from zero byte (Bunker Hill DVR)



```
   Offset     0  1  2  3   4  5  6  7   8  9 10 11  12 13 14 15
27674507872  BA 33 D4 92  9A 19 11 0F  30 30 64 63  48 32 36 34  º3Ô´▮    00dcH264
27674507888  9B 37 00 00  08 00 20 00  21 F8 74 CC  32 27 00 00  ▮7       !øtÌ2'
27674507904  0F 04 0D 11  25 00 00 00  00 00 00 01  67 42 E0 15      %        gBà
27674507920  DA 02 C1 F1  00 00 00 01  68 CE 30 A4  80 00 00 00  Ú Áñ     hÎ0¤▮
27674507936  01 06 E5 01  36 80 00 00  00 01 65 B8  00 01 CD 9C    å 6▮     e¸  Í▮
27674507952  2C 50 AE 1D  38 61 0B 89  00 01 04 3E  50 00 46 A6  ,P® 8a ▮    >P F¦
27674507968  5E 70 FF 9B  FF 58 4F 0A  80 57 ED FF  FD E3 FF F8  ^pÿ▮ÿXO ▮Wíÿýãÿø
```

Figure 6. Video data with beginning of parameter sets and I-frame highlighted from zero byte (Swann DVR)



```
Offset(h)   00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
020855DC90  30 30 64 63 48 32 36 34 A2 2D 00 00 08 00 00 00  00dcH264¢-......
020855DCA0  78 29 7B 10 00 00 00 00 64 31 77 56 30 30 64 63  x){.....d1wV00dc
020855DCB0  00 00 00 01 27 64 00 1F AC 17 2A 02 C0 96 84 00  ....'d..¬.*.À–„.
020855DCC0  00 0F A0 00 03 0D 43 80 00 01 38 80 00 01 38 80  .. ...C€..8€..8€
020855DCD0  2F 7B AC A0 00 00 00 01 28 FE 01 AE 2C 00 00 00  /{¬ ....(þ.®,...
020855DCE0  01 06 00 07 81 04 9B 80 5A F4 C0 01 07 00 00 32  ......›€ZôÀ....2
020855DCF0  00 00 03 00 04 80 00 00 00 01 25 88 80 00 01 0D  .....€....%ˆ€...
020855DD00  00 80 44 0B 61 F7 FF 0F 74 68 1C 35 D8 10 40 7D  .€D.a÷ÿ.th.5Ø.@}
020855DD10  5C 87 E3 E6 60 00 00 03 00 EE D0 48 78 50 47 31  \‡ãæ`....îÐHxPG1
```

Figure 7. NAL units decoded from table above (Bunker Hill DVR)

Offset	Bytes	NAL ref idc	NAL unit type	Content
27674507912	Hex 0000000167	6	7	Sequence parameter set
27674507924	Hex 0000000168	6	8	Picture parameter set
27674507933	Hex 0000000106	0	6	Supplemental enhancement information
27674507942	Hex 0000000165	6	5	Coded slice of an IDR picture (usually an I-frame)

Figure 8. NAL units decoded from table above (Swann DVR)

Offset	Bytes	NAL ref idc	NAL unit type	Content
0x020855DCB0	0x0000000127	2	7	Sequence parameter set
0x020855DCD4	0x0000000128	2	8	Picture parameter set
0x020855DCDD	0x0000000106	0	6	Supplemental enhancement information
0x020855DCF6	0x0000000125	2	5	Coded slice of an IDR picture (usually an I-frame)

5.2.3 Extract a sample

Beginning with the first zero byte of the first parameter set, select and copy a section of footage as a sample. A selection of 30MiB should be a reasonable practical size. In the example above, this would be a continuous section of 30MiB beginning at offset 0x020855DCB0.

Once the sample has been saved, the data can be encapsulated in an *Audio Video Interleave* (AVI) container using ffmpeg with the following options:

```
ffmpeg –f h264 –i input_file –c:v copy output_file.avi
```

If this fails, it is possible that transcoding the data are encapsulating the data into a different container may help. This can be done by substituting –c:v huffyuv for –c:v copy or by changing the output file extension from .avi to .mkv respectively.

If this still fails, it may be due to blocking or the wrap-around point may have been selected (not covered here). Another sample should be taken using the same method but from another location in the disk and the process repeated. This can be repeated a few times at various points, however if multiple sections fail it is likely that techniques beyond the scope of this document are required for recovery if at all possible.

5.3 Decoding headers

Many security DVR systems include *header* data at the beginning of each *chunk* of video. If the video is encoded as H.264, each chunk will typically include a header followed by a section of H.264 byte stream data and sometimes a footer. The byte stream in turn usually consists of either multiple NAL units (at least a picture parameter set, a sequence parameter set and an IDR slice) if the chunk contains an I-frame or a single NAL unit (a non-IDR slice) if the chunk contains a P-frame.

Each NAL unit is preceded by the four bytes 0x00000001 (a *zero byte* followed by a *start code prefix*). The first byte of a NAL unit can be used to determine the type of data contained within according to Section 7.3.1 and Table 7-1 of [4].

The header data often includes the camera number, date and time. It is usually possible to decode the needed parts of the headers and retrieve this data which can then be used to recover specific portions of video based on camera, date, and time. One method is given below.

Extracting headers can be done with a hex editor, but the process is laborious and lengthy for a large number of headers. It is best to use a computer program if one is available, or write one if it is not.

5.3.1 Sample footage

In many cases where recovery is required, the security DVR system is not available or cannot be used. If the system is available, or a very similar system is available, then it is best to create some test footage first. A small hard drive should be used, wiped with a known byte pattern such as 0x00 or 0xFF. A short amount of footage can then be recorded using multiple inputs. This way when decoding the headers, the true values of the parameters such as date, time, and the number of cameras are already known.

If this is not possible, the evidence can be used, but parameters such as the date and time should be verified by other means if possible or assumptions may need to be made.

