ISO/IEC 19794-2:2005/Cor.1:2009 Summary

Fingerprint templates » ISO 19794-2:2005

This is a complete description of fingerprint template format defined in *ISO/IEC 19794-2:2005* spec (often abbreviated *ISO 19794-2:2005* or ambiguously *ISO 19794-2*), including minor fixes described in *Technical Corrigendum 1* (*Cor.1:2009*), except for the following changes:

- Smartcard variation of this format is described separately.
- All content is distributed under permissive CC BY 4.0 license.
- Normative content was completely rewritten from scratch to avoid infringing copyrights.
- Non-normative commentary was dropped. Author added his own non-normative comments.
- Special notes were added for opensource and in-house software developers.
- Differences from other formats and other versions of this format were documented.

This format summary is complete in the sense that implementations written according to this document are also fully conforming implementations of ISO/IEC 19794-2:2005. There is no guarantee though and you should consult the original ISO/IEC 19794-2:2005 spec if in doubt.

Introduction

Human *fingerprint recognition* is usually performed in two separate steps: *feature extraction* and *matching*. Feature extraction takes fingerprint image (also called *fingerprint sample*), usually obtained from *fingerprint reader* (or *fingerprint sensor*) hardware, and produces so-called *fingerprint template* filled with fingerprint *features* useful for matching. Matching takes two fingerprint templates on input, compares fingerprint features contained in them, and produces *similarity score* on output, which is then compared to a *threshold* to produce *match* or *non-match* decision. There are many kinds of fingerprint features: *minutiae* (especially ridge *endings* and *bifurcations*), *core* and *delta* points, ridge shapes or *patterns*, orientation and ridge frequency fields, *edges* between minutiae, and many others.

Most fingerprint recognition systems have their own native fingerprint template format. Some people however insist on exchange of fingerprint templates between fingerprint recognition systems even though it is almost always a bad idea. ISO 19794-2:2005 is one of several fingerprint template formats that can be used for this purpose.

ISO 19794-2:2005 mainly encodes minutiae, including their position, angle, and type (ending, bifurcation, or other). It can optionally encode ridge counts on edges between minutiae, information about cores and deltas, and zonal image quality. It carries some fingerprint *metadata*, notably *finger position* (thumb, index, ...), *resolution*, and image size. It permits encoding several fingerprints in a single template.

Related formats

Smartcard variation of ISO 19794-2:2005 is described separately, because it is substantially different.

This template format has a new version: ISO/IEC 19794-2:2011. The two versions can be differentiated by looking at the VERSION field, which is " 20\0" in this version and "030\0" in the 2011 version.

This format shares MAGIC and VERSION fields with ANSI 378-2004. The only way to differentiate the two formats is to look at the TOTALBYTES field.

When embedded in CBEFF (*Common Biometric Exchange Framework Format*, ISO/IEC 19785-1:2006) *Biometric Data Block* (BDB), this template format will have *CBEFF Format Owner* 0x101 (decimal 257), which is registered to *ISO/IEC JTC 1 SC 37-Biometrics*. *CBEFF Format Type* registered for ISO 19794-2:2005 is 1 when there are no extension blocks and 2 when extension blocks are present, including predefined extension blocks for ridge counts, cores/deltas, and zonal quality.

Changes

ISO 19794-2:2005 is apparently derived from ANSI 378-2004. The following differences can be observed:

- TOTALBYTES has different size, which allows implementations to differentiate the two formats despite both having the same MAGIC and VERSION values.
- Fields VENDOR and SUBFORMAT are not present in this template format.
- DEVSTAMP requires certification against ISO's copy of IAFIS Image Quality Specifications rather than certification against the original FBI spec.
- Minimum resolution is 250dpi (see RESOLUTIONX).
- SAMPLETYPE doesn't have value for contactless sensor type.

- FPQUALITY references ISO 19784-1 instead of ANSI 358.
- Minutiae with MINTYPE value "other" are clearly defined to match any minutia type while wording in ANSI 378-2004 implies they should only match minutiae of the same type.
- Position (MINX and MINY) of minutiae is explicitly permitted to be implementation-defined as long as it approximates the prescribed skeleton-based method.
- Position (MINX and MINY) of minutiae with MINTYPE "other" is more precisely defined.
- MINANGLE, COREANGLE, and DELTAANGLE have different range of values.
- MINANGLE is calculated from bifurcation on single skeleton (either ridge or valley skeleton) rather than from relationship between ridge and valley skeletons.
- EXTTYPE has new value for zonal quality extension.
- EXTLEN contains length of EXTDATA instead of the whole EXTENSION.
- CORENUM may be zero.
- Core and delta positions and angles are defined (see COREX, COREANGLE, DELTAX, DELTAANGLE).
- Zonal quality extension (ZONALEXT) was added.

