Identity Based Open Exchange Protocol Specification (IBOPS)

Architecture and Security

Last Updated: Version 1.00, April 10, 2015

Specification URIs
This version:
Previous version:
Latest version:

Technical Committee:
Identity Based Attestation and Open Exchange Protocol Specification TC

Chairs:

Editors:

Related work:

Abstract:
This document is a specification that ...TBD
Table of Contents

Identity Background
   A Short History Of Identity
   Biometrics Overview
IBOPS Overview
   Introduction to BOPS
   Architecture Overview
IBOPS Identity Management
   Identity Genesis and User Enrollment
      Overview
      The Genesis/User Enrollment Process
      Secondary Devices and Backup
   Authentication/Identity Assertion
      Overview
      Client Software Authentication
      Post-genesis device/user authentication
      Server Authentication
   Authenticated Session Management
   Access Control
Intrusion Detection
   Overview
   Replay Prevention
Standards, Audit, and Quality Control
   Standards
   Audit Capabilities
   Implementation Quality
Data Protection
   Network Transmissions
   Data Protection Strategy
   Client-side data
   Server-side data
      IBOPS Server Data
      Host Data
IBOPS Server Resource Guide
1. Identity background and overview

Note: This clause expands on some core identity definitions and basic model from ITU-T X.1252

Consider the following working definitions of identity in this work

- **Identity**: A representation of an entity in the form of one or more attributes that allow the entity or entities to be sufficiently distinguished within context. For identity management (IdM) purposes, the term identity is understood as contextual identity (subset of attributes), i.e., the variety of attributes is limited by a framework with defined boundary conditions (the context) in which the entity exists and interacts.
  
  NOTE – Each entity is represented by one holistic identity that comprises all possible information elements characterizing such entity (the attributes). However, this holistic identity is a theoretical issue and eludes any description and practical usage because the number of all possible attributes is indefinite.

- **Identity assurance**: The degree of confidence in the process of identity validation and verification used to establish the identity of the entity to which the credential was issued, and the degree of confidence that the entity that uses the credential is that entity or the entity to which the credential was issued or assigned.

  NOTE – The confidence is based on the degree of confidence in the binding between the entity and the identity that is presented.

- **Entity authentication assurance (EAA)**: A degree of confidence reached in the authentication process that the entity is what it is, or is expected to be (this definition is based on the 'authentication assurance' definition given in [b-ITU-T X.1252]).

  NOTE – The confidence is based on the degree of confidence in the binding between the entity and the identity that is presented.

- **Identifier**: One or more attributes that uniquely characterize an entity in a specific context.

- **Identity information verification**: A process of checking identity information and credentials against issuers, data sources or other internal or external resources with respect to authenticity, validity, correctness and binding to the entity.

In the real world, the identity of a natural person is readily accepted and is based upon an extensive set of characteristics or attributes. Some of these are physical features such as height, hair colour, general appearance, mannerisms, behaviour, etc. Others, like date of birth, place of birth, home address, telephone number may also be used.

In on-line interactions, participating parties normally have the requirement to have enough confidence that they communicate with the correct partner. This process of seeking this confidence would often involve two or more individuals or "entities": the entity whose identity is to be confirmed – the requesting entity, and the entity that will rely on a confirmed identity – the
relying party. A third entity which manages identities may be involved – an identity service provider.

In the digital or "on-line" world, an "identity" is also made up of attributes, just like the real world. However, in this case, the "identity" may be limited to a single feature or it may have many; it will depend on the context in which it appears. This applies to inanimate objects as well as natural persons so users are often referred to as an entity.

Generally, identifiers, and/or attributes will uniquely characterize an entity within a particular context. Because of this, an entity may have a number of different identities some of which will be a subset of other identities.

1.1 Authentication and confidence

The authentication process may be described as follows:

- Most communication processes require that the communication partners have adequate confidence or trust that they are really communicating with the intended partner. Therefore, at the beginning of a communication, the partners try to reach an adequate level of confidence on the basis of available identity information about the partner, i.e., confidence in the binding between the entity and the presented identity.

- The process of establishing confidence is especially important when the communicating partners are remote from each other and connected only by a telecommunication link. The authentication process is executed in order to ascertain, with a sufficient degree of confidence that the identity presented by a communication partner really belongs to it. Communication always involves two or more distinct partners that exchange information. Due to the broad variety of possible partners (e.g., humans and things), a general term needs to be defined. The term chosen is entity which is defined as: something that has separate and distinct existence and that can be identified in context.