5.3.2 Rough header extraction

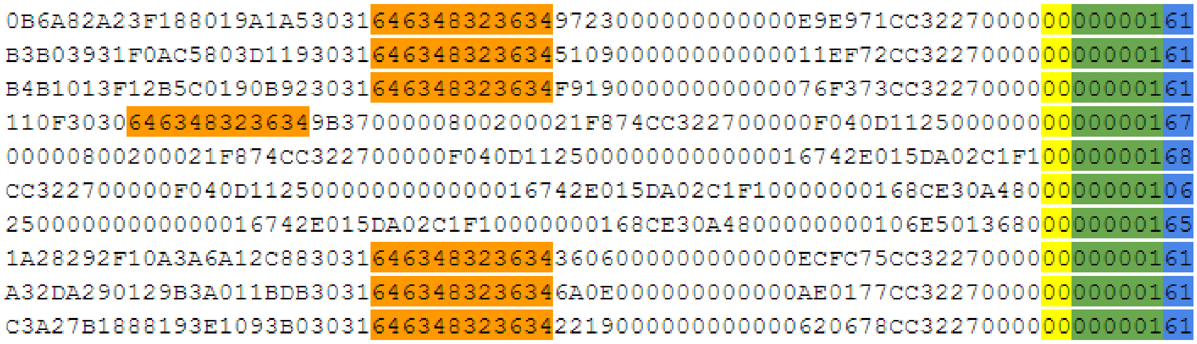
Find each occurrence of 0x00000001 within the video data. For each occurrence, extract *n* bytes beginning before the 0x00000001 and ending with the last byte following the 0x00000001 (which should be the first byte of the NAL unit, including the NAL unit type). In this case *n* is the estimated length of the header plus five bytes. It does not matter if this is over-estimated at this stage; 128 bytes should be more than sufficient.

5.3.3 Locate signature and refine headers

The rough extraction will likely result in false positives. Each chunk containing an I-Frame will typically contain multiple occurrences of the bytes 0x00000001, but only the first will be relevant. Correctly identified headers will likely have a significant degree of similarity between bytes in the same locations. This rough alignment can be used to locate a *signature*, a set of bytes which are unlikely to occur by chance but occur unchanged in each correctly identified header within the video data.

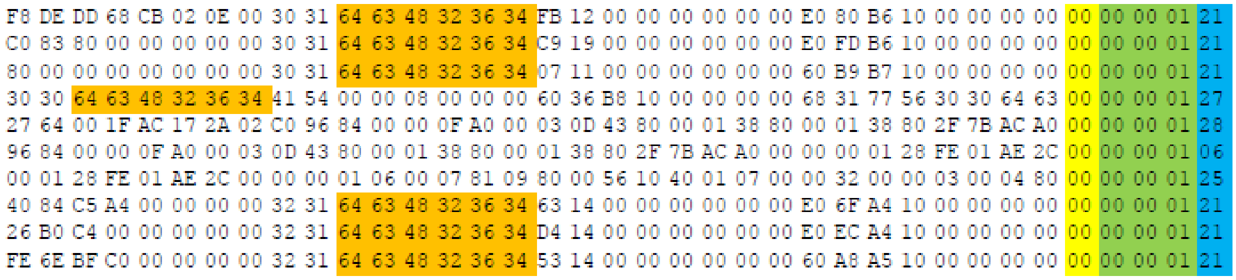
The example below consists of the last 37 bytes of ten consecutive “rough headers” extracted in the manner described above from a Swann brand security digital video recorder system. The zero bytes have been colored yellow, the start code prefixes colored green, and the first byte of the NAL units have been colored blue. The bytes 0x646348323634 (equivalent to the text “dcH264”) can be seen whenever the NAL unit is of type 1 or type 7, corresponding to chunks containing P-frames and chunks containing I-frames respectively. It cannot be seen from this small extract, but the type 8, type 6 and type 5 NAL units are all contained in the same I-frame chunk which is why they do not have (or need) a signature. Note that the headers for I-frames differ slightly to the headers for P-frames.

Figure 9. Part of rough header extraction (Bunker Hill DVR)



```
0B6A82A23F188019A1A53031646348323634972300000000000000E9E971CC3227000000000000161
B3B03931F0AC5803D1193031646348323634510900000000000000011EF72CC3227000000000000161
B4B1013F12B5C0190B923031646348323634F91900000000000000076F373CC3227000000000000161
110F3030646348323634 9B37000000800200021F874CC322700000F040D1125000000000000000167
0000080020002 1F874CC322700000F040D11250000000000000016742E015DA02C1F100000000168
CC322700000F040D112500000000000000016742E015DA02C1F10000000168CE30A48000000000106
25000000000000016742E015DA02C1F10000000168CE30A480000000000106E501368000000000165
1A28292F10A3A6A12C883031646348323634360600000000000000ECFC75CC3227000000000000161
A32DA290129B3A011BDB30316463483236346A0E000000000000AE0177CC3227000000000000161
C3A27B1888193E1093B03031646348323634221900000000000000620678CC3227000000000000161
```