Layout

The template is a binary file consisting of simple fields arranged in field groups. Fields or field groups may repeat. There are no keys or tags and fields are identified only by their position in the template. Numbers are in big-endian byte order.

Repeats	Length	Code	Title
	24+ bytes		ISO/IEC 19794-2:2005 template
once	23 bytes	HEADER	Template header
once	4 bytes	MAGIC	File signature / magic number
once	4 bytes	VERSION	Format version
once	4 bytes	TOTALBYTES	Total template length in bytes
once	4 bits	DEVSTAMP	Sensor compliance
once	12 bits	DEVID	Sensor ID
once	2 bytes	WIDTH	Image width
once	2 bytes	HEIGHT	Image height
once	2 bytes	RESOLUTIONX	Horizontal pixel density
once	2 bytes	RESOLUTIONY	Vertical pixel density
once	1 byte	FPCOUNT	Number of fingerprints
once	1 byte		Reserved byte (zeroed)

FPCOUNT	6+ bytes	FINGERPRINT	Fingerprint
once	1 byte	POSITION	Finger position on hands
once	4 bits	VIEWOFFSET	Finger view number
once	4 bits	SAMPLETYPE	Impression type
once	1 byte	FPQUALITY	Fingerprint quality
once	1 byte	MINCOUNT	Number of minutiae
MINCOUNT	6 bytes	MINUTIA	Minutia
once	2 bits	MINTYPE	Minutia type
once	14 bits	MINX	Minutia X position
once	2 bits		Reserved (zeroed)
once	14 bits	MINY	Minutia Y position
once	1 byte	MINANGLE	Minutia angle
once	1 byte	MINQUALITY	Minutia quality
once	2 bytes	EXTBYTES	Total length of extension data
0 or more	4+ bytes	EXTENSION	Extension data block
once	2 bytes	EXTTYPE	Extension type
once	2 bytes	EXTLEN	Length of extension data
once	EXTLEN	EXTDATA	Extension data

There are three predefined extensions blocks. When EXTTYPE is 1, EXTDATA contains ridge count extension.

Repeats	Length	Code	Title
	3n+1 bytes	RCOUNTEXT	Ridge count extension
once	1 byte	STARTYPE	Edge picking method
0 or more	3 bytes	EDGEDEF	Edge between minutiae
once	1 byte	EDGEFROM	Starting minutia of the edge
once	1 byte	EDGETO	Ending minutia of the edge
once	1 byte	RIDGECOUNT	Number of ridges the edge crosses

When EXTTYPE is 2, EXTDATA contains core and delta extension.

Repeats	Length	Code	Title
	2+ bytes	COREDELTA	Core and delta extension
once	1+ bytes	COREDATA	Core data
once	1 byte	CORENUM	Number of cores
CORENUM	4 or 5 bytes	COREDEF	Core point

once	1 bit		Reserved (zeroed)
once	1 bit	CHASANGLE	Core angle presence flag
once	14 bits	COREX	Core X position
once	2 bits		Reserved (zeroed)
once	14 bits	COREY	Core Y position
optional	1 byte	COREANGLE	Core angle
once	1+ bytes	DELTADATA	Delta data
once	1 byte	DELTANUM	Number of deltas
DELTANUM	4 or 7 bytes	DELTADEF	Delta point
once	1 bit		Reserved (zeroed)
once	1 bit	DHASANGLE	Delta angle presence flag
once	14 bits	DELTAX	Delta X position
once	2 bits		Reserved (zeroed)
once	14 bits	DELTAY	Delta Y position
0 or 3	1 byte	DELTAANGLE	Delta angle

When EXTTYPE is 3, EXTDATA contains zonal quality extension.

Repeats	Length	Code	Title
	4+ bytes	ZONALEXT	Zonal quality extension
once	1 byte	ZONEWIDTH	Zone width
once	1 byte	ZONEHEIGHT	Zone height
once	1 byte	ZONEBITS	Bits per zone
once	variable	ZONALQUALITY	Zonal quality data

Fields and field groups

Template header (HEADER)

Header contains information common to all fingerprints in the template:

- template format identification: MAGIC, VERSION, TOTALBYTES,
- sensor information: DEVID, DEVSTAMP, WIDTH, HEIGHT, RESOLUTIONX, RESOLUTIONY, and
- number of fingerprints in the template: FPCOUNT.

It might seem strange to include TOTALBYTES as means to identify the template format, but it turns out this template format cannot be distinguished from ANSI 378-2004 without examining TOTALBYTES.

