



Inspection and Sanitization Guidance for the Wavelet Scalar Quantization (WSQ) Biometric Image Format

Version 1.0

21 January 2014



**National Security Agency
9800 Savage Rd, Suite 6721
Ft. George G. Meade. MD 20755**

**Authored/Released by:
Unified Cross Domain Capabilities Office
cds_tech@nsa.gov**

DOCUMENT REVISION HISTORY

Date	Version	Description
1/21/14	1.0	Version 1.0 release
12/13/2017	1.0	Updated Contact information, IAC Logo, Cited Trademarks and Copyrights, Expanded Acronyms, and added Legal Disclaimer

Disclaimer

The information and opinions contained in this document are provided “as is” and without any warranties or guarantees. References herein to any specific commercial product, process, or service by trade name, trademark, manufacturer or otherwise, does not constitute or imply its endorsement, recommendation or favoring by the United States Government and this guidance shall not be used for advertising or product endorsement purposes

EXECUTIVE SUMMARY

This *Inspection and Sanitization Guidance (ISG) for Wavelet Scalar Quantization (WSQ)* document provides guidelines and specifications for developing file inspection and sanitization software for WSQ files, as defined by the WSQ Gray-Scale Fingerprint Image Compression Specification prepared by the Federal Bureau of Investigation's Criminal Justice Information Services Division (IAFIS-IC-0110(V3)).

The Criminal Justice Information Services Division (CJIS), along with the assistance of the National Institute of Standards and Technology (NIST) and Los Alamos National Laboratory (LANL), developed standards for fingerprint digitization in cooperation with the commercial vendor and criminal justice communities. The developers have discovered that fingerprint image files could be compressed using the WSQ compression technique at a compression ratio of 15:1 without introducing blocking artifacts that are created by other compression methods. WSQ preserves fine fingerprint details such as fingerprint friction ridges, minutiae, and sweat pores to support the fingerprint identification process.

This guidance document examines the WSQ specification for data attack, data hiding, and data disclosure risks that exists within the file structure. It provides a breakdown of each component of a WSQ file and provides recommendations that can help assure that the WSQ file is compliant with the specifications and does not contain any hidden data.

TABLE OF CONTENTS

1. SCOPE.....	1-1
1.1 PURPOSE OF THIS DOCUMENT	1-1
1.2 INTRODUCTION.....	1-1
1.3 BACKGROUND	1-1
1.4 DOCUMENT ORGANIZATION.....	1-2
1.5 RECOMMENDATIONS	1-3
1.5.1 Actions.....	1-3
1.5.2 Action Options	1-5
1.6 DATA TRANSFER GUIDANCE	1-5
1.7 DOCUMENT LIMITATIONS.....	1-6
1.7.1 Covert Channel Analysis.....	1-6
2. CONSTRUCT OVERVIEW	2-1
2.1 CONSTRUCTS	2-1
3. WAVELET SCALAR QUANTIZATION OVERVIEW	3-1
3.1 WSQ CHARACTER ENCODING.....	3-2
4. WSQ CONSTRUCTS.....	4-1
4.1 START OF IMAGE/END OF IMAGE MARKERS	4-1
4.2 GENERIC MARKER SEGMENTS	4-2
4.3 COMMENT SEGMENTS	4-4
4.4 START OF FRAME SEGMENT	4-7
4.5 START OF BLOCK SEGMENTS.....	4-9
4.6 ENTROPY CODED SEGMENTS AND RESTART MARKERS	4-10
4.7 TRANSFORM TABLE	4-11
4.8 QUANTIZATION TABLE.....	4-12
4.9 HUFFMAN TABLES	4-14
4.10 RESTART INTERVAL	4-15
4.11 WSQ IMAGE DATA	4-16
5. SUMMARY OF RISKS.....	5-1
6. ACRONYMS	6-1
7. REFERENCED DOCUMENTS	1

LIST OF FIGURES

Figure 3-1. WSQ File Overview [2].....	3-2
Figure 4-1. SOI Marker	4-1
Figure 4-2. EOI Marker	4-1
Figure 4-3. Generic Marker Structure.....	4-2
Figure 4-4. Valid Adjacent WSQ Segments.....	4-3
Figure 4-5. Hidden Data between WSQ Segments (i)	4-3
Figure 4-6. Hidden Data between WSQ Segments (ii)	4-3
Figure 4-7. Two Example WSQ Comment Segments [3].....	4-5
Figure 4-8. Sensitive Data in Comments.....	4-5
Figure 4-9. Windows Executable in WSQ Comments.....	4-6
Figure 4-10. Frame Header Format.....	4-7
Figure 4-11. SOF Segment Example	4-8
Figure 4-12. Block Header Format	4-9
Figure 4-13. Transform Table Format	4-11
Figure 4-14. Quantization Table Format.....	4-13
Figure 4-15. Huffman Table Syntax.....	4-14
Figure 4-16. Restart Interval Segment	4-15
Figure 4-17. Original WSQ File	4-17
Figure 4-18. WSQ File with Hidden Blocks.....	4-17
Figure 4-19. WSQ Compression Cycle Effects on Fingerprint Accuracy	4-18

LIST OF TABLES

Table 1-1. Document Organization..... 1-2

Table 1-2. Recommendation Actions 1-4

Table 1-3. Recommendation Action Options 1-5

Table 4-1. Start of Frame Header Fields [1] 4-7

Table 4-2. Transform Table Fields 4-11

Table 4-3. Quantization Table Fields..... 4-13

Table 5-1. Summary of Risks 5-1

Table 6-1. Acronyms 6-1

1. SCOPE

1.1 Purpose of this Document

The purpose of this document is to provide guidance for the development of a sanitization and analysis software tool for Wavelet Scalar Quantization (WSQ) biometric files. WSQ is a compression algorithm formally defined in [1]. This document also refers to WSQ as a file type, since the data representing the entire image defined in [1] is also commonly stored within a file. This document analyzes various elements and objects that are contained within the WSQ file structure and then discusses the data hiding, data attack, and data disclosure risks. It describes how those elements can be a cause for concern for either hiding sensitive data or possibly attempting to exploit a system. This report provides numerous recommendations and mitigations that could be used to ensure the WSQ file is safer and accurately conforms to the specification.

The intended audience of this document includes system engineers, designers, software developers, and testers who work with file inspection and sanitization applications that involve processing WSQ files.

1.2 Introduction

The WSQ class of encoders decomposes the fingerprint image into a number of subbands, each of which represents information in a particular frequency band. The subband decomposition is achieved by a discrete wavelet transformation of the fingerprint image.

Each of the subbands is then quantized using values from a quantization table. There are no default values for the quantization tables. The quantized coefficients are then passed to a Huffman encoding procedure which compresses the data. The wavelet transform table, quantization table, and Huffman table data must be provided to the decoder in order to decompress the image for later processing or viewing.

1.3 Background

The FBI began keeping inked paper fingerprint impressions in 1924, and their collection has grown from an initial 810,000 cards to a present collection of over 300 million cards. Archiving this information in the form of inked impressions on paper cards has obvious

drawbacks when it comes to transmission, storage, and automated analysis of fingerprints. While bit-mapped (black/ white) facsimile scans of inked impressions were previously used to provide rapid transmission of "Post Office-grade" reproductions, the quality of facsimile digitization was not high enough to permit replacing the original cards with their facsimile scans. Nonetheless, there were many advantages to storing and transmitting fingerprint records in some type of digital format. A number of municipal and state jurisdictions implemented different commercial digital imaging systems for recording fingerprint data. This has led to compatibility problems resulting from the use of multiple, competing, proprietary hardware systems and data formats, a situation that has generated a demand for standardization in the criminal justice community. In order to improve the FBI's response time to justice system inquiries regarding criminal histories or outstanding warrants for arrested suspects prior to arraignment, the FBI developed the Integrated Automated Fingerprint Identification System (IAFIS).