Figure 10. Part of rough header extraction (Swann DVR)



```
F8 DE DD 68 CB 02 0E 00 30 31 64 63 48 32 36 34 FB 12 00 00 00 00 00 00 E0 80 B6 10 00 00 00 00 00 00 00 01 21
C0 83 80 00 00 00 00 00 30 31 64 63 48 32 36 34 C9 19 00 00 00 00 00 00 E0 FD B6 10 00 00 00 00 00 00 00 01 21
80 00 00 00 00 00 00 00 30 31 64 63 48 32 36 34 07 11 00 00 00 00 00 00 60 B9 B7 10 00 00 00 00 00 00 00 01 21
30 30 64 63 48 32 36 34 41 54 00 00 08 00 00 00 60 36 B8 10 00 00 00 00 68 31 77 56 30 30 64 63 00 00 00 01 27
27 64 00 1F AC 17 2A 02 C0 96 84 00 00 0F A0 00 03 0D 43 80 00 01 38 80 00 01 38 80 2F 7B AC A0 00 00 00 01 28
96 84 00 00 0F A0 00 03 0D 43 80 00 01 38 80 00 01 38 80 2F 7B AC A0 00 00 00 01 28 FE 01 AE 2C 00 00 00 01 06
00 01 28 FE 01 AE 2C 00 00 00 01 06 00 07 81 09 80 00 56 10 40 01 07 00 00 32 00 00 03 00 04 80 00 00 00 01 25
40 84 C5 A4 00 00 00 00 32 31 64 63 48 32 36 34 63 14 00 00 00 00 00 00 E0 6F A4 10 00 00 00 00 00 00 00 01 21
26 B0 C4 00 00 00 00 00 32 31 64 63 48 32 36 34 D4 14 00 00 00 00 00 00 E0 EC A4 10 00 00 00 00 00 00 00 01 21
FE 6E BF C0 00 00 00 00 32 31 64 63 48 32 36 34 53 14 00 00 00 00 00 00 60 A8 A5 10 00 00 00 00 00 00 00 01 21
```

5.3.4 Refine list of headers using the signature

Step one is repeated, but using the signature bytes in place of the 0x00000001. This step as well as following steps can be seen in **Appendix B – Table of extracted headers from Swann DVR**.

5.3.5 Assess the header list

The new header list should consist of every valid header but no invalid headers. If this is not the case, another signature should be selected (return to step 3). If a signature cannot be found that singles out correct headers then it is better to include false positives than exclude correct headers, or a more complex method can be used using conditional logic. Note that *blocking* can cause headers to be divided and possibly excluded. This is addressed later (see 5.4.2).

If headers are being extracted by hand, it is not feasible to extract a great many. In this case, it is best to try and extract one or two corresponding to each different NAL unit that contains a header (usually just types 1 and 5).

5.3.6 Locate the camera, date/time, length and frame type in the header

The camera, date/time, frame type and the length of the chunk are usually stored in the header. A process of "reverse engineering" can be used to determine how these are stored. By lining up the headers, these parameters become much more obvious.

Camera. The camera will usually be stored as a number. The number is likely to be limited to a few values, usually low (such as the numbers 0 to 7 for an 8 camera system). The camera number will typically be repeated before changing with multiple consecutive chunks coming from the same camera.

Date/Time. The date and time is usually stored as a single set of four or more bytes. This often resembles ISO 8601 format or Unix/Epoch format, though usually stored relative to the local time and not UTC. The date/time will predominantly be monotonically increasing for each single camera and may decrease when the camera number changes. The number may repeat before increasing depending on frame rates. Sometimes it is only recorded for some chunks (such as chunks containing I-frames).

Length. The length of byte stream data contained in the chunk can be estimated manually using a hex editor. The byte stream begins with the zero byte and continues until approximately the next header. In some cases, there may be some padding bytes. The manually obtained number can be compared with numbers found in the header until the length is found. Alternatively, the length can simply be calculated for each chunk using the above method.

Frame type. As described above, chunks typically contain I-frames or P-frames, and usually only a single frame contained in a single slice. This frame type can be read from the NAL unit type; chunks containing I-frames typically begin with a sequence parameter set or picture parameter set (type 7 or 8) while P-frames typically have only one NAL unit, a non-IDR slice (unit type 1). Often the header will also contain the frame type as well. While it is not always essential to decode the frame type, sometimes the header format will depend on it and so knowing the frame type can help in decoding the other parameters.

Endianness. Often numbers are stored as little-endian integers, meaning that the least significant byte comes first so that the bytes 0x01020304 should be read as the hexadecimal number 0x04030201.

Example. Referring to the example of the Swann system represented in **Appendix B – Table of extracted headers from Swann DVR**, the headers have been aligned on their signatures and the various parameters have been colored.

The first column corresponds to the camera number where 0x30 (ASCII ‘0’) represents camera 1 and 0x32 (ASCII ‘2’) represents camera 3. The second (pink) column holds the frame type. In this case, 0x30 (ASCII ‘0’) represents a chunk containing an I-frame and 0x31 (ASCII ‘1’) represents a chunk containing a P-frame. These first two columns combined with the next two (always “dc”) may look familiar when considering the AVI standard. Though they bear a resemblance, they are not quite consistent.

Columns 0x02 to 0x07 (in orange) contain the signature used to identify and align the headers. In this case, the signature was the string “dcH264”. The signature often remains consistent between different systems from the same manufacturer but can vary between manufacturers. The signature is immediately followed by two columns holding the length of the byte stream data contained in the chunk (colored blue) as a little-endian integer.

The time code is stored in columns 0x18 to 0x1B, but only in chunks containing I-frames. The timecode is stored in a manner similar to Unix-time as a little-endian integer. This can be decoded using on-line decoders, hex editors, in spreadsheets or using computer programs. The byte stream containing the video data within the chunk begins at column 0x18 or column 0x20 depending on whether the frame type contained in the chunk is a P-frame or I-frame respectively.

5.4 Demultiplexing and extracting footage

Security DVRs often multiplex data so that multiple cameras may be stored on the same disk. This causes the video in extracted samples to switch rapidly between multiple camera views. If switching occurs part-way through a GOP, this will cause errors to occur in the footage. With blocking, the footage may even switch part way through a chunk. The rapid changing of camera and errors in decoding may make the video difficult or impossible to view or decode.

However, using the information decoded from the extracted headers it is possible to demultiplex the video at the data level. The optimum method to use depends on how the data has been multiplexed. The two most common methods are chunk-level multiplexing and block-level multiplexing.

Demultiplexing is laborious and is again best done using a computer program. Small sections of footage can be demultiplexed by hand, particularly deblocking. However, it is not practical to do this for large sections of footage, particularly if it is multiplexed at the chunk level. For this reason, the next sections are written with the assumption that a computer program is used to perform the task.