- The information that can be used for identification of an entity is based on the entity's attributes. An attribute is defined as: information bound to an entity that specifies a characteristic of the entity. In practical terms, identification of an entity is usually based on a subset of its attributes since identification is limited by what is called the context, within which the entity exists and interacts. The narrower the context and the clearer the boundary conditions, the fewer the number of attributes necessary for identification. Context is defined as: an environment with defined boundary conditions in which entities exist and interact.

- Since the definition of entity is based on the capability to be identified, it is necessary to have a proper definition of identification: the process of recognizing an entity as it is characterized within a context.
For the purpose of distinguishing entities, it is sufficient to use a sub-set of the attributes which is adequate to the context. This is referred to as the identity which is defined as: a representation of an entity in the form of one or more attributes that allow the entity or entities to be sufficiently distinguished within a context. For IdM purposes, the term identity is understood as a contextual identity (subset of attributes), i.e., the variety of attributes is limited by a framework with defined boundary conditions (the context) in which the entity exists and interacts.

An identity can be a subset of another identity. There also may be intersections of identities. However, for various reasons (such as for privacy concerns), intersections of identities, used for different purposes or in different contexts, may be explicitly undesirable or even excluded.

Figure xxx: Relationships between entity, identities and attributes (Source X.1252)

1.2 Level of Authentication Assurance

Degree of confidence reached in the authentication process that the entity is what it claims to be or is expected to be. NOTE – The assurance is based on the degree of confidence in the binding between the entity and the identity that is presented.
2. IBOPS Overview

a. Introduction to IBOPS

The Identity Biometric Open Protocol Standard (IBOPS) defines provides a guarantee that named users are who they claim to be. IBOPS identity assertion implies reliance on human biometrics, however IBOPS is an interoperable standard and can incorporate any number of identity asserters, be they biometrics-based or not, to allow retrofitting into existing identity assertion solutions.

The standard allows pluggable and interchangeable modules, including those that provide identity assertion, role gathering, assurance and auditing capabilities. IBOPS was designed to easily integrate into pre-existing identity management environments, where the pluggable components can replace the functionality of pre-existing components.

b. Architecture Overview

The IBOPS platform includes software running on client devices (eg mobile phones or desktop systems) and the back end IBOPS server.

The server infrastructure can easily be deployed within cloud or traditional on-premise environments as it integrated into web server systems as a web archive. The architecture is language-agnostic, allowing client-server and server-server communication via RESTful API’s and JSON.
The following diagram provides a high level overview of the IBOPS architecture, including data sets shared between the client and server, demonstrating the concept of a user's Account, a device, and an authenticated session. This is described in detail in the following sections.

The example below shows one example configuration. The keystore works in conjunction with the SSL/TLS layer in providing data confidentiality. The reference to SSL/TLS refers to the latest version of TLS, which provides transport layer data confidentiality.

The keys in the keystore are the keychain necessary for the hostname defined in the base key, from a valid certifying authority. This key along with the root and intermediary keys comprise a full key chain for the given hostname. This keystore works in providing the https for web access. This is differentiated from the truststore.

The truststore provides, through the use of a custom Certifying Authority (CA) the identity a given can be, at any point in time. The truststore performs the handshaking between a certificate created by a CA and the given Web Server. The information in the certificate is present in every call to the server and is used with an authentication token to specify an authentication state.

The use of the keystore and truststore is one common mechanism for data confidentiality. This specification allow other key mechanisms as long as the confidentiality of a keystore is maintained and the identity from a truststore is also maintained.
**Background**

A GENESIS is an established identity without a biometric. An entity may have one to many GENESIS processes according to user propositions.

Abbie, Andy, Cathy and Scott are all Entities. Scott owns a corporation called Computer Science Innovations, LLC which is also an Entity.  Abbie please add in Annex from X.1252.

Credentials are attributes of an Entity.

Entities have attributes. A set of attributes may be used to uniquely identify an Entity.
Identity a set of attributes related to an Entity.


Genesis is the process of defining an Entity to the System.

We can mutate attributes associated with Entity.

An Entity has a surrogate key called the Entity identifier, that is immutable and uniquely identifies the Entity.