Embedding sensor information in template header means the template cannot mix fingerprints from multiple sources. Shared width and height complicates representation of multiple highly processed fingerprints, for example fused fingerprints.

Compared to ANSI 378-2004 header, TOTALBYTES has different size and fields VENDOR and SUBFORMAT are missing.

File signature / magic number (MAGIC)

First four bytes of the template contain constant string "FMR\0" where '\0' stands for zero byte. This is a magic number (or file signature) that helps biometric software to automatically distinguish this format from other file formats. Somewhat confusingly, ANSI 378 uses the same magic number.

Format version (VERSION)

Version field contains 4-byte constant string " 20\0" (note the space before "20") where '\0' stands for zero byte. Version field makes it easy to distinguish this format from its later versions.

Unfortunately, ANSI 378-2004 uses the same version string " 20\0". The only way to differentiate this format from ANSI 378-2004 is to look at the TOTALBYTES field. This is of course a totally ridiculous situation and it makes me question motivations of people designing these formats.

Total template length in bytes (TOTALBYTES)

Total length in bytes is redundant, because the template simply ends after the last FINGERPRINT section. It might be nevertheless useful to easily find the end of the template in a stream of bytes.

Template length is encoded in 4 bytes as a big-endian unsigned 32-bit number. TOTALBYTES smaller than 24 bytes is invalid, because it cannot accommodate even the template header.

Since MAGIC and VERSION are not sufficient to tell the difference between this format and ANSI 378-2004, TOTALBYTES must be used to augment format detection. In ANSI 378-2004, template length is at least 26 bytes and it is encoded in 2 or 6 bytes. The 2-byte encoding is a single big-endian unsigned 16-bit number. The 6-byte encoding consists of two zero bytes followed by a big-endian unsigned 32-bit number. The following algorithm can determine which one of the two template formats is being decoded:

1. Read two bytes and interpret them as a big-endian unsigned 16-bit number A.
2. If A is 26 or higher, the template is in the ANSI 378 format. Decoding should continue according to the ANSI 378 format. Here we assume than no template in this format will be larger than 1,703,935 bytes, which is unlikely to occur in practice.
3. Read two more bytes and interpret them as a big-endian unsigned 16-bit number B.
4. If A is in range 1-25, which is invalid in ANSI 378, the template is in this format. Template length is 0x10000 * A + B.
5. At this point, A is zero, because all other options have been already considered.
6. If B is 24 or higher, a valid length in this format, the template is assumed to be in this format and B is its length. Here we assume than no template in the ANSI 378 format will be larger than 1,572,863 bytes, which is unlikely to occur in practice.
7. Otherwise the template is in the ANSI 378 format. Decoding should continue according to the ANSI 378 format.

Sensor compliance (DEVSTAMP)

This field lives in the higher 4 bits of the byte it shares with DEVID field. Top bit indicates the fingerprint reader has certificate of compliance with Annex B of this ISO 19794-2 spec, which is a copy of Appendix F (IAFIS Image Quality Specifications) of an unknown version of FBI's Electronic Fingerprint Transmission Specification (see all versions). Other bits are reserved. Opensource implementations should leave this field zeroed.

ANSI 378-2004 defines the same field with the same structure, but it requires certificate of compliance with the original FBI document rather than ISO's copy.

Sensor ID (DEVID)

Top 4 bits of this 12-bit field live in lower 4 bits of the byte this field shares with DEVSTAMP field. Remaining 8 bits are in the next byte. The 12-bit unsigned number is an ID of the fingerprint reader used to capture fingerprints in the template. Sensor IDs are implementation-defined and they may conflict between vendors. Value of zero is allowed and it means unknown fingerprint reader. Opensource implementations should leave this field zeroed.

Image width (WIDTH)

Image width in pixels encoded as a big-endian unsigned 16-bit number.

Image height (HEIGHT)

Image height in pixels encoded as a big-endian unsigned 16-bit number.

Horizontal pixel density (RESOLUTIONX)

Image resolution (commonly called DPI), specifically its horizontal component, is stored in this field in units of pixels per centimeter. It is encoded as a big-endian unsigned 16-bit number. If sensor resolution is fractional, it is rounded to the nearest integer. Minimum resolution is 250dpi, which translates to minimum RESOLUTIONX value of 99. The common resolution of 500dpi is equal to 197px/cm after rounding.

Minimum resolution is a new requirement compared to ANSI 378-2004.

Vertical pixel density (RESOLUTIONY)

Like RESOLUTIONX above but for vertical resolution. It's usually equal to RESOLUTIONX.