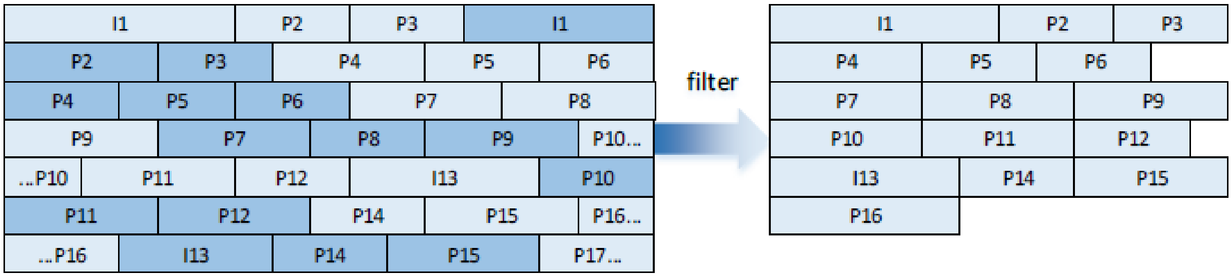
If a computer program is not available and a hex editor must be used, then it is best to locate the required headers within the video data rather than attempt to extract any useful number of headers to begin with. If the desired parameters are known and the header format has been at least partially reverse engineered, then it is possible to create at least part of a desired header using the desired parameters. The hex editor can then be used to search the video data for this header.

For example, from the Swann system decoded above, any header created on Sunday the 20th of December, 2015 at 22:53:33 corresponding to an I-frame captured from camera one should contain the bytes 0x000000006D317756303064630000000127. Searching the video data for these bytes should find the relevant section. This is a very long string of bytes and very unlikely to result in a false hit.

5.4.1 Demultiplexing chunk-level multiplexed data

Chunk-level multiplexing involves interleaving the chunks in a similar manner to that found in containers formats like the AVI format.

Figure 11. Demultiplexing chunk-level multiplexed data



Demultiplexing chunk-level multiplexed footage is simpler than for block-level multiplexed data and works quite well on both forms of multiplexing, so it is recommended that if the form of multiplexing is unknown that this method is tried first.

To extract a single camera, iterate through each header and perform the following steps:

- If the header is not from the selected camera, continue on to the next header.
- If the header is not from the required date and time period, continue on to the next header.
- Beginning with the first zero byte in the chunk, append all of the byte-stream data to the output file (that is, the contents of the chunk excluding the header and any footer or padding bytes).

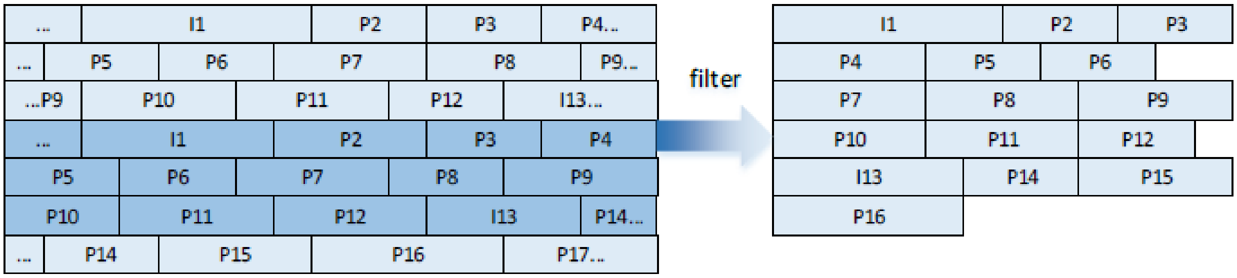
Once all headers have been filtered and all data extracted in this way, the output file should contain an H.264 byte stream. There is no guarantee that the first frame is an I-frame, so the beginning of the file should be trimmed first. This can be done by locating the first I-frame as done when extracting a sample and removing all data before it. Alternatively, this can be incorporated into the above algorithm.

If the footage was multiplexed at the block level (blocking) then there may be incomplete chunks remaining within the footage. It is likely that these will still cause errors, though they will likely be few, particularly when compared to the original sample taken. Again, whether or not further work needs to be done is dependent upon the case.

5.4.2 Demultiplexing block-level multiplexed data

Data may be multiplexed at the block level. The stream of data from each camera is converted into chunks as normal. These chunks are then concatenated into a separate stream for each camera. These streams are then divided into blocks of equal size and interleaved with each other.

Figure 12. Demultiplexing block-level multiplexed data



To demultiplex blocked data, the video data must be divided into blocks. To do this the start of the first block must be known and the block size must be either known or estimated. The start of the first block is usually at the start of the video data and the block size can be estimated with the help of the list of headers.

Each block contains footage from only a single camera. In addition, block lengths tend to be “nice” numbers, such as 0x10000 or 0x20000 bytes (the Swann system used in the example is blocked with a block size of 0x20000 bytes). The block boundaries can be estimated by looking at the offsets of the last header of one camera and the first header of another; the boundary must lie between the two. By observing a number of these camera transitions, an estimate can be made as to the block size.

Once the data has been separated into blocks it can be deblocked for a single camera using the following steps.

For each block:

- Get all of the headers in that block
- Check that all headers have the same camera number. If not, the block parameters are wrong; terminate the algorithm.
- If the camera is not the selected camera, continue to the next block.
- If none of the headers contain a date and time of interest, continue to the next block.
- Copy the entire block (do *not* extract the byte streams at this stage) into the output file.

It may be possible to play or transcode the deblocked data using ffmpeg without extracting the byte stream first. The file still requires trimming as with chunk-level demultiplexing, as again there is no guarantee that the footage begins with an I-frame and parameter set.

If the footage cannot be transcoded or played, or there are significant errors, the byte stream may have to be extracted. Headers should be extracted from the deblocked file. This should be done as any header that was divided across a block boundary will likely contain incorrect parameters, and some headers may have been missed altogether if the signature bytes were divided across different blocks.