Abbie add in Level of Assurance (LOA) levels - 4 levels. LOA levels are based on the combination of present attributes to the Entity. Each LOA level is defined as a minimal subset of authentication attributes.

First we have a Genesis, then we may enroll. Enrolment is the process of attaching additional attributes to an Entity. During Genesis we attach the identity attributes. During enrolment we attach the authentication attributes. Both the identity attributes and the authentication attributes, other than the Entity identifier may mutate over time.

Actors - Entity, User Agent, Identity Server, Authentication Server, Application Server

Precondition - A request to create an Entity is instantiated and Entity does not Exist

<table>
<thead>
<tr>
<th>Entity</th>
<th>User Agent</th>
<th>Identity Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request Access</td>
<td>Request Access (requestAccess (userId))</td>
<td>EntityDoesNotExist return code (-1)</td>
</tr>
<tr>
<td>Entity Does not exist</td>
<td>&lt;-----------------------------------</td>
<td></td>
</tr>
<tr>
<td>Request Create Entity</td>
<td>Create Entity createEntity (Map &lt;String&gt;, &lt;String&gt;)</td>
<td>Creation</td>
</tr>
</tbody>
</table>
Entity Identifier  return String Id
**Enrollment Flow:**

Pre-condition of a request to the entity for credential enrollment.

<table>
<thead>
<tr>
<th>Entity</th>
<th>User Agent</th>
<th>Identity Server</th>
<th>Authentication Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initiate credential enrollment w/ID</td>
<td>createEnrollment(entityId, Map &lt;String&gt;,&lt;String&gt;)</td>
<td>Retrieve identity record</td>
<td>return Map &lt;String&gt;&lt;String&gt;</td>
</tr>
<tr>
<td>Request 1-N type of credentials</td>
<td>Request enrollment of 1-N type of credentials</td>
<td>return Map &lt;String&gt;&lt;String&gt;</td>
<td></td>
</tr>
</tbody>
</table>

--- User makes choices ---

Present Initial credential data [....]

Option 1 (local authentication): enrollCredential(credential_type; credential_cert; signed nonce)

Option 2 (server authentication): enrollCredential(credential_type; byte_stream_auth_vector)

--- Store credential records ---

Return credential success & context; return enrollment_success

-- Store identity record update w/ credential type --
**Adjudication Flow:**

Pre-condition = Entity has existing, active identity within system
(Note: need new name for App server - possibly access control point. Also need to write clear assumptions of responsibilities for each and every actor. Especially app server since it hides much of the complexity of app access control, policy, and role based decision making.)

<table>
<thead>
<tr>
<th>Entity</th>
<th>User Agent</th>
<th>Ident Server</th>
<th>Auth Server</th>
<th>App Server</th>
</tr>
</thead>
</table>

Initiate access request to some app service w/ID access(serviceID; entityID)

> Request ID verification w/list of specific credentials or credential types

verify(entityID; MAP <credentialID> or <credentialGroup>)

--- Retrieve identity record ---

Request 1-N credential verifications
verify(entityID; credentialID)

1-N credential captures
capture(credentialID)

Authenticate UI

1-N credential responses
Option 1(local authentication): return(credential_type; credential_cert; signed nonce)
Option 2(server authentication): return(credential_type; byte_stream_auth_vector)

1-N credential results
return(success)

--- Identity server decides success/fail of overall verification ---

Return overall success / failure of verification
return(success)
GENESIS is an identity without a biometric. Devices have enrollments with initial biometrics. GENESIS is represented in storage as a singleton for each subject. Each subject may have many devices where each device has a biometric. The biometric per device may differ from device to device. For example, a phone may use a fingerprint as a biometric and a tablet may use an iris. For this example, the phone would have an initial enrollment of the fingerprint, and the tablet would have an initial enrollment of an iris.
3. IBOPS Identity Management

a. Identity and Genesis

Genesis is the initial creation of a subject. Genesis occurs once for each subject and is not bound to any particular device. A variety of options may occur to determine an identity for Genesis as the succeeding table shows.