Number of fingerprints (FPCOUNT)

An unsigned 8-bit number that indicates the number of FINGERPRINT blocks in the template. FPCOUNT may be zero, in which case the template contains no fingerprints. Since every fingerprint must have a unique combination of POSITION and VIEWOFFSET, there cannot be more than 11 x 16 = 176 fingerprints.

Fingerprint (FINGERPRINT)

A single *fingerprint view*, essentially a single scan from fingerprint reader. Fingerprint section contains MINUTIA records and some metadata about the fingerprint, notably finger's POSITION on hands. Fingerprint section can optionally contain EXTENSION blocks, including ridge count data (RCOUNTEXT), core/delta data (COREDELTA), and zonal quality data (ZONALEXT).

Single template can contain multiple FINGERPRINT sections. Their number is indicated by the FPCOUNT field. It is permitted to construct a template with no FINGERPRINT sections (zero FPCOUNT). Such template represents biometric system enrollee for whom fingerprint capture failed.

All fingerprint sections in the template must have unique combination of POSITION and VIEWOFFSET. If there are multiple FINGERPRINT sections with the same POSITION, then unique VIEWOFFSET is assigned to each and they must appear in the template ordered by VIEWOFFSET. The original ISO spec doesn't require it, but it makes sense to list FINGERPRINT sections with the same POSITION together in the template.

Finger position on hands (POSITION)

An unsigned 8-bit number encoding hand position of the finger. It must be in range 0-10 with meaning of allowed codes explained by the table below.

Code	Hand	Finger
0	Unknown	Unknown
1	Right	Thumb
2	Right	Index
3	Right	Middle
4	Right	Ring
5	Right	Little
6	Left	Thumb
7	Left	Index
8	Left	Middle
9	Left	Ring
10	Left	Little

Finger positions

Finger view number (VIEWOFFSET)

This unsigned 4-bit number is stored in the upper 4 bits of the byte it shares with SAMPLETYPE field. If multiple fingerprints with the same POSITION are present in the template, unique VIEWOFFSET must be assigned to each, starting with zero and incrementing with each view of the same finger. If there is only one fingerprint with given POSITION, its VIEWOFFSET is zero. VIEWOFFSET's allowed range of 0-15 limits every finger position to 16 finger views. Finger views of one finger must be listed in the template in order of increasing VIEWOFFSET.

Impression type (SAMPLETYPE)

This unsigned 4-bit number, occupying lower 4 bits of the byte it shares with VIEWOFFSET, indicates how the fingerprint was captured. Contrary to the vendor-defined DEVID, this field is actually useful, because its values are specified. Matching algorithms can use this information to adjust error tolerances and other parameters. Allowed values in range 0-3 and 8 are described in the table below.

Code	Liveness	Impression
0	Live	Plain
1	Live	Rolled
2	Non-live	Plain
3	Non-live	Rolled
8		Swipe

Impression types

There is no reasonable default. Implementations that lack knowledge of the sensor type should use the most likely value 0 for live plain scan.

These values are the same as in ANSI 378-2004 except that this format omits contactless sensor type.

Fingerprint quality (FPQUALITY)

An 8-bit unsigned number indicating overall fingerprint quality. Note that there is also a per-minutia quality in the MINQUALITY field. Allowed values are in range 0-100 with 0 meaning lowest quality and 100 highest quality. The original spec references section 2.1.42 of ISO/IEC 19784-1:2006 as a guideline for choosing values for this field, but there is no clear specification for what these quality values mean, which makes them useless in matching. There is no special value for unknown or unreported quality in the original spec. This format summary recommends using quality 100 as a reasonable default.

The only difference in this field between this format and ANSI 378-2004 is that this format references ISO 19784-1 while ANSI 378-2004 references ANSI 358. They both reference the same section, which suggests the two referenced specs are just copies of each other.

Number of minutiae (MINCOUNT)

An unsigned 8-bit number that indicates the number of MINUTIA records that follow. MINCOUNT may be zero, which means the feature extractor failed to find any minutiae.

Minutia (MINUTIA)

Minutiae are interesting points on the fingerprint, usually ridge endings and bifurcations.

MINUTIA records span 6 bytes each and carry minutia position (MINX, MINY), angle (MINANGLE), type (MINTYPE), and optionally quality (MINQUALITY). Field MINCOUNT contains the number of MINUTIA records present. There could be no MINUTIA records if the feature extractor failed to find any minutiae.