Once the headers are extracted, the byte stream can then be extracted from the deblocked file just as with demultiplexing the chunk-level multiplexed data, though there is no need to check the camera number in this case. The resulting file will be raw H.264 data just as with the chunk-level demultiplexing, though again it may need to be trimmed.

Note that sometimes the data may be divided into multiple partitions. This can cause the demultiplexing to fail if one partition is offset by a fraction of a block from another. The easiest solution is to demultiplex partitions separately. Often the extracted headers can be used to find the required partitions. If only a small amount of footage is needed, in most cases it will be found on a single partition. Identifying partition boundaries is out of the scope for this document, though it is often not difficult and only requires a hex editor.

6. Additional Guidance

Refer to *SWGIT Best Practices for the Analysis of Digital Video Recorders* [5] for guidance on equipment preparation, acquisition, analysis, documentation, and reporting.

7. References

The following documents are referenced in this document:

[1] "Original images," *Forensic Audio Video Evidence Unit, Victoria (Australia) Police Forensic Services Department*, 2016.

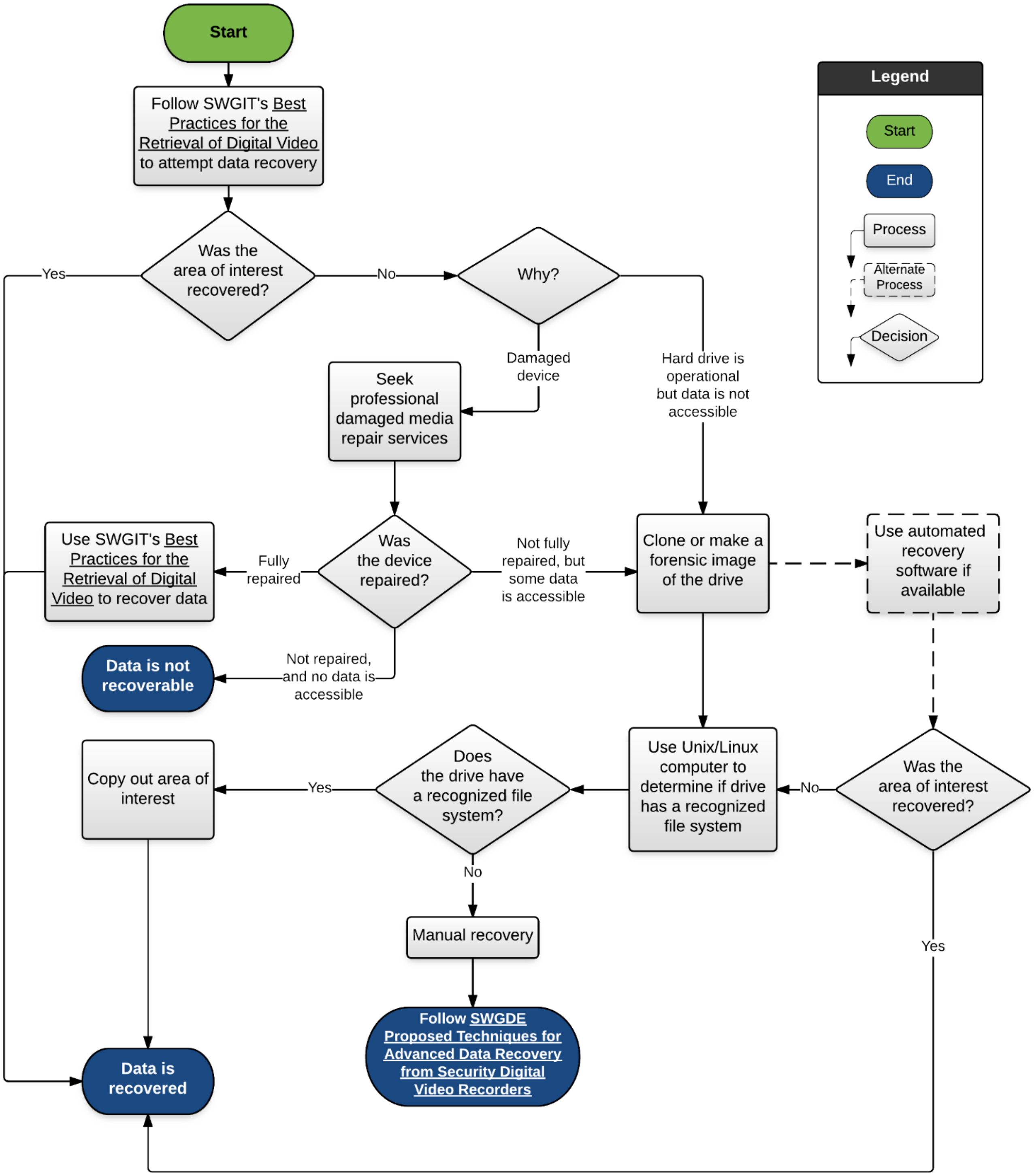
[2] Scientific Working Group on Imaging Technology. Section 24: Best Practices for the Retrieval of Digital Video. [Online]. https://www.swgit.org/documents/Current%20Documents

[3] Scientific Working Group on Digital Evidence and Scientific Working Group on Imaging Technology. SWGDE/SWGIT Guidelines & Recommendations for Training in Digital & Multimedia Evidence. [Online]. https://www.swgde.org/documents/Current%20Documents