With the assistance of the National Institute Of Standards and Technology (NIST) and Los Alamos National Laboratory (LANL), the Criminal Justice Information Services (CJIS) division developed standards for fingerprint digitization in cooperation with the commercial vendor and criminal justice communities.

The CJIS/LANL/NIST team discovered that fingerprint image files could be compressed using the WSQ compression technique at a compression ratio of 15:1 without introducing blocking artifacts that are created by other compression methods. WSQ preserves fine fingerprint details such as fingerprint friction ridges, minutiae, and sweat pores to support the fingerprint identification process.

1.4 Document Organization

The following table summarizes the organization of this document.

Table 1-1. Document Organization

Section	Description
Section 1: Scope	This section describes the scope, organization, and limitations of this document.
Section 2: Construct Overview	This section describes construct information for WSQ.
Section 3: WSQ Overview	This section describes an overview of WSQ

Section	Description
Section 4: WSQ - Constructs	This section describes the details of WSQ and provides constructs per marker and section in the file.
Section 5: Summary of Risks	Contains a table summary of the risks identified in this document.
Section 6: Acronyms	This section lists the acronyms that appear in this document.
Section 7: Referenced Documents	This section lists the sources used to prepare or cited in this document.

1.5 Recommendations


The following subsections summarize the categories of recommendation actions that appear in this document and associated action options.

1.5.1 Actions

Each construct description lists recommended actions for handling the construct when processing a document. Generally, inspection and sanitization programs will perform one or more actions on a construct: *Validate, Remove, Replace, External Filtering Required, Review, or Reject*.

The Recommendation section for each construct lists each of these actions and corresponding applicable explanations of the action to take. It notes if a particular action does not apply, indicates actions that are not part of the standard set of actions (listed in the previous paragraph). For example, a program may choose to reject a file if it is encrypted. Additionally, for some constructs, an action may further break down to specific elements of a construct (e.g., for hidden data in image file formats, the recommendations for Replace could include an action for decompressing and recompressing the image data again or decompressing then manipulating the image content, recompress, and then replace the entire image data) to give administrators the flexibility to handle specific items differently.

NOTE



The recommendations in this document are brief explanations rather than a How-To Guide. Readers should refer to the construct description or WSQ

Gray-Scale Fingerprint Image Compression specification for additional details.

Table 1-2 summarizes the recommendation actions.

Table 1-2. Recommendation Actions

Recommendation Action	Comments
<i>Validate</i>	Verify the data structure’s integrity, which may include integrity checks on other components in the file.(This should almost always be a recommended action)
<i>Replace</i>	Replace the data structure, or one or more of its elements, with values that alleviate the risk (e.g., replacing a user name with a non-identifying, harmless value, or substituting a common name for all authors).
<i>Remove</i>	Remove the data structure or one or more of its elements and any other affected areas.
<i>External Filtering Required</i>	Note the data type and pass the data to an external action for handling that data type (e.g., extract text and pass it to a dirty word search).
<i>Review</i>	Present the data structure or its constructs for a human to review. (This should almost always be recommended if the object being inspected can be revised by a human)
<i>Reject</i>	Reject the entire file.

NOTE



No recommendations for logging all actions and found data are included here because all activity logging in a file inspection application should occur “at an appropriate level” and presented in a form that a human can analyze further (e.g., the audit information may be stored in any format but must be parsable and provide enough information to address the issue when presented to a human.)

1.5.2 Action Options

The companion to this document, *Data Transfer Guidance for WSQ*, specifies four options for each recommended action: *Mandatory*, *Recommended*, *Optional*, or *Ignore*. Depending on the circumstances (e.g., a low to high data transfer versus a classified to unclassified transfer), programs can be configured to handle constructs differently.

Table 1-3 summarizes the recommendation action options.

Table 1-3. Recommendation Action Options

Action Options	Comments
<i>Mandatory</i>	For the given direction (e.g., secure private network to unsecure Internet), the file inspection and sanitization program must perform this recommended action.
<i>Recommended</i>	Programs should implement this action if technically feasible.
<i>Optional</i>	Programs may choose to perform or ignore this recommended action.
<i>Ignore</i>	Programs can ignore this construct or data structure entirely

1.6 Data Transfer Guidance (DTG)

Each format documented for inspection and sanitization programs has a companion document (i.e., the aforementioned DTG document). The DTG serves as a checklist for administrators and others to describe expected behaviors for inspection and sanitization programs. For instance, an administrator may decide to remove a particular segment in an image or it could elect to replace image segment data.

The *DTG* gives the administrator the flexibility to specify behaviors for inspection and sanitization programs. The workbook contains a worksheet for each security domain (i.e., the originating domain). Each worksheet lists the numbered constructs from this document and enumerated recommendations in a row. After the recommendations, the worksheet displays a cell for each possible destination domain. This enables an administrator to select the action option for data transfer from the originating domain to the particular destination domain. Each construct row also contains two comment cells: one for low to high transfers and another for high to low transfers.

The recommended actions address three broad risk types: data hiding, data attack, and data disclosure. Most data structures are vulnerable to one risk type, while others are susceptible to all risk types. Each construct row in the DTG worksheet contains a cell for designating the risk type (i.e., data attack, data hiding, data disclosure, or all three) and another cell for assessing the risk level for that construct (i.e., high, medium and low). This enables administrators to assign the risk type and risk level to each specific construct.

1.7 Document Limitations

1.7.1 Covert Channel Analysis

It is impossible to identify all available covert channels, whether in a file format or a communications protocol. Since WSQ files contain free-form text and binary content for compressed image data, searching for hidden data becomes increasingly difficult. No tool can possibly analyze every channel, so this document highlights the highest risk areas to reduce or eliminate data spills and malicious content.

Additionally, this document does not discuss steganography within block text or media files, such as a hidden message that is embedded within an innocuous image or paragraph. Separate file format filters that specialize in steganography should handle embedded content, such as text, images, videos, and audio.

2. CONSTRUCT OVERVIEW

2.1 Constructs

Although this document delves into many low level constructs in the WSQ syntax, it does not serve as a complete reference for all of the different constructs in the specification. This document focuses on particular areas of concern for developers of file inspection and sanitization programs; however, there are additional details in the standard that should be examined alongside this documentation. The constructs discussed are defined according to the format below. For each area of concern that is mentioned, the following sections exist:

- **Description:** a high level explanation of the data structure or element including an example, if applicable.
- **Risks and Recommendations:**
 - **Risks:** An explanation of potential problems posed by the element is provided. For example, some metadata elements can cause inadvertent data leakage and others can be used for data exfiltration. A specific recommendation that follows can be used to address that concern.
 - **Recommendations:** As defined in Table 1-2.
- **Product:** provides the version or specification number where this construct exists.
- **Location:** provides a textual description of where to find the element in the document.

Recommendations are provided for each of the WSQ constructs. For the purposes of this document, these recommendations are “alternatives.” Some recommendations may be more applicable, given a specific implementation, than others and some recommendations may be more difficult to implement. Certain recommendations complement each other and can be grouped together (e.g., “Replace field with default value” and “Pass free text to external filter”). Other recommendations may seem contradictory (e.g., “Remove field” and “Replace field”).

3. WAVELET SCALAR QUANTIZATION OVERVIEW

Wavelet Scalar Quantization (WSQ) data is defined by the specification as a “compressed data format consisting of an ordered collection of parameter markers and entropy-coded data segments” [1]. Each data segment is defined as a marker segment, and numerous markers are defined in the specification and highlighted in the next section of this paper. The WSQ specification does not explicitly define a file structure; it defines an interchange format for organizing a block of data containing an image. Most, if not all, WSQ files will contain a single image which maps to one instance of the image data structure. Often, the term file may be used to describe a file containing one instance of the image data structure. In this report, the term image viewer is defined as an application that parses the image interchange format and decompresses the image data, rendering the image to the user.