Minutia type (MINTYPE)

Minutia type is packed in the top two bits of the first byte occupied by the MINX field. Three minutia types are supported:

Bits	Type
01	Ridge ending
10	Ridge bifurcation
00	Other minutia type

Minutia types

The original spec says that the third minutia type "other" (code 00), should not be used for endings and bifurcation, but then it says that it should match not only itself but also endings and bifurcations, which means that it can be used as "either" minutia type, which is generated by feature extractors when they cannot clearly classify the minutia as ending or bifurcation. This interpetation was confirmed by ISO 19794-2:2011.

Interpretation of "other" minutia type differs from ANSI 378-2004, which has wording that implies minutiae of type "other" should only match each other.

Minutia X position (MINX)

Location of the minutia on the fingerprint in pixel units along X axis. Pixels are counted left-to-right, starting with zero. Minutia is in the center of the pixel specified by MINX and MINY.

MINX is a 14-bit unsigned number that is stored in the lower 14 bits of the big-endian 16-bit number that also contains MINTYPE.

In order to determine minutia position, feature extractor first constructs ridge *skeleton* by thinning ridges down to one pixel wide lines. Valley skeleton is constructed similarly by thinning valleys. Ridge endings lie at forking points of the valley skeleton. Similarly, ridge bifurcations lie at forking points of the ridge skeleton. Location of minutiae other than ending and bifurcation is implementation-specific, but it should be always present.

Implementation may calculate position differently as long as it approximates the above skeleton-based method. This is different from ANSI 378-2004, which doesn't provide such freedom.

Position of minutiae of type "other" is implementation-defined, but it should be done in such a way that these minutiae can match endings and bifurcations in case of ambiguous minutia type. This is a more precise definition than in ANSI 378-2004, which places no restriction on implementation-defined position of "other" minutiae.

Minutia Y position (MINY)

Like MINX above but for Y axis. Pixels on Y axis are counted top-down, starting with zero. MINY is stored in a 16-bit unsigned big-endian field, but only the lower 14 bits of this field are used for MINY. Upper 2 bits are zero.

Minutia angle (MINANGLE)

An 8-bit unsigned number holding quantized minutia angle in range 0-255. Zero angle points to the right and angle increases counterclockwise. Dequantization to floating-point degrees is performed by multiplying MINANGLE by (360/256). Quantization is performed by dividing the floating-point angle by (360/256), rounding to nearest integer, and taking the lowest 8 bits.

Ridge ending angle is defined as mean angle of two of the three valley skeleton legs that run around the ridge terminated by the ending. Skeleton leg angle is defined as the angle of its tangent, but this is highly ambiguous, because legs curve a lot around skeleton forking points. Ridge bifurcation angle is similarly derived from legs of ridge skeleton. Minutiae of type "other" must have an implementation-defined angle.

Both allowed value range and method of calculation differs from ANSI 378-2004. Next version of this format, ISO 19794-2:2011, offers recommended algorithm for calculating skeleton leg tangent.

Minutia quality (MINQUALITY)

An 8-bit unsigned number encoding minutia quality from 1 (lowest quality) to 100 (highest quality). If minutia quality is not reported, this field is zero. Meaning of minutia quality depends on feature extractor and it is thus useless in matching. Opensource implementations should set this field to zero.

ISO 19794-2:2011 changes range of values and defines recommended interpretation of minutia quality.

Total length of extension data (EXTBYTES)

A big-endian 16-bit unsigned number holding the total size of all extensions in bytes. If EXTBYTES is zero, there are no extension blocks. If it is non-zero, one or more EXTENSION blocks follow. In that case, EXTBYTES is the sum of EXTLEN fields of all EXTENSION blocks.

Extension data block (EXTENSION)

If EXTBYTES is non-zero, one or more EXTENSION blocks are attached to the fingerprint. Every extension has two compulsory fields: EXTTYPE and EXTLEN. EXTLEN lets implementations skip over unrecognized extensions without having to parse them. EXTTYPE informs the implementation about internal structure of the extension, so that appropriate parser can be chosen. Actual extension data is stored in the EXTDATA field.

There are three predefined extensions: ridge count extension (RCOUNTEXT), core and delta extension (COREDELTA), and zonal quality extension (ZONALEXT). Extensions are intended to store data that cannot be expressed otherwise. Implementations must not abuse extensions to introduce alternative encoding for data that can be already encoded in documented fields of this format, including the predefined extensions.

Extension type (EXTTYPE)

Two bytes encode extension type according to the table below.