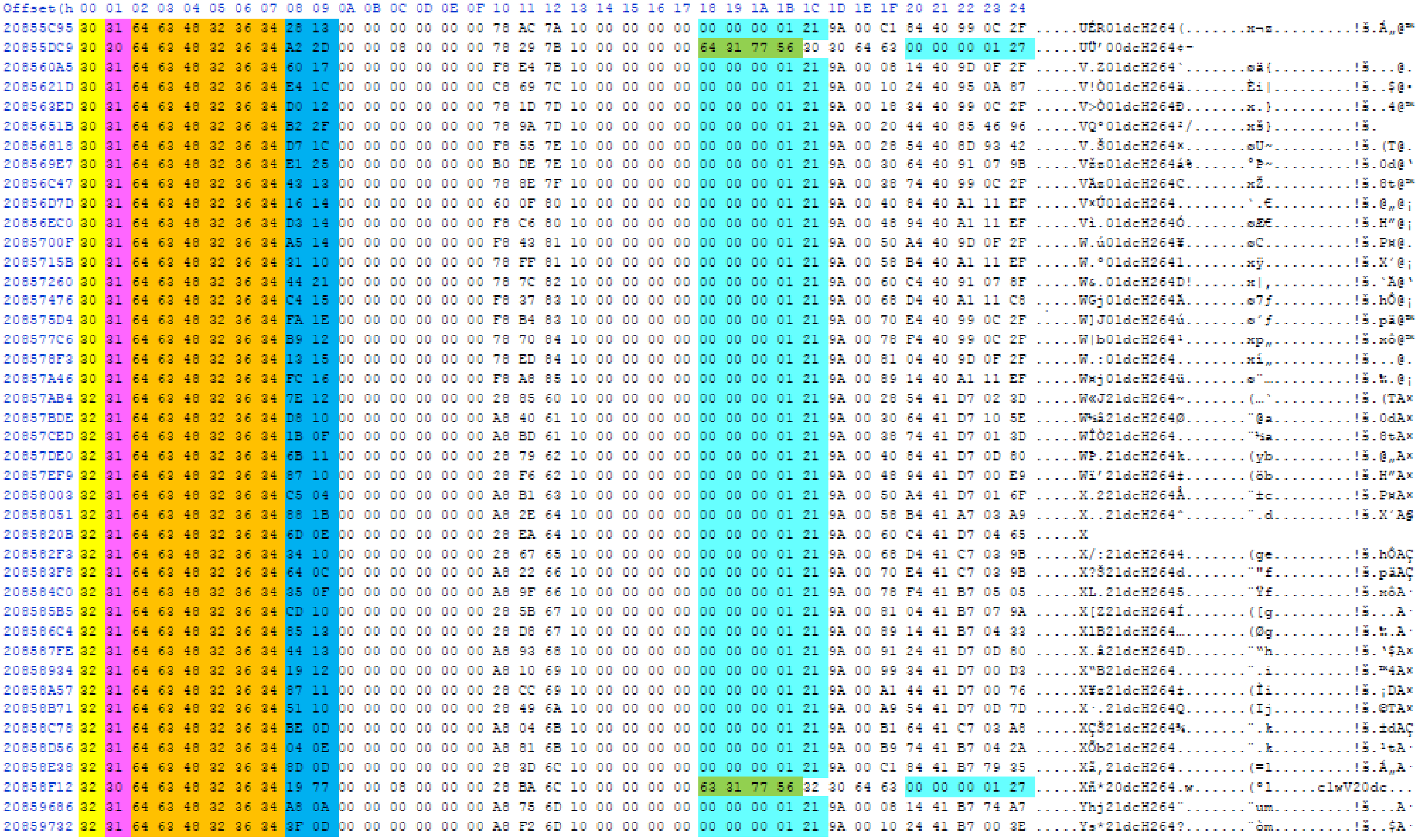
[4] Telecommunication Standardization Sector of ITU, "H.264 : Advanced video coding for generic audiovisual services," Standard No. H.264, 2013.

[5] Scientific Working Group on Imaging Technology, "Section 23: Best Practices for the Analysis of Digital Video Recorders". [Online]. https://www.swgit.org/documents/Current%20Documents

Appendix A – Decision-Making Process for DVR Recovery Methods


Start
Follow SWGIT's Best Practices for the Retrieval of Digital Video to attempt data recovery
Was the area of interest recovered?
Yes
No
Why?
Damaged device
Hard drive is operational but data is not accessible
Seek professional damaged media repair services
Was the device repaired?
Fully repaired
Not fully repaired, but some data is accessible
Not repaired, and no data is accessible
Use SWGIT's Best Practices for the Retrieval of Digital Video to recover data
Data is not recoverable
Clone or make a forensic image of the drive
Use automated recovery software if available
Was the area of interest recovered?
No
Yes
Use Unix/Linux computer to determine if drive has a recognized file system
Does the drive have a recognized file system?
Yes
No
Copy out area of interest
Manual recovery
Follow SWGDE Proposed Techniques for Advanced Data Recovery from Security Digital Video Recorders
Data is recovered
Legend
Start
End
Process
Alternate Process
Decision