<table>
<thead>
<tr>
<th>Genesis Level</th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Verify SMS</td>
<td>Verify SMS</td>
<td>Verify SMS</td>
</tr>
<tr>
<td></td>
<td>Verify Email</td>
<td>Verify Email</td>
<td>Verify Email</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ask 3 Questions</td>
<td>Ask 3 Questions</td>
</tr>
<tr>
<td></td>
<td></td>
<td>only know by</td>
<td>only know by</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Identity</td>
<td>Identity</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Previous Identity Information</td>
</tr>
</tbody>
</table>

As table 1 shows, we have 3 Genesis Levels. The levels allow certain privileges to occur and IBOPS may be queried for the Genesis level of any Subject. For Level 0, the users was confirmed by sending an SMS message and a confirming email. This is the lowest level of Genesis.

Level 1 uses the properties of Level 0 plus asking 3 questions that only the Subject should know. Proper answering of these questions places the Subject at Genesis Level 1.
Genesis Level 2 uses the properties of Level 1 plus using any other identity information an organization may have. This information was used prior to IBOPS to determine an identity.

During the Genesis process additional information comes to the IBOPS Server. Included in this information are two values (between 0 and 59) which are used as replay mitigation. Additionally, a certificate is given to the client which relates all subsequent application programming interface (API) calls to the subject. The diagram below illustrates the flow of a Genesis.

b. Identity and Enrollment

ABBIE we need an Enrollment matrix that show an enrollment level based on the quality of the biometric

The following diagram provides a high level overview of the Enrollment process:
The enrolment process links a pre-existing digital identity (from a Genesis) with a single biometrically-identifiable user linked to a device. Other devices are linked to the Genesis phase through an Enrollment of a second, third or nth device.

Depending upon implementation, the enrollment process may include verification of a pre-existing digital identity/account - for example, email address or phone number confirmation or validation utilizing a third party/pre-existing system. This can happen in several stages, and the order in which this happens depends on the specific product or solution. It is vendor dependent whether the certificate is tied to a Genesis, meaning the Subject has one and only one certificate, or alternately may use a separate certificate per enrollment and tie each enrollment to a Genesis in the IBOPS Server.
During enrollment the following steps take place.

**ABBIE** we need to consider the interface between the biometric device reader and the client device. How can we trust this device? How can we prevent replay?

1. **Obtain Client Certificate.**

A Client Certificate is generated on IBOPS Sever. This certificate is what represents a device/user pairing. The format of a certificate may vary from implementation to implementation. The following paragraph assumes an X.509 certificate, but it is the general concept which applies to the IBOPS standard. In summary, the certificate must be locked with a password or some referencing value, must be stored on the client device and must not store the password in plain text. The server must use the concept of a Certifying Authority and a truststore to create and validate the use of the client certificate.

For example, assume a Client Certificate is an X.509 certificate. This certificate contains a unique public key representing the device/user pair. This public key and the associated private key are generated on the server but not maintained or stored on the server.

The Client Certificate also contains an identifier known as a GUID (a part of a Distinguished Name in the certificate’s Subject field) that allows the server to refer to an individual user account. The GUID is unique to each client certificate/device.

The IBOPS server is also a Certificate Authority (CA), which signs a Client Certificate using the Server CA private key. The Server CA key pair and related certificate are referred as “Server CA keys” and “Server CA Certificate”. Those are stored by the IBOPS server’s host system.
A Server CA certificate, a Client Certificate, and an associated private key are incorporated into a PKCS #12 (p12) file, encrypted with a password¹, and sent to the client via an API call over the pre-established TLS session.

The encrypted p12 with its embedded private key, Client Certificate, and Server CA Certificate are employed to establish subsequent TLS sessions.

The encrypted p12 is stored on a device².

¹ More Details needed
² Refer to the section titled Client-side data for further information.
Client Certificate and associated private key example:

2. Biometric enrollment.

Biometric enrollment occurs as follows:

a. Acquire biometric factor data through the device (for example, face, fingerprint, iris etc).
b. Generate the biometric vector (hash).
c. Store biometric vector on the client for user authentication.

b. Authentication/Identity Assertion

i. Overview

Successful authentication of a user/device combination is required in order to allow access to requested resources. The IBOPS has multiple points of authentication to enable that:

- Initial client software is authenticated using embedded default certificates that are shared with the server within a standard TLS session handshake.

- After genesis/enrollment, users are authenticated using biometrics, matched locally on a mobile device.

- The pre-registered device authenticates to the server using a Client Certificate, shared over a standard TLS session handshake. The Client Certificate represents a pre-registered user/device pairing. A client certificate can't be submitted to a server until the certificate is decrypted with a password, which is obtained under a condition of a successful user authentication.

