UIDAI

Unique Identification Authority of India
Planning Commission, Govt. of India (GoI),
3rd Floor, Tower II,
Jeevan Bharati Building,
Connaught Circus,
New Delhi 110001

AADHAAR BEST FINGER DETECTION

API SPECIFICATION - VERSION 1.6

FEBRUARY 2012

Table of Contents

1. Introduction

The Unique Identification Authority of India (UIDAI) has been created, with the mandate of providing a Unique Identity (Aadhaar) to all Indian residents. The UIDAI proposes to provide online authentication using demographic and biometric data.

Aadhaar "*authentication*" means the process wherein Aadhaar Number, along with other attributes, including biometrics, are submitted to the Central Identities Data Repository (CIDR) for its verification on the basis of information or data or documents available with it. UIDAI will provide an online service to support this process. Aadhaar authentication service only responds with a "yes/no" and no personal identity information is returned as part of the response.

1.1 Best Finger

When authenticating a resident using any single finger, the accuracy or the chances of being matched would be different due to differences in quality across all his/her fingers. This variation may also be present due to the manner in which the resident normally interacts with a typical fingerprint scanner and the different fingers may inherently have different amount of identifying information depending on the size of the finger and the commonness of the pattern it carries. It may thus be useful to appraise each resident of finger providing the best accuracy and successful matching results. We shall refer to this finger with best accuracy as the **best finger**. Resident may possess one or more best fingers. This knowledge allows the resident to provide his/her best finger(s) during authentication thereby increasing the chances of successful match.

1.2 Target Audience and Pre-Requisites

This is a technical document and is targeted at software professionals working in technology domain and interested in incorporating Aadhaar authentication into their applications. Readers must be fully familiar with Aadhaar Authentication API 1.5 published on UIDAI website (http://uidai.gov.in/) before reading this document.

1.3 Objective of this document

This document provides Aadhaar Best Finger Detection API (Application Programming Interface) specification. It contains details including API data format, protocol, and security specifications.

2. Best Finger Detection API

This chapter describes the principles of Best Finger Detection (BFD), its usage, and API details including the flow, communication protocol, and data formats.

2.1 Best Finger Detection (BFD)

Since many residents in India are engaged in manual labour, the quality of fingerprints vary considerably even between fingers of the same resident. So it is important to identify the best finger(s) to improve authentication accuracy and hence be more inclusive in supporting Aadhaar authentication across all sections of society.

The Best Finger for a resident is the one that, when selected for authentication, provides the highest chance of successful authentication for that resident. The best finger to be used for authentication depends on the intrinsic qualities of the finger (ex. ridge formation, how worn out they are, cracked, etc.), as well as the quality of images captured during enrolment process and the authentication transaction.

2.2 Best Finger Detection Process

AUAs who use fingerprint based Aadhaar authentication within their applications should implement BFD application as part of their Aadhaar biometric authentication enabled applications. BFD application should use this API and is used to categorise resident's authentication readiness. In addition, this helps the resident understand how Aadhaar biometric authentication work and which of their fingers are best usable for such purposes. A sample Java application is made available by UIDAI on the developer portal (https://developer.uidai.gov.in/).

There are two scenarios under which BFD application needs to be used:

- Authentication API returns error code asking resident BFD to be done (error code "812", see Aadhaar Authentication API Specification 1.6). Whenever this error comes back, either automatically or by operator, BFD application must be launched and have the resident do the BFD to identify his/her best fingers.
- If resident specifically wants to get his best finger identified and get a BFD receipt, proactively BFD application could be launched and used. This may happen due to authentication errors even after initial BFD typically resulting from wear and tear of fingers, climatic conditions, etc. or due to the fact that resident may have re-enrolled updating his/her biometrics in Aadhaar system.