WSQ data is defined by a series of markers. These markers are defined by two bytes; the first byte is always 0xFF and the second byte is a unique value that defines the specific marker. WSQ data is primarily identified by the very first marker (its magic number) which is the Start of Image (SOI) marker and the image data is terminated by the End of Image (EOI) marker. In between those two markers data can appear in a similar manner as illustrated in Figure 3-1 below.

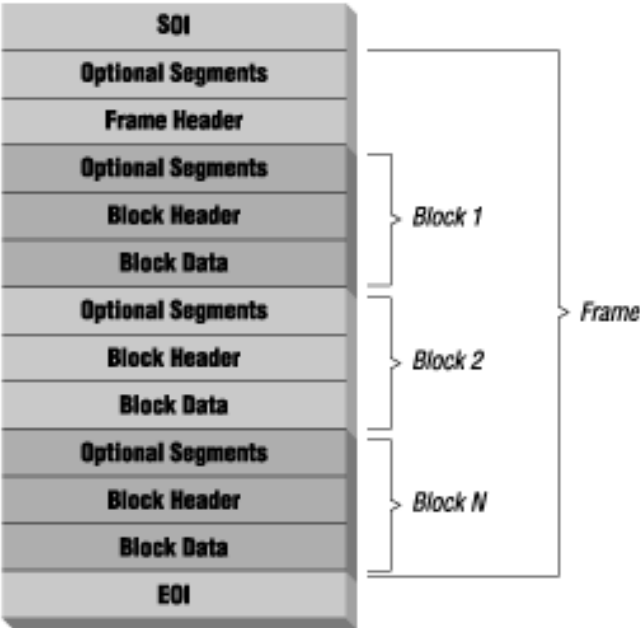


Figure 3-1. WSQ File Overview [2]

A WSQ image consists of a single frame defined by the Start of Frame (SOF) marker, which is the first two bytes of the Frame Header block in Figure 3-1. A WSQ file also contains numerous blocks of data, introduced by the Start of Block (SOB) marker, the first two bytes in the Block Header section in Figure 3-1. Block data follows the Block Header as shown in Figure 3-1.

Optional segments are Comments, Transform Tables, Quantization Tables, and Huffman Tables. They are individually defined in their own marker segment throughout various locations in the image data. Optional segments may exist between the SOI marker and SOF marker. The second location is in the Block Segment before each block of data and SOB marker. In the specification, the optional segments at this location are considered part of that block as shown in Figure 3-1.

3.1 WSQ Character Encoding

WSQ is a binary file with many fields and markers that are of fixed size and byte order. The specification does not mention a character encoding but does mention that numerical values are stored as big endian. Some fields indicate that the data sequence is

ordered from left to right, top to bottom. Text may exist within WSQ files but is only described as a block of individual bytes that can be interpreted by the application in any manner.

4. WSQ CONSTRUCTS

The WSQ specification defines an interchange format, which defines the structure and organization of a block of data that contains the image. The specification does not mention anything about a file, but rather a block of data containing the image. This block of data can be placed within a 'WSQ' file; therefore, when a file is mentioned in this section, it refers to a block of image data. The interchange format for WSQ is similar to other image formats. WSQ implements a number of markers which identify different sections of the data. File inspection software should be implemented to inspect each marker of the file, including the data that resides within each marker. This section highlights all of the marker sections and its corresponding data.

4.1 Start of Image/End of Image Markers

DESCRIPTION:

WSQ is organized by a series of markers. The image begins with the Start of Image (SOI) marker, which is defined as 0xFFA0. The End of Image (EOI) marker is the last two bytes of the file, terminating the file. The value for the EOI is 0xFFA1. The specification is not clear on whether there could be multiple SOI and EOI markers since the WSQ specification does not necessarily define a file structure. The specification seems to indicate that there could be multiple images because it states that "once a [Huffman/Quantization] table has been defined, it may be used for subsequent images." Many applications may treat a WSQ file as a single block of data with one SOI and one terminating EOI. Others could treat a single file with multiple images (an SOI following the previous EOI) correctly.

The following example illustrates the first 2 bytes of a valid WSQ file:

Address	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	Dump
00000000	ff	a0	ff	a8	00	7b	4e	49	53	54	5f	43	4f	4d	20	39	ÿ.ÿ".{NIST_COM 9
00000010	0a	50	49	58	5f	57	49	44	54	48	20	35	34	35	0a	50	PTX WIDTH 545 P

Figure 4-1. SOI Marker

The following example illustrates the final 2 bytes of a valid WSQ file:

00005c90	3c	6b	1f	16	63	e8	01	0f	7c	67	4f	a0	04	04	fa	2b	<k..cè... gO...ú+
00005ca0	cb	ff	00	ff	a1												ÿÿ.ÿi

Figure 4-2. EOI Marker

RISKS AND RECOMMENDATIONS:

Data Attack and Data Hiding: Files that do not conform to the specification may be the source of an attack or hiding risk.

- 1 **Validate:** Check that the first two bytes of the file are 0xFFA0 and the final two are 0xFFA1.
- 2 **Validate:** Check that the SOI and EOI markers are not nesting (A SOI marker should not follow another SOI unless there is an EOI marker before the next SOI).
- 3 **Validate:** If the application supports only a single image per ‘WSQ’ file, check that there is only one of each SOI and EOI marker in the entire file.
- 4 **Validate:** If the application supports multiple WSQ images in a single file, ensure that each WSQ block (data between SOI and EOI) undergoes separate filtering.
- 5 **Remove:** Remove all extraneous data prior to the SOI marker and following the EOI marker. In the case of multiple SOI/EOI pairs, remove extraneous data between EOI and the subsequent SOI.
- 6 **Reject:** Reject a file with nested SOI/EOI pairs.
- 7 **Reject:** Reject the file if the SOI is not the first two bytes of the file or if the EOI is not the final two bytes of the file.
- 8 **Reject:** If the application supports only a single image per ‘WSQ’ file, reject files with multiple SOI and EOI markers.

PRODUCT: WSQ

LOCATIONS:

The first two bytes of the file should be the SOI and the last two bytes of the file should be the EOI marker.

4.2 Generic Marker Segments

DESCRIPTION:

A WSQ file is comprised of marker segments. The structure for a marker segment is a 2-byte marker, typically followed by a Length field, and then by data as illustrated below.



Figure 4-3. Generic Marker Structure

This construct section focuses on the issues related to all marker segments and provides some general recommendations to be applied to every marker segment found within WSQ. The first issue is determining

that only valid data exists in the segment data. In a valid WSQ structure, the next segment marker would immediately follow the previous segment’s valid segment data. This is shown in the figure below.



Figure 4-4. Valid Adjacent WSQ Segments

A filter examining WSQ data needs to check that the length matches the actual length of the segment data. If there is structure to the segment data defined by the specification then this needs to be validated as well. The filter should also ensure the the next segment marker follows the validated segment data of the previous marker and that no data exists in between as shown in the next figure. In the figure below, the length represents the length of the actual data, but the next marker does not follow the last byte of the segment data. This data might go undetected in some applications. This could also be a problem for an application allocating memory, since the length is incorrect and does not reflect all the data prior to the next segment marker.