- Server authentication is also enabled using server certificates shared with the client during the TLS session handshake.

Each of these authentication points are described in detail in the following sections.

ii. Client Software Authentication

When installed, the client application comes pre-loaded with a default X.509 certificate that allows 2-way TLS-based authentication to be immediately enforced and active throughout the genesis/enrollment stages. This certificate allows establishment of an TLS session with a URI dedicated to allowing the initial stages of genesis alone, with a limited number of APIs available over that URI. After genesis/user enrollment, this default certificate is replaced with the Client Certificate.
iii. Post-enrolment device/user authentication

User Authentication:

A user is authenticated by matching biometric vectors and passing liveness tests. The user's biometrics, single factor or multi-factor, are acquired through the device (for example, face, fingerprint, iris etc). Biometric vector(s) are extracted from the acquired biometrics. The vector(s) are matched against saved biometric vector(s) obtained during genesis and enrollment.

IBOPS performs matching on the client side. After the client software validates the user via their biometrics, the biometrics authentication result is sent to the server regardless of the matching result.

In line with the IBOPS standard, all applications should include some form of liveness detection or an ability to ensure that the enrolled or authenticated user is an actual person and not an image or other representation of the user. For a face recognition system, it could be blink detection, for instance. Adjustable liveness detection levels are configurable on the Boriken server. Their adjustment affects false negative and false positive thresholds, timing, and usability. The choice of a liveness level is up to the organization using IBOPS and its particular needs.

Device authentication

IBOPS authenticates at two levels: within the underlying TLS session and within the server itself:

1. TLS-level authentication of the client device utilizes an X.509 certificate that was generated and signed by the Boriken server during user genesis/enrollment, linked to a stored user/device pairing, and then provided to the client. We refer to this as the Client Certificate. To authenticate a device, the client software unlocks the Client Certificate and submits it to the server as a part of the TLS session handshake. In line with the TLS handshake protocol the client then allows the Client Certificate to be verified by the server via the TLS “Certificate Verify” message. The Client Certificate is unlocked (decrypted) using a password obtained after successful authentication.

---

3 If enabled.
4 Because the TLS session is using a different client certificate to that used initially at genesis (which was the default certificate), the TLS session is established with a different URI endpoint.
2. After verification of the Client Certificate, the Boriken server login module extracts the client certificate’s device identifier (the GUID), looks up the appropriate account, checks for certificate revocation, determines roles associated with that account, and establishes an authenticated and authorized session.

iv. Server Authentication

Server authentication is achieved using the underlying TLS session. To meet the IBOPS requirements of two-way authentication between client and server, IBOPS requires TLS or TLS using client and server X.509 certificates.

The IBOPS server certificate used for TLS sessions is verified (signed) by a third party Certificate Authority, in this case GoDaddy, using their private key.

The client is able to verify the integrity of the IBOPS certificate by referring to the 3rd party CA (GoDaddy)’s public key, included as a certificate in the distribution of most common browsers and operating systems. This verification is completed as part of the standard SSL/TLS session handshake.

c. Authenticated Session Management

There are important characteristics of Session Management within IBOPS. First of all, the IBOPS protocol is stateless with respect to HTTP. This means that every call to a IBOPS-compliant server must have an identifier to the account, and the server must persist the current state. Through this mechanism IBOPS has session management. So iBOPS is HTTP stateless but stateful within its own context.

Sessions are tied to a specific user interacting with resources via the IBOPS server for a period of time. Each authorized user/device has its own unique Client Certificate, and that certificate contains a GUID which the server uses to identify the user. The identification of the User occurs by taking the GUID for the device and looking up the user for the device in the IBOPS server.

Initially the IBOPS server creates an opportunity for a session. This may be done in a variety of ways, but is most commonly done with a QR Code display. This call is a Session Opportunity meaning that the system displaying the QR Code may, after authentication, allow the establishment of a session. Or, alternately, the Session Opportunity may timeout or not authenticate.

In the case of successful user authentication and Session Opportunity, we have a Session.
Within the IBOPS server a persistent session is established specific to the account referenced via the Client Certificate.

Sessions may take in any kind of data as of tag/value pairs. This may be geospatial data, variable transaction amounts, or any other data we wish to store throughout the duration of the session. In the case of geo-fencing, which is wanting to control the physical location of a session, we add in latitude and longitude. In the case of an ATM, we add withdrawal amounts to the session.