BFD application should do the following:

1. Capture one finger at a time.
    - One at a time, capture all fingers.
    - During the capture of fingerprints by the BFD application, all captured fingerprint images should be subjected to image check to measure NFIQ.
2. Ensure resident and operator clearly knows which finger is being scanned.
    - When scanning different finger, local match should be done to ensure resident is not placing finger that is already scanned again by mistake.
    - Also, when scanning same finger (for capturing best attempt), ensure they are matched locally to ensure same finger is being placed.
3. Up to three attempts are carried out so that good quality images are captured for each finger. Templates extracted from best quality image of three attempts should be sent to Aadhaar server for BFD purposes.
4. Once the best attempt is captured for all fingers, application forms the input XML for the BFD API as specified in this document.
5. Application invokes the BFD API through AUA server (similar to authentication).
6. Based on the response, provide a printed receipt to the resident indicating the ranking for each finger.

AUA devices using biometric authentication implementing this BFD API should have the user interface for capture and sending as described below:

1. Resident/operator needs to clearly know which finger to capture and should be visible on screen.
2. There must be options to rescan after the capture is complete.
3. Capture high quality fingers up to 3 attempts and application must pick up highest NFIQ image (if possible NFIQ 1 & 2).
4. Application should remember the finger position because it has to be sent along side NFIQ for every finger as part of BFD API input.
5. Application must do local matching to avoid same finger being sent against different positions and to also ensure same finger is indeed used during multiple attempts.
6. Application should send only one best template per finger (maximum 10 templates in total) and not images.
7. Provide for exception where resident may not have all ten fingers.
8. Provide an option to buffer the transaction in case the connectivity is not there and replay from database.
9. Application must not keep any unencrypted data that involves resident information including biometrics.
10. BFD application must provide a printed receipt UI indicating rank details from API response, preferably with a picture of the hand

Logging information related to each of the attempts will help UIDAI to characterise and analyze such data. BFD server at UIDAI end processes incoming requests as described below:

1. Fingers are ranked in descending order based on the matching score.
2. BFD application feedback helps resident to clearly identify which of his/her fingers are good for authentication. Resident is expected to use his/her fingers in the order of rank starting from 1 while doing authentication.

NOTE: Above is a high level logic and UIDAI may decide to enhance this logic at the backend from time to time. Change of such logic will not have any impact on the actual API input/output signatures. It is provided for general understanding. Sample BFD application is available on developer portal (https://developer.uidai.gov.in).

2.3 API Protocol

Aadhaar Best Finger Detection (henceforth referred as BFD) service is exposed as stateless service over HTTPS. Usage of open data format in XML and widely used protocol such as HTTP allows easy adoption and deployment of these services. To support strong end to end security and avoid request tampering and man-in-the-middle attacks, it is essential that encryption of data happens at the time of capture on the capture device.

Following is the URL format for Aadhaar BFD service:

```
https://<host>/bfd/<ver>/<ac>/<uid[0]>/<uid[1]>/<asalk>
```

API input data should be sent to this URL as XML document using Content-Type "application/xml" or "text/xml".

For security reason data collected for Aadhaar BFD service must not be stored in the devices or log files. It's essential for ASA and AUA to maintain audit records for all the request metadata along with the response.

As a best practice, for all SSL communications the agencies should automatically validate the SSL certificate and ensure it is validated against the revocation list online.

2.3.1 Element Details

host – Aadhaar BFD server address. Actual production server address will be provided to ASAs. Note that production servers can only be accessed through secure leased lines. For development and testing purposes, public URL "*auth.uidai.gov.in*" can be used. ASA server should ensure that actual URL is configurable.

Next part of the URL "bfd" indicates that this is a Best Finger Detection API call instead of regular authentication API call. Ensure that this is provided.

ver – BFD API version (optional). If not provided, URL points to current version. UIDAI may host multiple versions for supporting gradual migration. As of this specification, default production version is "1.6".

ac – A unique code for the AUA which is assigned by UIDAI. This is an alpha-numeric string having maximum length 10. (A default value "public" is available for testing.)

uid[0] and **uid[1]** – First 2 digits of Aadhaar Number. Used for load-balancing.

asalk – A valid ASA license key. ASAs must send one of their valid license keys at the end of the URL. It is important that license keys are maintained safely. When adding license key to the URL, ensure it is "URL encoded" to handle special characters.