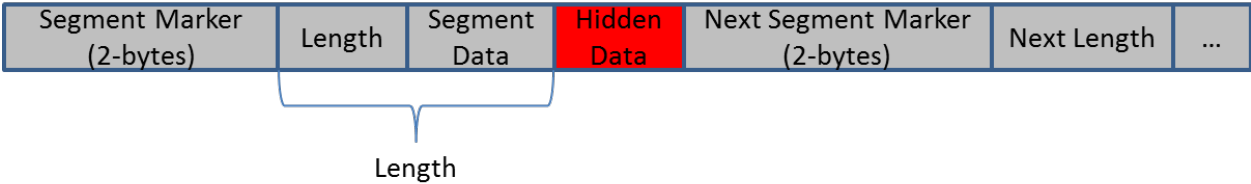


Figure 4-5. Hidden Data between WSQ Segments (i)

The length field could be modified such that the next segment marker is found after ‘Length’ bytes following the previous marker. An example of this is shown below.

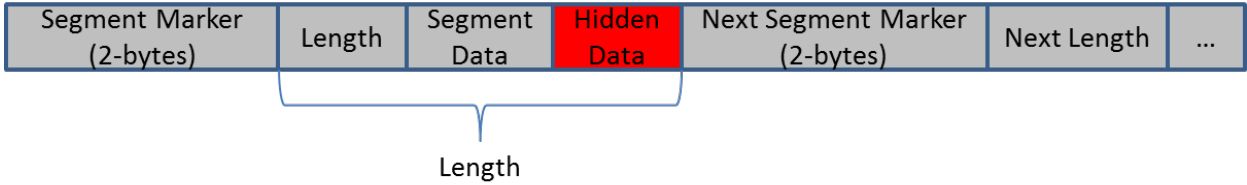


Figure 4-6. Hidden Data between WSQ Segments (ii)

The above example demonstrates that the segment data structure needs to be validated and its length verified against the provided value.

RISKS AND RECOMMENDATIONS:

Data Attack and Data Hiding: Files with invalid segment lengths or data following the ‘claimed’ data length could be the source of a hiding risk or an attack risk.

- 1 **Validate:** Check all marker segment 2-byte values are on a whitelist from the WSQ specification.
- 2 **Validate:** Check all the Length fields so that they both reflect the validated segment data length and immediately following is a valid marker segment for the next segment.
- 3 **Validate:** Check each segment data block (defined in the next sections in this report) is structurally correct according the WSQ specification.
- 4 **Remove:** Remove hidden data in the case shown in Figure 4-5. The length field is accurate without that data.
- 5 **Reject;** Reject files with invalid lengths, invalid segment structure, or invalid data between segments.

PRODUCT: WSQ

LOCATIONS:

The data discussed in this construct directly follows any marker segment throughout the file.

4.3 Comment Segments

DESCRIPTION:

WSQ implements a Comment Marker Code, defining an optional segment with comments. There could be numerous Comment Segments throughout the WSQ structure. Content following this Comment Marker (0xFFA8) is not in any specific format, just simple fixed number of bytes until the next valid marker. The following example illustrates a Comment Marker following the Start of Image Marker (0xFFA0). It also shows a second Comment Marker following the first block of comments.

The WSQ Comment Segment is introduced with the marker 0xFFA8. The next field is the Lc field, a 2 byte field indicating the segment length. The segment length includes the 2-byte length field plus the length of the comment. The minimum value for the Comment Length field is 2, which means there is no comment, and the next byte must be a valid marker. The maximum value is 65,535.

Address	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	Dump
00000000	ff	a0	ff	a8	00	7b	4e	49	53	54	5f	43	4f	4d	20	39	ÿ.ÿ".{NIST_COM 9
00000010	0a	50	49	58	5f	57	49	44	54	48	20	35	34	35	0a	50	.PIX_WIDTH 545.P
00000020	49	58	5f	48	45	49	47	48	54	20	36	32	32	0a	50	49	IX_HEIGHT 622.PI
00000030	58	5f	44	45	50	54	48	20	32	34	0a	50	50	49	20	35	X_DEPTH 24.PPI 5
00000040	30	30	0a	4c	4f	53	53	59	20	31	0a	43	4f	4c	4f	52	00.LOSSY 1.COLOR
00000050	53	50	41	43	45	20	47	52	41	59	0a	43	4f	4d	50	52	SPACE GRAY.COMPR
00000060	45	53	53	49	4f	4e	20	57	53	51	0a	57	53	51	5f	42	SSION WSQ.WSQ_B
00000070	49	54	52	41	54	45	20	30	2e	37	35	30	30	30	30	ff	ITRATE 0.750000ÿ
00000080	a8	00	0a	43	6f	67	6e	61	78	6f	6e	ff	a4	00	3a	09	"..Cognaxonÿ".:..
00000090	07	00	09	32	d3	25	cd	00	0a	e0	f3	19	9a	01	0a	41	...2Ó%í...à6.š...A
000000a0	ef	f1	9a	01	0b	8e	27	64	cd	00	0b	e1	79	a3	33	00	ĩñš...Ž'dí...áy£3.
000000b0	09	2e	ff	56	00	01	0a	f9	33	d3	33	01	0b	f2	87	21	..ÿV...ù3Ó3...ð‡!

Figure 4-7. Two Example WSQ Comment Segments [3]

In the example above, the length is 0x007b, or 123 bytes. The next marker is at byte offset 0x007f, and is another Comment Marker, 0xFFA8, with a length field of 0x000A, or 10 bytes. Following the length field is the actual comment itself, which could be in any format (including potentially executable binary code) and can be interpreted by the application in any manner. The next example shows how hidden data could be inserted into the comment segment data.

Address	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	Dump
00000000	ff	a0	ff	a8	00	7b	54	48	45	52	45	20	43	4f	55	4c	ÿ.ÿ".{THERE COUL
00000010	44	20	42	45	20	53	45	4e	53	49	54	49	56	45	20	49	D BE SENSITIVE I
00000020	4e	46	4f	52	4d	41	54	49	4f	4e	20	4c	4f	43	41	54	NFORMATION LOCAT
00000030	45	44	20	48	45	52	45	20	49	4e	20	54	48	45	20	43	ED HERE IN THE C
00000040	4f	4d	4d	45	4e	54	20	53	45	47	4d	45	4e	54	20	31	OMMENT SEGMENT 1
00000050	32	33	34	35	36	37	38	39	30	20	4d	4f	52	45	20	53	234567890 MORE S
00000060	45	4e	53	49	54	49	56	45	20	44	45	54	41	49	4c	53	ENSITIVE DETAILS
00000070	2e	2e	2e	2e	2e	2e	2e	2e	2e	2e	2e	2e	2e	2e	2e	ffÿ