Byte 1	Byte 2	Description
0	0	Reserved
0	1	Ridge count extension (RCOUNTEXT)
0	2	Core and delta extension (COREDELTA)
0	3	Zonal quality extension (ZONALEXT)
0	4-255	Reserved
1-255	0	Reserved
1-255	1-255	Implementation-defined extension

Extension types

Interpretation of implementation-defined extension type codes requires knowledge of vendor ID, which can be only provided externally (perhaps via CBEFF header), because this template format has no field for vendor ID. Consequently, when this template format is used alone (without wrapping or context), implementation-defined extension types are meaningless and thus useless in matching. Since opensource implementations usually don't have assigned vendor ID, they should not include any custom extension blocks.

Values are the same as in ANSI 378-2004 except that zonal quality extension was added.

Length of extension data (EXTLEN)

Length of EXTDATA field is encoded as a big-endian 16-bit unsigned number. Implementations can use this field to skip over unrecognized extensions.

This field is subtly different from similar field in ANSI 378-2004, which contains length of the entire EXTENSION block, including EXTTYPE field and EXTLEN field itself. This format records only length of EXTDATA here.

It is however not clear whether this is an intention or an error. Spec language is unclear. Example template in the spec includes only length of EXTDATA, but there are templates in the wild that use ANSI 378-2004 definition of EXTLEN. Technical Corrigendum 1 says nothing about EXTLEN or the example template, but the next version of this format, ISO 19794-2:2011, returns to ANSI 378-2004 definition of EXTLEN. To be safe, implementations should tolerate parsing errors in extension data caused by incorrect length information. Given the situation, it is advisable to avoid adding extensions to templates.

Extension data (EXTDATA)

Actual extension data. Length of this field is stored in EXTLEN. Internal structure of the EXTDATA field depends on the value of EXTTYPE field. This format specifies only structure of ridge count extension (RCOUNTEXT), core and delta extension (COREDELTA), and zonal quality extension (ZONALEXT).

Ridge count extension (RCOUNTEXT)

Minutia set defined in MINUTIA records can be annotated with ridge counts. Ridges are counted on a line (or *edge*) connecting two minutiae. Edge's ridge count is the number of ridges crossed by the edge as precisely defined in RIDGECOUNT field. RCOUNTEXT extension encodes this ridge count data. RCOUNTEXT has EXTTYPE of 0x0001.

Not all edges in the template need to have ridge count data. Edge picking method is specified in STARTYPE field. Information about every picked edge is then stored in EDGEDEF records that follow. The number of EDGEDEF records can be derived from the length of this extension block (EXTLEN) using expression `(EXTLEN - 1) / 3`. Ordering of EDGEDEF records depends on STARTYPE.

Edge picking method (STARTYPE)

This template format recognizes three edge picking methods: *quadrants*, *octants*, and *custom*. STARTYPE field contains an 8-bit unsigned code according to the table below.

Code	Method
0	Custom
1	Quadrants
2	Octants

Edge picking methods

Custom edge picking (code 0) simply means the implementation is free to choose edges however it wants. It can also choose how to order EDGEDEF records.

Quadrant (code 1) and octant (code 2) edge picking involves splitting the area around central minutia into four or eight sectors respectively. There are no restrictions on how the sectors are rotated relatively to the minutia. For every sector, ridge count is recorded for the edge to the nearest minutia in that sector. Quadrant and octant picking places strict requirements on presence, ordering, and contents of EDGEDEF records as explained below.

Edge between minutiae (EDGEDEF)

The 3-byte EDGEDEF record contains ridge count (stored in RIDGECOUNT field) for a single edge between minutiae identified by EDGEFROM and EDGETO fields. The number of EDGEDEF records is derived from EXTLEN as `(EXTLEN - 1) / 3`. Field STARTYPE determines which edges are present.

EDGEDEF record ordering depends on STARTYPE:

- If STARTYPE is custom (code 0), EDGEDEF records are ordered first by EDGEFROM and then by EDGETO.
- If STARTYPE is set to quadrants (code 1) or octants (code 2), EDGEDEF records for single central minutia are listed together. There are no other ordering constraints for neither central nor neighbor minutiae.

If STARTYPE is set to quadrants (code 1) or octants (code 2), the structure of EDGEDEF records is defined in more detail. EDGEFROM identifies the central minutia while EDGETO identifies the neighbor minutia. If some quadrant or octant does not have any minutiae, implementation must emit a placeholder record with EDGETO and RIDGECOUNT fields zeroed, thus padding edge list for every central minutia to four or eight records respectively.

Note that the original spec is unclear about the ordering of edges in ridge count extension. For maximum compatibility, template encoders should sort everything by EDGEFROM and then by EDGETO while decoders should accept arbitrary ordering. The original spec is also ambiguous as to whether quadrant and octant ridge picking (see STARTYPE) requires implementations to consider all minutiae as central minutiae or whether central minutiae can be picked arbitrarily. To be safe, encoders should include edges for all minutiae while decoders should accept edge information for a subset of all minutiae.