For all valid responses, HTTP response code 200 is used. All application error codes are encapsulated in response XML element. In the case of connection and other server errors, standard HTTP error response codes are used (4xx codes such as 403, 404, etc.). HTTP automatic redirects also should be handled by ASA server.

ASA server must send one of their valid license keys as part of the URL (see details above). Authentication related APIs are enabled only for valid ASAs and only for their registered static IP addresses coming through a secure private network.

2.4 Best Finger Detection API: Input Data Format

Aadhaar BFD service uses XML as the data format for input and output. To avoid sending unnecessary data, do not pass any optional attribute or element unless its value is different from default value. Any bad data or extra data will be rejected.

Following is the XML data format for BFD API:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Bfd uid="" tid="" ac="" sa="" ver="" txn="" lk="">
  <Meta udc="" fdc="" pip="" lot="G|P" lov=""/>
  <Skey ci="" ki="">encrypted and encoded session key</Skey>
  <Data type="X|P">encrypted RBD block</Data>
  <Hmac>SHA-256 Hash of RBD block, encrypted and then encoded </Hmac>
  <Signature>Digital signature of AUA</Signature>
</Bfd>
```

"Data" element contains "Rbd" (Resident Biometric Data) element which is a base-64 encoded encrypted block. Complete "Data" block should be encrypted at the time of capture on the capture device. But, encoding (base-64) of "Data" block and packaging it with enveloping XML under "Bfd" element can either be done on the device or on the AUA server based on the AUA needs. Device capability, protocol between devices and AUA server, and data format used between devices and AUA server, etc. should be considered for making that choice.

When using RBD block in XML format (which is the default), following is the format for "Rbd" element:

```
<Rbd ts=”” ver=””>
  <Bios>
    <Bio nfiq=”” na=”” pos=””>encoded biometric</Bio>
  </Bios>
</Rbd>
```

Instead of XML format, this version also allows RBD block to be in binary format based on Protocol Buffers standard ((http://code.google.com/p/protobuf/). Notice that “Bfd” XML must be in XML format. Binary format is only supported for RBD block to enable smaller packet sizes to be transmitted from devices. See Appendix for details.

2.4.1 Element Details

Element: **Bfd** (mandatory)

- Root element of the input XML for BFD service.

Attributes:

- **uid** – (mandatory) Aadhaar Number of the resident
- **tid** – (mandatory) For Registered devices, send its unique Terminal ID. For Public devices, value should be passed as “public”.
- **ac** – (mandatory) A unique code for the AUA which is assigned by UIDAI during AUA registration process. This is an alpha-numeric string having maximum length 10. (A Default value “public” is available only for testing.)
- **sa** – (mandatory) A unique “Sub-AUA” code. AUAs are expected to manage these codes within their system and ensure uniqueness within their system. This allows auditing and business intelligence to be provided at SA level. If AUA and SA are same agency, use value of “ac” for this attribute. This is an alpha-numeric string having maximum length 10.
- **ver** – (mandatory) version of the API. Currently only valid value is “1.5”.
- **txn** – (mandatory) AUA specific transaction identifier. AUA can choose to pass this as part of input. This is returned as part of response as is. This is very useful for linking transactions full round trip across systems. This is an alpha-numeric string of maximum length 50. Only supported characters are A-Z, a-z, 0-9, period, comma, hyphen, backward & forward slash, left & right parenthesis, and colon. No other characters are supported. It is highly recommended that AUAs use this attribute for correlating requests with responses for auditing and verification.
- **lk** – (mandatory) A valid “License Key” assigned to the AUA. Administration portal of UIDAI will provide a mechanism for AUA administrator to generate license. This is an alpha-numeric string of length up to 64 characters.

Note: You can use any valid license key that has fingerprint authentication feature enabled for this purpose.

Element: **Meta** (Mandatory)

- This element specifies metadata related to the device and transaction. This is mandatory for better tracking, reporting, and trouble shooting.

Attributes:

- **udc** – (mandatory) Unique Device Code. This is a unique code for the authentication device assigned within the AUA domain. This is an alpha-numeric string of maximum length 20.
  - This allows better reporting and tracking of devices as well as help resolve issues at the device level.
  - It is highly recommended that AUAs define a unique codification scheme for all their devices.
  - Suggested format is *"[vendorcode][date of deployment][serial number]"*
- **fdc** – (mandatory) Fingerprint device code. This is a unique code provided for the fingerprint sensor-extractor combination. AUAs will have access to this code through UIDAI portal. This is an alpha-numeric string of maximum length 10.
  - While using fingerprint authentication, this code is mandatory and should be provided. If the code is unknown or device is not certified yet, use "NC".
- **pip** – (mandatory) Public IP address of the device. If the device is connected to Internet and has a public IP, then this must be populated with that IP address. If the device has a private IP and is behind a router/proxy/etc, then public IP address of the router/proxy/etc should be set. If no public IP is available, leave it as "NA".
- **lot** – (mandatory) Location type. Valid values are "G" and "P".
  - G – stands for geo coding in lat, long format
  - P – stands for postal pin code
- **lov** – (mandatory) Location Value.
  - If "lot" value is "G" then, this "lov" attribute must carry actual "lat,long,alt" of the location. Can be derived using GPS or using alternate method. Lat/Long should be in positive or negative decimal format (ISO 6709). Altitude is optional and may be populated if available.
  - If "lot" value is "P" then, this "lov" attribute must have a valid 6-digit postal pin code.

Element: **Skey** (mandatory for Public devices)

- Value of this element is base-64 encoded value of encrypted 256-bit AES session key. **Session key must be dynamically generated for every transaction (session key must not be reused) and must not be stored anywhere except in memory**. See next chapter for encryption details.

Attributes:

- **ci** – (mandatory) Public key certificate identifier using which "skey" was encrypted. UIDAI may have multiple public keys in field at the same time. Value of this attribute is the certificate expiration date in the format "YYYYMMDD". Expiry date of the certificate can be obtained from the certificate itself.
- **ki** – (optional) **This is for advanced use only**. Do not use this attribute unless you clearly understand what you are doing. See chapter on "API and Data Security" for details.

Element: **Data** (mandatory)

- Contains the encrypted “Rbd” element in base-64 encoding. See “Rbd” element definition later.

Attributes:

- **type** – (optional) Type of the RBD block format. It can have two values – “X” for XML and “P” for Protobuf binary format. Default value is assumed to be “X”.

Element: **Hmac** (mandatory)

- Devices which is constructing the “Rbd” element must perform the following to provide the Hmac value:
    - If Rbd type is “X” (XML), then:
        - After forming Rbd XML, compute SHA-256 hash of Rbd XML string
        - Then encrypt using session key (skey)
        - Then encode using base-64 encoding (as described earlier, encoding can be done on the AUA server when forming final BFD request XML)
    - If Rbd type is “P” (Protobuf), then:
        - After forming Protobuf byte array for Rbd, compute SHA-256 hash of Rbd protobuf bytes.
        - Then encrypt using session key (skey)
        - Then encode using base-64 encoding (as described earlier, encoding can be done on the AUA server when forming final BFD request XML)

Element: **Signature** (mandatory)

- The request XML should be digitally signed for message integrity and non-repudiation purposes.
- Digital signing should always be performed by the entity that creates the final request XML
    - AUA can digitally sign after forming the API input XML. This is almost always the case. In such cases, AUA ensures the message security and integrity between AUA servers and its client applications.
    - ASA can digitally sign the request XML if it is a domain-specific aggregator and forms the request XML on behalf of the AUA. In such cases, ASA and AUA ensure the message security and integrity between their servers.
- Procuring digital signature certificates:
    - It should be procured from a valid certification authority as per Indian IT Act (see http://cca.gov.in/rw/pages/faqs.en.do#thecaslicensedbythecca)
    - Digital certificates have two parts:
        - X.509 certificate representing public key.
        - Private Key which is used for digital signing. Private Key should be stored securely and is the responsibility of the owner of the certificate to ensure that it is not compromised.
    - It should be a class II or class III certificate.

    - X.509 certificate contains information about the owner of the certificate; in this case it will be details of the person and the organization to which he/she belongs. UIDAI server checks to ensure that certificate belongs to the ASA or AUA organization. Hence, it is mandatory that "O" attribute of "Subject" in the X.509 certificate matches the name of the organization.
- Digital signing of request XML
    - XML digital signature algorithm as recommended by W3C.
    - Signature should include key info element that contains X.509 certificate details. This is needed for UIDAI server to validate the signer.
- Verification of digital signature by UIDAI servers. UIDAI server validates the signature in the following sequence:
    - Checks if the signature element is present. If not, it throws an error.
    - If signature element is present, then it validates if the certificate is issued by one of the valid certification authority. If not valid, throws error.
    - If it is a valid certificate, then it validates whether the "O" attribute in the X.509 certificate's subject matches the AUA organization name. If yes, proceeds with API logic.
    - If it does not match AUA organization name, it checks configuration to see if ASA is allowed to sign on behalf of that AUA. If not, throws error.
    - If ASA is allowed to sign on behalf of that AUA, it checks whether the "O" element of the certificate matches with the organization name of the ASA. If not, throws error.
    - If it matches, it proceeds with API logic.
    - In future, UIDAI may choose to conduct additional validations against white listed certificates within UIDAI database.

Element: **Rbd** (mandatory) – Resident Biometric Data element.

Attributes:

- **ts** – (mandatory) Timestamp at the time of capture of BFD input. This is in format "YYYY-MM-DDThh:mm:ss" (derived from ISO 8601). Time zone should not be specified and is automatically defaulted to IST (UTC +5.30).

  ver – (mandatory) version of the "Rbd" element. Currently only valid value is "1.0". Notice that this is NOT same as BFD API version. If there is a change in the structure of "Rbd" XML, all the devices may not upgrade to latest software version.

AUAs can buffer BFD requests and send it to Aadhaar authentication server to support occasional lack of network connectivity on the field. Maximum time up to which requests can be queued (buffered) will be defined by UIDAI policy. During initial release, this will be configured to 24 hours. All requests with "ts" value older than this limit will be rejected. It is recommended not to use synchronized keys for buffered transactions and if synchronized session key scheme is used, entire set of transactions within a session should be buffered together.

Element: **Bios –** (mandatory)

- This element can have one or many "Bio" elements carrying biometric records to be matched.

Element: **Bio** (at least 1 element mandatory)

- Base-64 encoded biometric record. **This is always of type FMR** ("Fingerprint Minutiae Record").
- Number of "Bio" elements under "Bios" must be between 1 and 10.

Attributes:

- **nfiq** – (mandatory) NFIQ score of the finger image calculated during capture. For BFD service to work well, it is necessary that BFD application attempts to capture best NFIQ score for each finger.
- **na** – (mandatory) Number of attempts before this capture was taken. It is suggested that BFD applications provide at least 3 attempts to residents to obtain best NFIQ.
- **pos** – (mandatory) In order to reduce the unnecessary matching of biometric templates, it is mandatory that BFD request contain the biometric position along with biometric templates. This attribute can have following values:

LEFT_INDEX
LEFT_LITTLE
LEFT_MIDDLE
LEFT_RING
LEFT_THUMB
RIGHT_INDEX
RIGHT_LITTLE
RIGHT_MIDDLE
RIGHT_RING
RIGHT_THUMB

For BFD service to work well, all available fingers of the resident should be captured and sent. All fingers must be captured and used while calculating the match scores for each finger and detecting best fingers. It is also necessary that BFD application attempts to capture best NFIQ score for each finger.

2.5 Best Finger Detection API: Response Data Format

BFD API does not provide any identity data as part of the response. All it does is to match given input and respond with a matchability rank for each finger.

Response XML is given below.