Figure 4-8. Sensitive Data in Comments

Comments could be any type of data, as long as it is under 65,533 bytes. In the following example, a Windows executable was inserted into the comments, the length was adjusted to match, and the file still opened correctly as the application ignores the comment. The following image shows an executable in the comment segment. The blue box below shows the Comment Segment Marker, 0xFFA8. The green box is the length field (0x7C02). The next two bytes ('MZ' or 0x4D5A) are the start of a windows executable.

```
00005ba0 8e 1f 40 0f 19 f4 00 88 3e 31 05 3e 36 2f d0 04 ž.0...ô.^>1.>6/D.
00005bb0 f4 78 c7 be 31 ef b1 f1 68 3e 80 20 47 c7 39 fd ôxÇ³4lî±ñh>€ GÇ9ý
00005bc0 e7 c5 d5 fa 00 4f cf ec 7f b1 f1 66 3d 07 d0 03 ċĂŌú.Ōîi.±ñf=.Đ.
00005bd0 24 f8 bd 30 fa 00 8e 5f 1b a3 c6 1c fe a3 e8 01 $ø½Ōú.ž_.fÆ.pfè.
00005be0 40 3f 74 fe 27 bc f1 87 3c 6a 1f a0 06 69 f1 77 @?tþ'¼ñ+<j...iñw
00005bf0 30 78 c1 9e 3e cf d0 02 78 9f 40 0c 73 e2 ce f8 0xĂž>İĐ.xŸ0.sâİø
00005c00 1e 3e e7 8d bb fd ac 3c 6b 1f 16 63 e8 01 0f 7c .>ç.»ý~<k..cè..|
00005c10 67 4f a0 04 04 fa 2b cb ff 00 ff a8 7c 02 4d 5a gŌ...ú+ĚŸ.Ÿ"|.MZ
00005c20 90 00 03 00 00 00 04 00 00 00 ff ff 00 00 b8 00 .....ýŸ...
00005c30 00 00 00 00 00 00 40 00 00 00 00 00 00 00 00 .....@.....
00005c40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00005c50 00 00 00 00 00 00 00 00 00 00 80 00 00 00 0e 1f .....€. ....
00005c60 ba 0e 00 b4 09 cd 21 b8 01 4c cd 21 54 68 69 73 °..'.í!|.Lí!This
00005c70 20 70 72 6f 67 72 61 6d 20 63 61 6e 6e 6f 74 20 program cannot
00005c80 62 65 20 72 75 6e 20 69 6e 20 44 4f 53 20 6d 6f be run in DOS mo
00005c90 64 65 2e 0d 0d 0a 24 00 00 00 00 00 00 00 50 45 de....$. ....PE
00005ca0 00 00 4c 01 05 00 84 cc d7 4b 00 00 00 00 00 00 ..L...„İ×K.....
00005cb0 00 00 e0 00 0f 03 0b 01 02 38 00 5c 00 00 00 1c ..à.....8.\....
```

Figure 4-9. Windows Executable in WSQ Comments

Comments may provide a way to perform padding for alignment purposes [2], but it is more common for the image data to contain 0xFF as filler. Comments may exist in the file with the only purpose for alignment.

RISKS AND RECOMMENDATIONS:

Data Attack, Data Hiding, and Data Disclosure: The content of a comment field does not have a strict structure so it could be malicious payload or an executable instead of valid comments. A comment could also contain hidden sensitive information since it does not have an impact on the viewing application. Sensitive data could be accidentally or purposely placed at this location.

- 1 **Replace:** Replace the Comments content with all 0xFF to remove any potential sensitive information.
- 2 **Remove:** Remove the entire comment marker and contents until the next valid marker.
- 3 **External Filtering Required:** Extract the comments and pass to an external filter. This may introduce risk since the encoding of comments is not defined in the specification; it will be a series of bytes.

PRODUCT: WSQ

LOCATIONS:

A comment segment could exist in any table or miscellaneous marker portion of the file. A comment marker may exist as the last marker before the EOI.

4.4 Start of Frame Segment

DESCRIPTION:

A start of frame (SOF) segment is identified by the marker 0xFFA2. There is one Frame per image, so the SOF must follow the SOI and exist before the EOI. There is no End of Frame marker. Optional Segments (e.g., Coding Tables, Comments) may reside before the SOF marker. The SOF segment includes a Frame Header, followed by a number of blocks. Each block may contain Optional Segments (e.g., Coding Tables, Comments) and then a Block header with data. There could be numerous blocks that follow the start of frame.

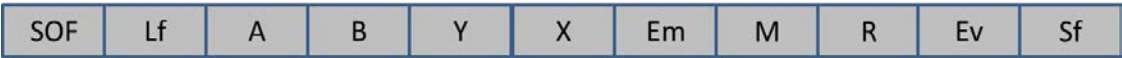


Figure 4-10. Frame Header Format

The following table defines the fields in the frame header [1].

Table 4-1. Start of Frame Header Fields [1]

Frame Header Field	Description	Size (bytes)
SOF	Start of Frame Marker – 0xFFA2	2
Lf	Frame Header Length (Length of Figure above, should be equal to 0x0011)	2
A	Scanner Black Calibration Value	1
B	Scanner White Calibration Value	1
Y	Number of lines in image	2
X	Number of samples per line in image	2
Em	Scale Exponent for M field: the decimal point in M is shifted left Em places.	1
M	Location value for image transformation parameters.	2
Er	Scale Exponent for R field: the decimal point in R is shifted left Er places.	1
R	Scale value for image transformation parameters.	2
Ev	WSQ Encoder Algorithm	1
Sf	Software that encoded the image	2
Total Size		19
Total Size minus Start of Frame Marker (Lf)		17 (0x0011)

Invalid values for fields in this header may cause problems. The number of lines in image and the number of samples per line in image must be correct. If those values are negative or incorrect it may cause a memory problem in applications and may cause an application to crash. In some other fields in this header, if the values are negative or incorrect the application may not crash, but it may not render the

fingerprint image or it may alter the visibility of the image. For example, if the Em field in a valid WSQ file is changed from 0x02 to 0xFE, the image becomes very dark and difficult to see. If the value in the Er field is changed from 0x04 to 0xFE, then the application will not display the fingerprint image at all.

An example of a SOF segment is shown in the image below. The marker is identified as 0xFFA2 and the length is correct with a value of 0x0011. The number of lines in the image is 0x026e and the number of samples per line is 0x0221.

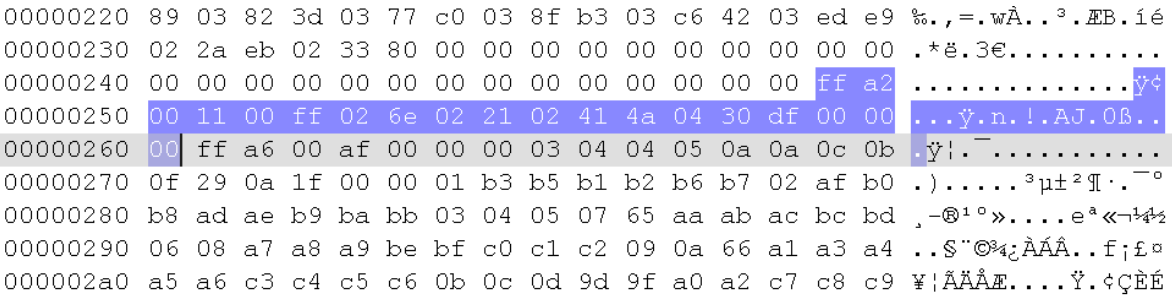


Figure 4-11. SOF Segment Example

RISKS AND RECOMMENDATIONS:

Data Hiding and Data Attack: Multiple SOF markers could mean additional data that is not shown in some image viewers (image decompressors). Invalid image description information about the image size may also cause portions of the image not to be shown or cause memory issues for parsers.

- 1 **Validate:** Check that there is only one SOF marker/header in the entire file.
- 2 **Validate:** Check that the Number of Lines per image and the Number of Samples per Line are valid according to the image data present in the rest of the file.
- 3 **Validate:** Check that the scale parameters are within a range that the image is still viewable.
- 4 **Reject:** Reject single image files (SOI/EOI pair data) with multiple SOF markers between.

Data Hiding: If the field R in the Start of Frame equals zero, no image is displayed in certain viewers.

- 5 **Validate:** Check that the R field (scale value for image transformation parameters) is non-zero.
- 6 **Reject:** Reject files with a non-zero R field in the SOF header.

Data Hiding and Data Attack: Invalid length fields may target parsers handling the SOF header. There could be extra data following the header.

- 7 **Validate:** Check the length field Lf is equal to 0x0011 (17 bytes).
- 8 **Remove:** If found, remove data following the SOF segment until the next valid marker.

9 **Reject:** Reject files with an invalid length field in the SOF segment or contain arbitrary data following the SOF segment.

Data Hiding: Images without a frame are non-standard and may have hidden data in other locations in the file.

10 **Reject:** Reject files without a SOF marker, there must be one in the file.

PRODUCT: WSQ

LOCATION:

The Start of Frame segment follows the SOI marker, but there could be tables and comments between the SOI marker and the SOF marker. It should be identified by the marker value and once a frame is defined, blocks of data will follow.