Starting minutia of the edge (EDGEFROM)

An 8-bit unsigned offset into the list of MINUTIA records. It identifies minutia that is on one end of the edge leading to EDGETO. If STARTYPE is quadrants (code 1) or octants (code 2), EDGEFROM is the central minutia.

Ending minutia of the edge (EDGETO)

An 8-bit unsigned offset into the list of MINUTIA records. It identifies minutia that is on the other end of the edge starting in EDGEFROM. If STARTYPE is quadrants (code 1) or octants (code 2), EDGETO is the neighbor minutia. If the current EDGEDEF record is a placeholder for quadrant or octant devoid of minutiae, EDGETO is zero.

Number of ridges the edge crosses (RIDGECOUNT)

An 8-bit unsigned number containing the number of ridges crossed by the edge. Ridge count does not include ridges connected to either EDGEFROM or EDGETO minutiae.

Curved ridges may be crossed more than once. The original spec doesn't say how to handle such cases, but it is reasonable to count every ridge only once.

If the current EDGEDEF record is a placeholder for quadrant or octant devoid of minutiae, RIDGECOUNT is zero. Note that zero is a valid value for very short edges. Zeroed RIDGECOUNT is ambiguous if EDGETO is also zero. Since very short edges are rare, implementations can assume that zero RIDGECOUNT means placeholder EDGEDEF record.

Core and delta extension (COREDELTA)

Besides normal minutiae defined in MINUTIA records, this format permits encoding of *cores* and *deltas*, which can be thought of as very special minutiae. COREDELTA extension encodes information about cores and deltas found on the fingerprint. COREDELTA extension has EXTTYPE of 0x0002. COREDELTA extension has two parts: COREDATA and DELTADATA.

Core data (COREDATA)

COREDATA, the first part of COREDELTA extension, contains a list of cores found on the fingerprint. Core is loosely defined as a point that is partially or completely encircled by ridges.

Every core is defined in a COREDEF record. Number of cores is specified by CORENUM field. Presence or absence of COREANGLE field is indicated by CHASANGLE flag.

COREDATA in ANSI 378-2004 had CHASANGLE field that was moved into individual COREDEF records in this format.

Number of cores (CORENUM)

Number of COREDEF records in the COREDELTA extension encoded as an unsigned 8-bit number. Valid values are in range 0-15.

ANSI 378-2004 required at least one core to be present. This format allows zero core count.

Core point (COREDEF)

COREDEF describes one core found on the fingerprint. COREDEF record repeats CORENUM times. Its structure is somewhat similar to MINUTIA record. Every core has position (COREX, COREY). If CHASANGLE is set, every COREDEF also contains COREANGLE.

Compared to ANSI 378-2004, COREDEF includes CHASANGLE field that was moved here from COREDATA header.

Core angle presence flag (CHASANGLE)

This 1-bit field occupies bit 14 (counting from zero) of the 16-bit big-endian unsigned number that also contains COREX. If it is set, COREANGLE field is present. If it is zero, COREANGLE is not recorded.

ANSI 378-2004 has the same field in COREDATA header. This format includes CHASANGLE field in every COREDEF record, allowing some cores to have an angle while others don't.

Core X position (COREX)

Equivalent of MINX for cores.

This field occupies lower 14 bits of the 16-bit big-endian unsigned number that also contains CHASANGLE field.

If there are ridge endings enclosed in the core, position of the innermost ridge ending defines position of the core. Innermost ridge ending is defined only as "nearest to maximal curvature" of the enclosing ridge, which is not clear at all. It is probably safe to pick ending closest to extreme (farthest) point of the enclosing ridge.

If there are no enclosed ridge endings, then the end of the enclosed valley defines position of the core.

ANSI 378-2004 didn't define exact core position.

Core Y position (COREY)

Equivalent of MINY for cores. See notes under COREX.

Core angle (COREANGLE)

This field is present only if the CHASANGLE flag is set. It contains an angle encoded in the same way as MINANGLE, but its meaning is different.

Core angle is derived from direction of ridges closest to the core position, but the original spec is not clear whether core direction should point out or inside the core. It says that it "points towards the open side of the concave ridge", but it's not clear what is "open side". We can only guess this means the core angle points out of the core.

Whorl cores have no angle. CHASANGLE must be zero in this case.

Range of values differs from ANSI 378-2004 just like in MINANGLE. ANSI 378-2004 also didn't specify how to determine core angle.