```
<BfdRes code=”” txn=”” err=”” ts=”” actn=”” msg=””>
 <Ranks>
   <Rank pos=”” val=””/>
 </Ranks>
 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
   <CanonicalizationMethod
     Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
   <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha256"/>
   <Reference URI="">
     <Transforms>
      <Transform
       Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
     </Transforms>
     <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha256"/>
       <DigestValue></DigestValue>
   </Reference>
  </SignedInfo>
  <SignatureValue></SignatureValue>
 </Signature>
</BfdRes>
```

2.5.1 Element Details

Element: **BfdRes**

Attributes:

- **code** – unique alphanumeric “BFD response” code having maximum length 40.
- **txn** – AUA specific transaction identifier that was part of input.
- **ts** – Timestamp when the response is generated. This is of type XSD dateTime.
- **err** – Failure error code. If BFD request fails, this attribute provides any of the following codes (for updates, see https://developer.uidai.gov.in/site/api_err):
    - **“300”** – Biometric data did not match
    - “314” – Number of fingers should not exceed 10
    - **“500”** – Invalid encryption of Skey
    - **“501”** – Invalid certificate identifier in “ci” attribute of “Skey”
    - **“502”** – Invalid encryption of Rbd
    - **“503”** – Invalid encryption of Hmac
    - **“504”** – Session key re-initiation required due to expiry or key out of sync
    - **“510”** – Invalid “Bfd” XML format
    - **“511”** – Invalid “Rbd” XML format
    - **“520”** – Invalid device
    - **“530”** – Invalid AUA code
    - **“540”** – Invalid BFD XML version
    - **“541”** – Invalid RBD XML version
    - **“542”** – AUA not authorized for ASA. This error will be returned if AUA and ASA do not have linking in the portal
    - **“543”** – Sub-AUA not associated with “AUA”. This error will be returned if Sub-AUA specified in “sa” attribute is not added as “Sub-AUA” in portal

- **"561"** – Request expired ("Rbd->ts" value is older than *N* hours where *N* is a configured threshold in BFD server)
- **"562"** – Timestamp value is future time (value specified "Rbd->ts" is ahead of BFD server time beyond acceptable threshold)
- **"563" –** Duplicate request (this error occurs when exactly same BFD request was re-sent by AUA)
- **"564" –** HMAC Validation failed
- **"565" –** AUA license key has expired or is invalid
- **"566" –** ASA license key has expired or is invalid
- **"569"** – Digital signature verification failed
- **"570"** – Invalid key info in digital signature (this means that certificate used for signing the BFD request is not valid – it is either expired, or does not belong to the AUA or is not created by a CA)
- "**572**" – Invalid biometric position
- **"583"** – Best finger detection not allowed as per license
- **"800**" – Invalid biometric data
- **"811"** – Missing biometric data in CIDR for the given Aadhaar number
- **"940"** – Unauthorized ASA channel
- **"941"** – Unspecified ASA channel
- **"999"** – Unknown error

actn – Actionable feedback in case resident or operator needs to take specific actions. Valid action codes are:

- "00" - No action necessary
- "01" - Please check if your AADHAAR number was entered correctly. If it was correct, you may have to update your enrollment.
- "02" – Reserved for future use
- "03" - Please try to use BFD again. Your biometrics have not been captured properly.
- 04 – Please check whether your Aadhaar number was entered correctly. If it was correct, you may have to update your enrolment.
- "99" - Please review BFD result and determine suitable action.

- **msg** – Actionable feedback message in English in case resident or operator needs to take specific actions. BFD applications should display message to enable operator and resident take appropriate actions. The messages for various actions are defined in the previous bullet.

Element: **Ranks**

- Biometric matching ranks for each finger that was part of input for BFD service.

Element: **Rank**

- Rank element for each finger. Number of "Rank" elements will match the number of "Bio" elements in the input XML.
- This is provided in the case there are no errors.

Attributes:

- **pos** – Finger position for which matching rank is provided. Possible values are same as that of "pos" attribute within "Bio" element of input XML.