4.5 Start of Block Segments

DESCRIPTION:

There are multiple block segments that follow the Start of Frame segment discussed in the previous section. In other words, a frame consists of a number of blocks. Quantization, Huffman Coding, and Transform Tables may also appear within the frame in addition to blocks. A block consists of a block header, a short three field data structure, which contains the Start of Block (SOB) marker: 0xFFA3. The Block header is illustrated in the figure below. The Ls field is a length field of the entire block header minus the SOB marker. This value is always equal to 0x0003 since the Block header is a fixed size. The Td field is the Huffman coding table selector. There could be up to eight entropy coding tables, so this value must be between 0 and 7.



Figure 4-12. Block Header Format

Following the Block header are entropy coded data segments (ECS) until the next valid marker. ECS is discussed as its own construct.

RISKS AND RECOMMENDATIONS:

Data Hiding: WSQ images without blocks do not contain any actual data, and the file could just be comments or tables of information without displaying image content. Images with an invalid Td field may also not display an image.

- 1 **Validate:** Check that the SOB marker is within a SOF segment.
- 2 **Validate:** Check that the Td field is first within range of 0-7 and also points to a valid Huffman coding table present in the file.

Data Attack and Data Hiding: Any invalid length or invalid portion of the SOB segment may introduce a data attack and possible data hiding and data is skipped over when parsing.

- 3 **Validate:** Check that the `Ls` field is equal to 0x0003.
- 4 **Remove:** Remove any block and subsequent blocks after an invalid header is found (everything until the EOI marker).
- 5 **Remove:** If found, remove any additional data following the SOB segment that is not a valid part of the next segment.
- 6 **Reject:** Reject files with invalid block headers in any of the blocks of data.

PRODUCT: WSQ

LOCATION:

Block segments must be located within the single Frame Segment; there could be multiple block segments which are identified by the SOB marker.

4.6 Entropy Coded Segments and Restart Markers

DESCRIPTION:

Entropy Coded Segments (ECS) follow the Block Header; there could be numerous segments per block. Handling each block, one at a time may not be ideal for filters, as the image data itself should be examined in its raw pixel form; this is covered in later constructs. Each ECS is separated by a Restart Marker (RST_m). RST_m markers are special values that can range from 0xFFB0 – 0xFFB7. The second byte of each RST_m marker increases (modulo 8) throughout the block. Restart markers are used to separate two restart intervals in an image, restart intervals are covered later in their own construct.

ECS data contain the actual image data. A reserved 2-byte marker should not appear within this block of data. Malicious software could improperly insert markers in the image data which could cause some parsers to malfunction. The specification states that “to ensure a marker does not occur within an entropy-coded segment any 0xFF byte generated by the Huffman encoder is followed by a ‘stuffed’ zero byte.”

RISKS AND RECOMMENDATIONS:

Data Attack: Invalid markers may cause the parser to become out of synchronization and could lead to issues. Missing data may also be the sign of a crafted file targeting an image viewer.

- 1 **Validate:** Check that the RST_m increases modulo 8 and that there is no missing RST_m markers throughout each block.
- 2 **Reject:** Reject files with missing markers.
- 3 **Reject:** Reject files with markers embedded in the image data block.

4 **Reject:** Reject files that do not contain the full data needed for decompression.

Data Hiding: If all the ECS blocks are not used then they may contain hidden data.

5 **Validate:** Go through the Huffman decoding process to read the image. When all the data expected for the image size has been read, anything left over is hidden data. Validate that the total ECS sizes match what is needed for decoding.

6 **Remove:** Remove any left over hidden data that is not part of the ECS that is used for decoding.

PRODUCT: WSQ

LOCATION:

ECS and RSTm are both located following a Block header.

4.7 Transform Table

DESCRIPTION:

Transform tables reside in the Optional Segments area of a file, which may reside before the Start of Frame or the Start of Block. The segment is denoted by the Define Transform Table (DTT) marker. The value of this marker is equal to 0xFFA4. A graphical illustration of the table is shown below.



Figure 4-13. Transform Table Format

The following table describes each field in the Transform Table.

Table 4-2. Transform Table Fields

Field	Description	Size (bytes)
DTT	Define Transform Table: 0xFFA4	2
Lt	Equal to the length of the entire table (variable).	2
L0	Equals the number of analysis low-pass filter coefficients (h_0).	1
L1	Equals the number of analysis high-pass filter coefficeients (h_1).	1
Sn _k	Sign of the k th low-pass filter coefficient.	1
Ex _k	Scale exponent for the k th low-pass filter coefficient.	1
H0 _k	Low-pass analysis filter element.	4
Sn _k	Sign of the k th high-pass filter coefficient.	1
Ex _k	Scale exponent for the k th high-pass filter coefficient.	1
H1 _k	High-pass analysis filter element.	4

Total Size minus Define Transform Table Marker (Lt)	4 + 6(L0+L1)
---	--------------

RISKS AND RECOMMENDATIONS:

Data Hiding: A table not within a valid location in the file (this should also be detected with other constructs) could introduce hidden data.

1 **Validate:** Check that the Transform Table is within the optional sections of the file.

Data Attack and Data Hiding: Invalid length fields and dimensions of the table could introduce problems for an image parser, or hide data beyond the actual coefficient information.

2 **Validate:** Check that the L_t Field equals the length of the entire transform table.

3 **Validate:** Check that the values in L0 and L1 are accurate with the L_t Field, $L_t = 4 + 6 (L_0 + L_1)$, since each value in L0 and L1 equate to an entry in the table for low-pass or high-pass filter coefficients.

4 **Replace:** Replace incorrect length field with correct value and remove excess content.

Data Hiding – Only one Transform table is relevant in a WSQ image. Additional DTT segments may be an attempt to hide information within the file.

5 **Validate:** Check that only one DTT marker segment is present in the image.

6 **Validate:** Check all DTT segments against known fixed DTT segments specified by FBI WSQ 1.0 (1997) and FBI WSQ 3.0/3.1 (2010).

7 **Remove:** Remove all DTT segments except for the last DTT segment.

PRODUCT: WSQ

LOCATION:

The transform may reside in any of the optional sections of the file (after the Start of Image but before Start of Frame), or within any of the blocks. It is designated by the DTT marker (0xFFA4).

4.8 Quantization Table

DESCRIPTION:

The Quantization table can appear in any of the optional segment sections in the WSQ file. The table is included by the Define Quantization Table (DQT) marker, which is defined in the specification as 0xFFA5. An illustration of the table is shown in the figure below.

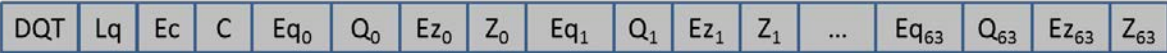


Figure 4-14. Quantization Table Format

The Quantization table includes the following fields:

Table 4-3. Quantization Table Fields

Field	Description	Size (bytes)
DQT	Define Quantization Table (DQT): 0xFFA5	2
Lq	The length of the entire quantization table.	2
Ec	Scale exponent for the field C	1
C	Quantizer bin center parameter	2
Eq _k	Scale exponent for Q _k	1
Q _k	Quantization table elements.	2
Ez _k	Scale exponent for Z _k	1
Z _k	Zero bin table elements.	2
Total Size minus Define Quantization Table Marker (Lq)		5 + 64(6) = 389 = 0x0185

Disrupting the DQT segment by adding text in place of an actual table often results in an image that is not viewable, or a blank image according to some WSQ viewers. Changing some of the table elements and scale exponents often renders an image unviewable. The values in the DQT are critical to the decompression process; however, the specification allows the entry of more than one DQT. Only the last one read is used for computation. The earlier ones could contain hidden info with no impact on the decoded image.