Delta data (DELTADATA)

DELTADATA, the second and last part of COREDELTA extension, contains a list of deltas found on the fingerprint. Delta is loosely defined as the point where three distinct ridge flows meet.

Every delta is defined in a DELTADEF record. Number of deltas is specified by DELTANUM field. Presence or absence of DELTAANGLE fields is indicated by DHASANGLE flag.

DELTADATA in ANSI 378-2004 had DHASANGLE field that was moved into individual DELTADEF records in this format.

Number of deltas (DELTANUM)

Number of DELTADEF records in the COREDELTA extension encoded as an unsigned 8-bit number. Valid values are in range 0-15.

Delta point (DELTADEF)

DELTADEF describes one delta found on the fingerprint. DELTADEF record repeats DELTANUM times. Its structure is somewhat similar to MINUTIA record. Every delta has position (DELTAX, DELTAY). If DHASANGLE is set, every DELTADEF also contains three repeats of the DELTAANGLE field.

Compared to ANSI 378-2004, DELTADEF includes DHASANGLE field that was moved here from DELTADATA header.

Delta angle presence flag (DHASANGLE)

This 1-bit field occupies bit 14 (counting from zero) of the 16-bit big-endian unsigned number that also contains DELTAX. If it is set, three repeats of the DELTAANGLE field are present. If it is zero, no DELTAANGLE is recorded.

ANSI 378-2004 has the same field in DELTADATA header. This format includes DHASANGLE field in every DELTADEF record, allowing some deltas to have angles while others don't.

Delta X position (DELTAX)

Equivalent of MINX for deltas.

This field occupies lower 14 bits of the 16-bit big-endian unsigned number that also contains DHASANGLE field.

Delta position is defined as the center of weight of three points, each of which lies where two ridges approaching the delta diverge. Such definition doesn't cover all types of deltas though.

ANSI 378-2004 didn't define exact delta position.

Delta Y position (DELTAY)

Equivalent of MINY for deltas. See notes under DELTAX.

Delta angle (DELTAANGLE)

This field is present three times if DHASANGLE flag is set and zero times otherwise. The field contains an angle encoded in the same way as MINANGLE, but its meaning is different.

Delta angle is defined by direction of ridges before they diverge as they approach the delta. Delta angle points away from the delta.

If any of the three angles cannot be determined, the unused angle field should be filled with a copy of another angle. When the template is decoded, duplicate delta angles should be discarded.

Range of values differs from ANSI 378-2004 just like in MINANGLE. ANSI 378-2004 also didn't specify how to determine delta angle.

Zonal quality extension (ZONALEXT)

For the purposes of this extension, original fingerprint image is split into rectangular zones. Zone size is ZONEWIDTH times ZONEHEIGHT pixels. Rightmost and bottom zones may be smaller to fit image width and height exactly. Quality score is assigned to every zone in the image. A two-dimensional array of these scores is stored in this extension.

Meaning of zone quality scores is implementation-defined, which renders zonal quality data nearly useless. Implementation also chooses number of bits allocated for each zone and records this information in ZONEBITS field.

Contrary to the other two extensions, zonal quality was not part of ANSI 378-2004.

Zone width (ZONEWIDTH)

Horizontal size of one quality zone in pixels encoded as a non-zero unsigned 8-bit number. If ZONEWIDTH is not a factor of WIDTH, then the rightmost zones will be narrower.

Zone height (ZONEHEIGHT)

Vertical size of one quality zone in pixels encoded as a non-zero unsigned 8-bit number. If ZONEHEIGHT is not a factor of HEIGHT, then the bottom zones will be narrower.

Bits per zone (ZONEBITS)

Number of bits allocated for each zone in ZONALQUALITY data. This field is encoded as an unsigned 8-bit number. Any non-zero value is allowed, but more than 8 bits per zone are unlikely to be supported.

Next version of this format, ISO 19794-2:2011, clarifies that only 1-8 bits should be used per zone.

Zonal quality data (ZONALQUALITY)

This field encodes a two-dimensional array of unsigned zonal quality values. Higher values mean higher image quality in the zone.

Number of zones can be calculated by dividing WIDTH and HEIGHT by ZONEWIDTH and ZONEHEIGHT respectively and rounding up to get array width and height.

Zones are encoded row by row top-down and left-to-right. Each zone takes up ZONEBITS bits. Zones are aligned to bits rather than bytes, i.e. they are packed without any padding bits, spanning bytes if necessary. Zones in one byte are laid out starting from the most significant bits and ending with the least significant bits. Only the last byte of ZONALQUALITY has unused least significant bits filled with zeroes.

Robert Važan robert.vazan@tutanota.com