- **val** – This attribute indicates a value between 1 and 10. This attribute indicates the order of preference in which resident should use his/her fingers for authentication. Value 1 indicates first choice and 10 indicates last choice.

NOTE: If some of the fingers are not matchable at all (that means those fingers cannot be used for authentication), then records corresponding to those fingers will not be part of the "Ranks" XML. So, it is important to note that number of "Rank" elements may be less than what was provided in the input. Application should interpret it using "pos" attribute corresponding to each "Rank" element.

Element: **Signature**

This is the root element of UIDAI's digital signature. This signature can be verified using UIDAI public key. Signature complies with W3C XML signature scheme.

For more details, refer: http://www.w3.org/TR/xmldsig-core/

3. API and Data Security

For Public devices, data should be encrypted with a dynamic session key using **AES-256** symmetric algorithm (AES/ECB/PKCS7Padding). Session key, in turn, is encrypted with **2048-bit UIDAI public key** using asymmetric algorithm (RSA/ECB/PKCS1Padding). Reference implementation demonstrates this in detail. <u>Session key must not be stored anywhere except in memory and should not be reused across transactions unless synchronized session key scheme is used</u>.

As an advanced feature, this revision of the API version also supports a scheme called Synchronized Session Key.

BFD API security is designed to be same as that of Aadhaar Authentication API security. See Aadhaar Authentication 1.6 API specification for details.

4. Appendix

4.1 Related Publications

The following standards are applicable and related to the information in this document.

Aadhaar Authentication API 1.5 (Rev 2)	http://uidai.gov.in/images/FrontPageUpdates/aadhaar_authentication_api_1_5_rev2.pdf
Biometric Standards	http://uidai.gov.in/UID_PDF/Committees/Biometrics_Standards_Committee_report.pdf
Aadhaar biometric APIs	http://uidai.gov.in/UID_PDF/Working_Papers/Aadhaar_ABIS_API.pdf
Data Encryption Algorithm	ANXI X3.92
Banking—Retail Financial Services Symmetric Key Management	ANSI X9.24
Public Key Cryptography for the Financial Service Industry: Agreement of Symmetric Keys Using Discrete Cryptography	ANSI X9.42
Triple Data Encryption Algorithm: Modes of Operation	ANSI X9.52
Security Requirements for Cryptographic Modules	FIPS PUB 140-2
Information Technology – Security Techniques – Hash Functions	ISO 10118
Information Technology – Security Techniques – Key Management	ISO 11770
Information Technology – Security Techniques – Encryption Algorithms	ISO 18033
Biometric standards	ISO 19794-4, ISO 19794-6
Date and Time format standard	ISO_8601
XML Signature	http://www.w3.org/TR/xmldsig-core/
Protocol Buffers	http://code.google.com/p/protobuf/
Geo Location Standard	ISO 6709

4.2 Binary format for RBD block

Protocol Buffers are a way of encoding structured data in an efficient yet extensible format. Google uses Protocol Buffers for almost all of its internal RPC protocols and file formats. It provides a flexible, efficient, automated mechanism for serializing structured data. See https://developers.google.com/protocol-buffers/docs/overview for details.

This version of the API supports RBD block to be sent to AUA server in protobuf format as an alternate to default XML format. This allows compact binary representation of the device data and avoids extra encoding required for XML format. If this scheme is used, device applications are expected to form the RBD block in protobuf format using the ".proto" file for RBD block. Final ".proto" files for this version are available at https://developer.uidai.gov.in/site/downloads

4.3 Changes in Version 1.6 from Version 1.5

Old (1.5)	New (1.6)
RBD block is always XML	Added support for binary Protobuf format
"Txn" attribute was optional	"Txn" attribute made mandatory to ensure transactions are correlated and managed across systems well
Session Key is always sent	Allows advanced use case where session key is synchronized
Meta element was inside RBD block	Moved "Meta" outside RBD block to allow AUAs to validate them, log them, and use it in report
Meta element was optional	"Meta" element and its attributes made mandatory
--	"Meta" element structure changed to accommodate geo location, public IP address of the device, and a unique device code
--	Additional error codes "314, 504" added
Response indicated a color as part of rank	Response indicates only rank. Concept of color is removed.