Appendix B – Table of extracted headers from Swann DVR



```
Offset(h 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24
20855C95 30 31 64 63 48 32 36 34 28 13 00 00 00 00 00 00 78 AC 7A 10 00 00 00 00 00 00 00 01 21 9A 00 C1 84 40 99 0C 2F .....UÉR01dcH264(.......x¬z.........!š.Á„@™
20855DC9 30 30 64 63 48 32 36 34 A2 2D 00 00 08 00 00 00 78 29 7B 10 00 00 00 00 64 31 77 56 30 30 64 63 00 00 00 01 27 .....UÜ'00dcH264¢-
208560A5 30 31 64 63 48 32 36 34 60 17 00 00 00 00 00 00 F8 E4 7B 10 00 00 00 00 00 00 00 01 21 9A 00 08 14 40 9D 0F 2F .....V.Z01dcH264`.......øä{.........!š...@.
2085621D 30 31 64 63 48 32 36 34 E4 1C 00 00 00 00 00 00 C8 69 7C 10 00 00 00 00 00 00 00 01 21 9A 00 10 24 40 95 0A 87 .....V!Ò01dcH264ä.......Èi|.........!š..$@•
208563ED 30 31 64 63 48 32 36 34 D0 12 00 00 00 00 00 00 78 1D 7D 10 00 00 00 00 00 00 00 01 21 9A 00 18 34 40 99 0C 2F .....V>Ò01dcH264Ð.......x.}.........!š..4@™
2085651B 30 31 64 63 48 32 36 34 B2 2F 00 00 00 00 00 00 78 9A 7D 10 00 00 00 00 00 00 00 01 21 9A 00 20 44 40 85 46 96 .....VQ°01dcH264²/......xš}.........!š.
20856818 30 31 64 63 48 32 36 34 D7 1C 00 00 00 00 00 00 F8 55 7E 10 00 00 00 00 00 00 00 01 21 9A 00 28 54 40 8D 93 42 .....V.Š01dcH264×.......øU~.........!š.(T@.
208569E7 30 31 64 63 48 32 36 34 E1 25 00 00 00 00 00 00 B0 DE 7E 10 00 00 00 00 00 00 00 01 21 9A 00 30 64 40 91 07 9B .....Vžz01dcH264á%......°Þ~.........!š.0d@‘
20856C47 30 31 64 63 48 32 36 34 43 13 00 00 00 00 00 00 78 8E 7F 10 00 00 00 00 00 00 00 01 21 9A 00 38 74 40 99 0C 2F .....VÃz01dcH264C.......xŽ..........!š.8t@™
20856D7D 30 31 64 63 48 32 36 34 16 14 00 00 00 00 00 00 60 0F 80 10 00 00 00 00 00 00 00 01 21 9A 00 40 84 40 A1 11 EF .....V×Ů01dcH264........`.€.........!š.@„@¡
20856EC0 30 31 64 63 48 32 36 34 D3 14 00 00 00 00 00 00 F8 C6 80 10 00 00 00 00 00 00 00 01 21 9A 00 48 94 40 A1 11 EF .....Vì.01dcH264Ó.......øÆ€.........!š.H”@¡
2085700F 30 31 64 63 48 32 36 34 A5 14 00 00 00 00 00 00 F8 43 81 10 00 00 00 00 00 00 00 01 21 9A 00 50 A4 40 9D 0F 2F .....W.ú01dcH264¥.......øC..........!š.P¤@.
2085715B 30 31 64 63 48 32 36 34 31 10 00 00 00 00 00 00 78 FF 81 10 00 00 00 00 00 00 00 01 21 9A 00 58 B4 40 A1 11 EF .....W.°01dcH2641.......xÿ..........!š.X´@¡
20857260 30 31 64 63 48 32 36 34 44 21 00 00 00 00 00 00 78 7C 82 10 00 00 00 00 00 00 00 01 21 9A 00 60 C4 40 91 07 8F .....W&.01dcH264D!......x|‚.........!š.`Ä@‘
20857476 30 31 64 63 48 32 36 34 C4 15 00 00 00 00 00 00 F8 37 83 10 00 00 00 00 00 00 00 01 21 9A 00 68 D4 40 A1 11 C8 .....WGj01dcH264Ä.......ø7ƒ.........!š.hÔ@¡
208575D4 30 31 64 63 48 32 36 34 FA 1E 00 00 00 00 00 00 F8 B4 83 10 00 00 00 00 00 00 00 01 21 9A 00 70 E4 40 99 0C 2F .....W]J01dcH264ú.......ø´ƒ.........!š.pä@™
208577C6 30 31 64 63 48 32 36 34 B9 12 00 00 00 00 00 00 78 70 84 10 00 00 00 00 00 00 00 01 21 9A 00 78 F4 40 99 0C 2F .....W|b01dcH264¹.......xp„.........!š.xô@™
208578F3 30 31 64 63 48 32 36 34 13 15 00 00 00 00 00 00 78 ED 84 10 00 00 00 00 00 00 00 01 21 9A 00 81 04 40 9D 0F 2F .....W.:01dcH264........xí„.........!š...@.
20857A46 30 31 64 63 48 32 36 34 FC 16 00 00 00 00 00 00 F8 A8 85 10 00 00 00 00 00 00 00 01 21 9A 00 89 14 40 A1 11 EF .....W¤j01dcH264ü.......ø¨….........!š.‰.@¡
20857AB4 32 31 64 63 48 32 36 34 7E 12 00 00 00 00 00 00 28 85 60 10 00 00 00 00 00 00 00 01 21 9A 00 28 54 41 D7 02 3D .....W«J21dcH264~.......(…`.........!š.(TA×
20857BDE 32 31 64 63 48 32 36 34 D8 10 00 00 00 00 00 00 A8 40 61 10 00 00 00 00 00 00 00 01 21 9A 00 30 64 41 D7 10 5E .....W¼â21dcH264Ø.......¨@a.........!š.0dA×
20857CED 32 31 64 63 48 32 36 34 1B 0F 00 00 00 00 00 00 A8 BD 61 10 00 00 00 00 00 00 00 01 21 9A 00 38 74 41 D7 01 3D .....WÎÒ21dcH264........¨½a.........!š.8tA×
20857DE0 32 31 64 63 48 32 36 34 6B 11 00 00 00 00 00 00 28 79 62 10 00 00 00 00 00 00 00 01 21 9A 00 40 84 41 D7 0D 80 .....WÞ.21dcH264k.......(yb.........!š.@„A×
20857EF9 32 31 64 63 48 32 36 34 87 10 00 00 00 00 00 00 28 F6 62 10 00 00 00 00 00 00 00 01 21 9A 00 48 94 41 D7 00 E9 .....Wï'21dcH264‡.......(öb.........!š.H”A×
20858003 32 31 64 63 48 32 36 34 C5 04 00 00 00 00 00 00 A8 B1 63 10 00 00 00 00 00 00 00 01 21 9A 00 50 A4 41 D7 01 6F .....X.221dcH264Å.......¨±c.........!š.P¤A×
20858051 32 31 64 63 48 32 36 34 88 1B 00 00 00 00 00 00 A8 2E 64 10 00 00 00 00 00 00 00 01 21 9A 00 58 B4 41 A7 03 A9 .....X..21dcH264ˆ.......¨.d.........!š.X´A§
2085820B 32 31 64 63 48 32 36 34 6D 0E 00 00 00 00 00 00 28 EA 64 10 00 00 00 00 00 00 00 01 21 9A 00 60 C4 41 D7 04 65 .....X
208582F3 32 31 64 63 48 32 36 34 34 10 00 00 00 00 00 00 28 67 65 10 00 00 00 00 00 00 00 01 21 9A 00 68 D4 41 C7 03 9B .....X/:21dcH2644.......(ge.........!š.hÔAÇ
208583F8 32 31 64 63 48 32 36 34 64 0C 00 00 00 00 00 00 A8 22 66 10 00 00 00 00 00 00 00 01 21 9A 00 70 E4 41 C7 03 9B .....X?Š21dcH264d.......¨"f.........!š.päAÇ
208584C0 32 31 64 63 48 32 36 34 35 0F 00 00 00 00 00 00 A8 9F 66 10 00 00 00 00 00 00 00 01 21 9A 00 78 F4 41 B7 05 05 .....XL.21dcH2645.......¨Ÿf.........!š.xôA·
208585B5 32 31 64 63 48 32 36 34 CD 10 00 00 00 00 00 00 28 5B 67 10 00 00 00 00 00 00 00 01 21 9A 00 81 04 41 B7 07 9A .....X[Z21dcH264Í.......([g.........!š...A·
208586C4 32 31 64 63 48 32 36 34 85 13 00 00 00 00 00 00 28 D8 67 10 00 00 00 00 00 00 00 01 21 9A 00 89 14 41 B7 04 33 .....X1B21dcH264…........(Øg.........!š.‰.A·
208587FE 32 31 64 63 48 32 36 34 44 13 00 00 00 00 00 00 A8 93 68 10 00 00 00 00 00 00 00 01 21 9A 00 91 24 41 D7 0D 80 .....X.â21dcH264D.......¨“h.........!š.‘$A×
20858934 32 31 64 63 48 32 36 34 19 12 00 00 00 00 00 00 A8 10 69 10 00 00 00 00 00 00 00 01 21 9A 00 99 34 41 D7 00 D3 .....X"B21dcH264........¨.i.........!š.™4A×
20858A57 32 31 64 63 48 32 36 34 87 11 00 00 00 00 00 00 28 CC 69 10 00 00 00 00 00 00 00 01 21 9A 00 A1 44 41 D7 00 76 .....X¥z21dcH264‡.......(Ìi.........!š.¡DA×
20858B71 32 31 64 63 48 32 36 34 51 10 00 00 00 00 00 00 28 49 6A 10 00 00 00 00 00 00 00 01 21 9A 00 A9 54 41 D7 0D 7D .....X·.21dcH264Q.......(Ij.........!š.©TA×
20858C78 32 31 64 63 48 32 36 34 BE 0D 00 00 00 00 00 00 A8 04 6B 10 00 00 00 00 00 00 00 01 21 9A 00 B1 64 41 C7 03 A8 .....XÇŠ21dcH264¾.......¨.k.........!š.±dAÇ
20858D56 32 31 64 63 48 32 36 34 04 0E 00 00 00 00 00 00 A8 81 6B 10 00 00 00 00 00 00 00 01 21 9A 00 B9 74 41 B7 04 2A .....XŐb21dcH264........¨.k.........!š.¹tA·
20858E38 32 31 64 63 48 32 36 34 8D 0D 00 00 00 00 00 00 28 3D 6C 10 00 00 00 00 00 00 00 01 21 9A 00 C1 84 41 B7 79 35 .....Xã,21dcH264........(=l.........!š.Á„A·
20858F12 32 30 64 63 48 32 36 34 19 77 00 00 08 00 00 00 28 BA 6C 10 00 00 00 00 63 31 77 56 32 30 64 63 00 00 00 01 27 .....Xñ*20dcH264.w......(°l.....c1wV20dc...
20859686 32 31 64 63 48 32 36 34 A8 0A 00 00 00 00 00 00 A8 75 6D 10 00 00 00 00 00 00 00 01 21 9A 00 08 14 41 B7 74 A7 .....Yhj21dcH264¨.......¨um.........!š...A·
20859732 32 31 64 63 48 32 36 34 3F 0D 00 00 00 00 00 00 A8 F2 6D 10 00 00 00 00 00 00 00 01 21 9A 00 10 24 41 B7 00 3E .....Ys*21dcH264?.......¨òm.........!š..$A·
```