RISKS AND RECOMMENDATIONS:

Data Hiding: Table data at an incorrect location (within entropy coded data segments, or after all blocks of data) may introduce data hiding since the information might not be parsed.

1 Validate: Check that the DQT marker is in a valid location in the file (in the optional segment locations).

Data Hiding – If the Quantization table is not used by any block in the WSQ file then the unreferenced data may be an attempt to hide information within the file.

2 Validate: Check that only one DQT marker segment is present in the image.

3 Remove: Remove any DQT segment instances that are not referenced by DHT segments.

Data Attack or Data Hiding: Length fields that parsers rely upon for a variable block of data may introduce a vulnerability. Unused length may be used for data hiding.

4 Validate: Check the length field L_q is equal to 0x0185 (389 bytes) with a tolerance of +/- 4 bytes (in case of word alignment).

- 5
- Replace: Replace incorrect length field with correct value and remove excess content.

PRODUCT: WSQ

LOCATION:

The Quantization segment may reside in any of the optional segment locations (before the Start of Frame but after the Start of Image) and in any of the blocks of data.

4.9 Huffman Tables

DESCRIPTION:

A Huffman table segment is introduced by the Define Huffman Table (DHT) marker segment, defined by the specification as 0xFFA6. A diagram from the specification of the Huffman table is illustrated below.

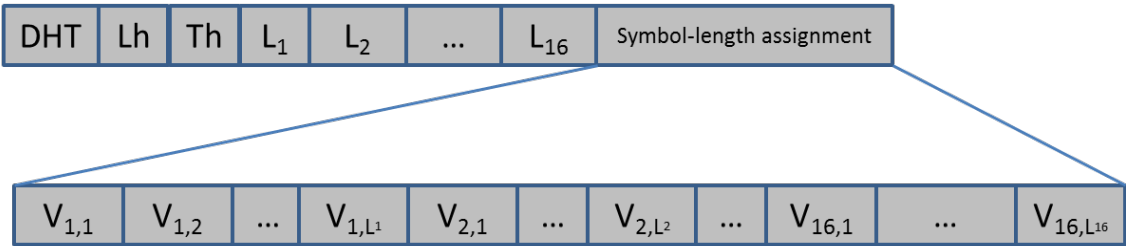


Figure 4-15. Huffman Table Syntax

The DHT segment consists of several fields including a Length field L_h , which defines the entire length of the table and everything covered in the figure above. The field T_h is a table identifier, which is referenced by a block segment elsewhere in the file. The area from T_h through end of Symbol assignment can appear multiple times (one DHT can define multiple Huffman tables).

The field L_i is a single byte field that defines the Huffman codes of length i . The fields $v_{1,1}$ through v_{ij} represent a value that is associated with each Huffman code.

The Length of a single Huffman table is denoted by H_m below, which is the size of T_h , plus the size of each L_i , plus the full length of the DHT is defined below in the L_h equation, which depends on how many Huffman tables exist (m) in the entire DHT segment.

$$H_n = 17 + \sum_{i=1}^{16} L_{i_n}$$
$$L_h = 2 + \sum_{n=1}^m H_n$$

RISKS AND RECOMMENDATIONS:

Data Attack or Data Hiding – An invalid length field may indicate the presence of a data attack risk or data hidden beyond the length.

- 1 **Validate:** Check that L_h is the actual length of the Huffman marker segment, which requires parsing until finding the next valid marker.
- 2 **Reject:** Reject files with invalid lengths for the Huffman marker segment or contains data following all valid Huffman table data.

Data Hiding – If the Huffman table is not used by any block in the WSQ file then the unreferenced data may be an attempt to hide information within the file.

- 3 **Validate:** Check that the T_h field values are referenced by T_d field values in a block segment that exists in the file.
- 4 **Validate:** Check that ECS data referenced to the T_d field exists before another DHT marker redefines table T_h .
- 5 **Remove:** Remove portion of the DHT marker that corresponds to an invalid T_h . Adjust L_h to reflect the changed size of the entire DHT marker. (If all tables within the DHT are removed, remove the DHT marker entirely.)

Data Hiding – A table placed at the wrong location could be ignored by image viewers. The entire image may not be displayed in image viewers when this happens.

- 6 **Validate:** Check that the Huffman table is in the optional segment portion of the file.

PRODUCT: WSQ

LOCATION:

A Huffman table may exist in the optional segments portion of the file, after the SOI and before the SOF and before any Block definition.

4.10 Restart Interval

DESCRIPTION:

The Restart segment is defined by the 2 byte marker, equal to 0xFFA7. Following the marker is the 2 byte Length L_r , which should be equal to 4 since the Restart segment is a fixed length. The last parameter is a 2 byte field called R_i . This field defines the number of coefficients in the restart interval.

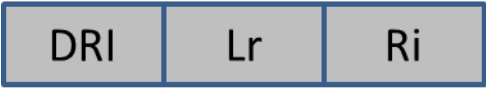


Figure 4-16. Restart Interval Segment

RISKS AND RECOMMENDATIONS:

Data Hiding – Multiple invalid restart segments could hide information within the file.

1 Validate: Check that this segment does not follow the last block of entropy coded data segments, as this is not allowed.

2 Remove: Remove restart segments that follow the last block of data.

Data Attack – Although the Length field should be a fixed size, it may be unlikely to be a source of data attack, all length fields should be validated for correctness in case an application uses an untrusted and incorrect value for parsing.

3 Validate: Check that the length field L_r is equal to 0x0004.

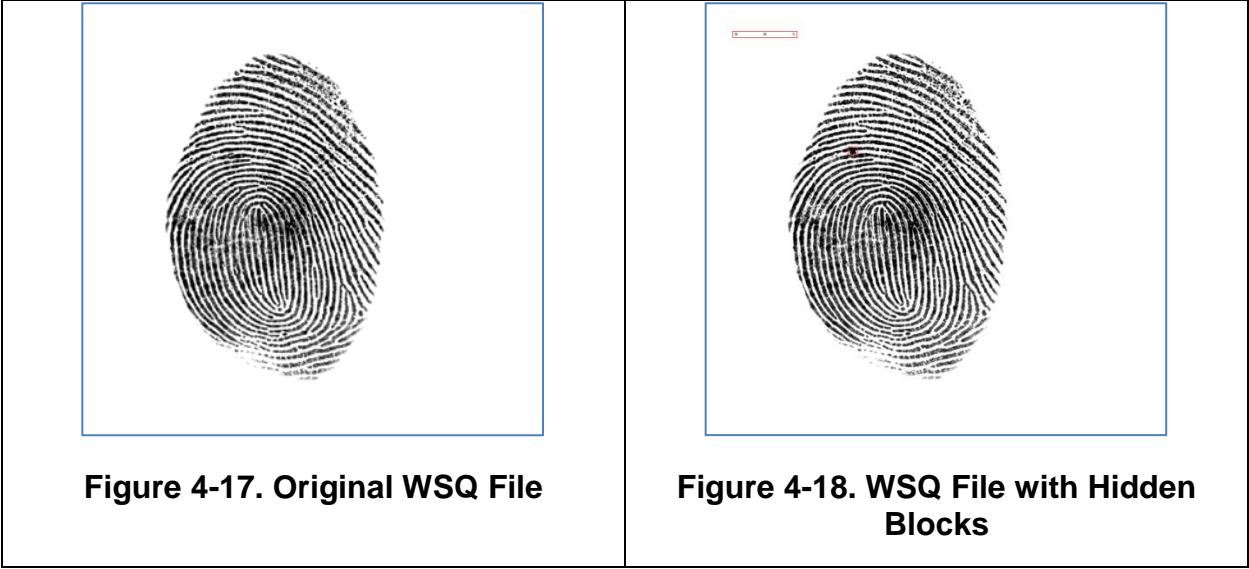
PRODUCT: WSQ**LOCATION:**

Restart interval segments will be located between block segments.

4.11 WSQ Image Data

DESCRIPTION:

The actual image data in a WSQ file can contain hidden data, and still resemble a quality fingerprint. WSQ files can be created from raw pixel files, TIFF, PNG, or other rectangular array formats describing 256 grayscale, one byte-per-pixel images, and often hidden/steganographic data can survive the compression algorithms defined in the WSQ specification. Blocks of data can be inserted into various locations within the file and not have an impact on the image quality, such as shown in Figure 4-17 and Figure 4-18. In this example, the pixels were modified to contain random data and remain present in the new WSQ file on the right. Although a user can see this small but noticeable difference, the two files will match with fingerprint software.



The WSQ compression algorithm is used to attain high compression rates while preserving image details critical to fingerprint matching; however, WSQ is a “lossy” compression. Repeated WSQ compression/decompression cycles gradually degrade the image and subsequent fingerprint matching performance.

The effects of repeated WSQ compression/decompression cycles on fingerprint matching algorithm accuracy has not been exhaustively studied. A set of operational quality fingerprint images was subjected to repeated WSQ compression/decompression cycles, and the accuracy of a commercial fingerprint detection algorithm by a commercial fingerprint algorithm vendor used in multiple states was determined for each compression cycle.

True Match Rate (TMR) is defined as the proportion of fingerprint images, acquired from “genuine” attempts, which correctly match a fingerprint image from the same finger for the same individual at a fixed threshold score. The False Match Rate (FMR) is defined as the proportion of fingerprint images, acquired from “imposter” attempts, which incorrectly match a fingerprint image from a different individual or different finger. Matching accuracy is expressed as the TMR achieved corresponding with a fixed score threshold that corresponds to a fixed FMR.

At a threshold that produced one false match in one million imposter opportunities, the TMR for the algorithm is 75.8% on a pair of images that have been subjected to one WSQ compression/decompression cycle. A second compression cycle reduced the TMR to 74.6%. Additional compression cycles result in a gradual reduction of the accuracy of the fingerprint matcher as shown in Figure 4-19.

Fingerprint matching algorithms tolerate small (+/- 2 degrees) rotations since fingerprint image capture is a manual process with small random finger orientation differences between collection events. Digital rotation of a fingerprint image is a possible method of disrupting steganographic content embedded in

the fingerprint image. However, post-collection image rotation may cause greater degradation of subsequent matcher performance.

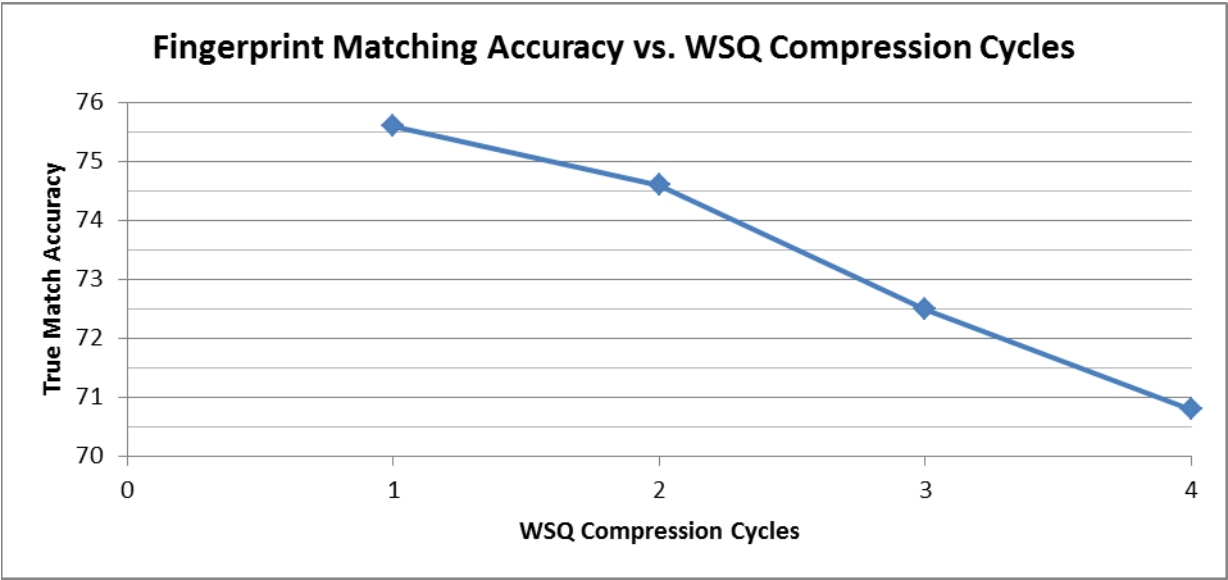


Figure 4-19. WSQ Compression Cycle Effects on Fingerprint Accuracy

RISKS AND RECOMMENDATIONS:

Data Hiding – Image data can contain hidden information with least significant pixel values in raw pixels and there may be some data remaining through compression and decompression. The WSQ compression algorithm significantly changes the lower significant bits.

- 1 Replace:** Recompress WSQ files again by decompressing to raw pixels, recompressing the raw pixels into WSQ with an appropriate 15:1 or 10:1 ratio. This process could be repeated several times to disrupt channels of hidden information, but would result in reduction of fingerprint matching accuracy.
- 2 Replace:** Image rotation by less than one degree coupled with a single WSQ compression/decompression cycle will randomize the four least significant bits of the image and disrupt rectangular position information of covert data.
- 3 Review:** Manually examine the WSQ image for data that should not exist (for example, outside fingerprint area, or anomalies within the area).

PRODUCT: WSQ

LOCATION:

The actual image data is located in Entropy-Coded data segments (ECS). A filter would need to use the decompression process to get the raw pixels.

5. SUMMARY OF RISKS

Table 5-1. Summary of Risks

Construct Section	Data Attack	Data Hiding	Data Disclosure
4.1 Start of Image/End of Image Markers	X	X	
4.2 Generic Marker Segments	X	X	
4.3 Comment Segments	X	X	X
4.4 Start of Frame Segments	X	X	
4.5 Start of Block Segments	X	X	
4.6 Entropy Coded Segments	X	X	
4.7 Transform Table	X	X	
4.8 Quantization Table	X	X	
4.9 Huffman Tables	X	X	
4.10 Restart Interval	X	X	
4.11 Image Data		X	
Sum of each risk category:	10	11	1
Total Risks:			22

6. ACRONYMS

Table 6-1. Acronyms

Acronym	Denotation
WSQ	Wavelet Scalar Quantization
SOI	Start of Image
EOI	End of Image
SOF	Start of Frame
SOB	Start of Block
DTT	Define Transform Table
DQT	Define Quantization Table
DHT	Define Huffman Table
DRI	Define Restart Interval
RST _m	Restart with modulo 8 count “m”
COM	Comment
CJIS	Criminal Justice Information Services
NIST	National Institute of Standards and Technology
LANL	Los Alamos National Laboratory
DTG	Data Tranfer Guidance
ECS	Entropy Coded Segment
ECS	Entropy Coded Data Segment
TMR	True Match Rate
FMR	False Match Rate

7. REFERENCED DOCUMENTS

- [1] Criminal Justice Information Services (CJIS). WSQ GRAY-SCALE Fingerprint Image Compression Specification, IAFIS-IC-0110(V3). December 19, 1997.
- [2] O'Reilly Formats: Online Reference available at:
<http://oreilly.com/www/centers/gff/formats/fbi/index.htm#FBI-DMYID.2>
- [3] Cognaxon – cognition for machines. Online reference available at:
<http://www.cognaxon.com/>