Camera number	Frame type	Signature	Length of byte stream	Time code (Unix time)	Zero byte, start code prefix and first byte of NAL unit

SWGDE Proposed Techniques for Advanced Data Recovery from Security Digital Video Recorders Containing H.264 Data

History

Revision	Issue Date	Section	History
1.0	01/15/2015	All	Original release; titled SWGDE Best Practices for the Recovery of Data from CCTV Digital Video Recorders. Voted by SWGDE for release as a Draft for Public Comment.
1.0	02/05/2015	All	Formatting and technical edit performed for release as a Draft for Public Comment.
--	06/04/2015	All	Committee voted to remove the draft from public comment for rewriting.
1.1	09/16/2015	Title, All, Appendix A	Committee retitled the document, SWGDE Proposed Techniques for Advanced Data Recovery from Security Digital Video Recorders, restructured and rewrote all sections, and added a decision process flow chart as an appendix. Voted by SWGDE to re-release as a new Draft for Public Comment.
1.1	9/29/2015	All	Formatting and technical edit performed for release as a Draft for Public Comment.
1.2	01/14/2016	All, Appendix B	Significant content changes were made throughout and Appendix B was added in response to public comments. Voted by SWGDE to re-release as a new Draft for Public Comment.
1.2	02/08/2016	All	Formatting and technical edit performed for release as a Draft for Public Comment.
1.2	06/09/2016	--	SWGDE voted to publish as an Approved document.
1.2	06/23/2016	--	Formatted and posted as an Approved document.