At some point in time, the session completes. At session completion, all information about the session is written to a history/audit.

In the case of disallowing a device, user or account, the appropriate Client Certificates get revoked by removing the GUID from the IBOPS server. All future calls to the server will no longer result in successful communication.
d. Access Control

The Client Certificate is used to authenticate a user’s identity and to manage access control.

In line with the IBOPS standard, the IBOPS server enforces a mandatory access control policy over all subjects and objects\(^5\) under its control. The mandatory access control policy guarantees all data is labelled and all users have roles.

Roles are attributes associated with a user, such as “Administrator” or “Guest”, which are either gathered and stored in the server or a pre-existing enterprise system such as Active Directory. The Boriken server may be the definitive role source, may delegate the role source, or may work in conjunction with a role source. For example, some implementations of an IBOPS server stores role information that deals with security levels such as liveness. Alternately, a IBOPS Server may be implemented as part of an AD integration for a large enterprise. Active Directory may be the role source for a given organization, but the same organization may use the IBOPSServer to dictate more refined liveness for certain other roles. In this case the combination of the Boriken server and Active Directory create the authoritative role source, available via an API call.

Data is labeled with characteristics that describe access or access patterns - for example data may be labeled as “All”, allowing all access. The server maintains these sensitivity labels.

A mapping between roles and sensitivity labels allows us to implement role-based adjudication. Boriken server administrators can specify and control sharing of objects with subjects and determine if a given subject may read, write or execute an object via an API within a particular session.

Roles are typically hierarchical in nature. A subject can write an object only if the hierarchical classification in the subject’s security level is less than or equal to the hierarchical classification in the object’s security level and all the non-hierarchical categories in the subject’s security level are included in the non-hierarchical categories in the object’s security level.

To further illustrate, the IBOPS server receives a request. The request has a certificate which links the device to the user. The user has many roles so the roles (after the hierarchy is flattened) are stored in memory in something that the Java Authentication Services. The authenticated user is called a Principal. In subsequent processing, web pages may allow access for certain roles and data may have labels that correspond to the users roles. Data adjudication compares the data labels against the roles for the Principal. So access control is both on functionality and data.

\(^5\) A subject is an individual or group of named users, devices or programs. An object may be data, processes, files, segments, devices.
4. Intrusion Detection

a. Overview

The IBOPS standard requires the implementation of intrusion detection capabilities. Depending on the specific implementation, these capabilities may provide active monitoring and blacklisting functionality to help address attacks, including credential spoofing via certificate impersonation, session replay, detection and prevention of account brute-forcing, session creation failures, and/or the detection of denial-of-service (distributed or single DOS) attacks. These functions should be governed by one or more intrusion detection systems which can take appropriate actions such as terminating session creation or blacklisting devices and/or IP addresses.

The IBOPS standard also requires specific intrusion detection capabilities implemented within a system on both server and client sides. Client-side intrusion detection must be capable of identifying patterns of trial and error in order to blacklist itself, or if the user fails to authenticate more than a configurable consecutive times. On a typical IBOPS implementation this could be around 4. Blacklisting may be enacted for a fixed period of time or indefinitely until the device can be properly assured that it is safe and valid. The IBOPS server can query the Intrusion Detection System (IDS) whether a device is blacklisted or not. If the device is blacklisted, all communication ceases. Depending on implementation, blacklisting may occur at the device level, the IP address level, the subnet level or beyond. A typical implementation may only blacklist on Device ID.

The following outlines an intrusion detection function for IBOPS.

b. Replay Prevention

One of the largest threats to network based systems is replay. Consider the scenario of an attacker modifying target software on a device and/or stealing session information from a valid session for replay, in an attempt to gain unauthorized access to a server.

Replay mitigation uses replay check values that are built into the communication to help prevent this type of attack.

From the server side perspective, the server has to recognize that the replay check values are not correct when under attack so the device can be blacklisted. The replay check values are requested by the client during Genesis and are one way encrypted by the IBOPS Server. The Boriken server looks up the one way encryption to find the values used for replay mitigation. Subsequent client calls use the check values placed by the client and expected by the server. If
the values are as expected, communication continues. If the replay values are not expected the IDS receives a notification. Enough notifications and the client is blacklisted.

5. Standards, Audit, and Quality Control

a. Standards

IBOPS is designed to meet several minimum standards, including assessment against the Trusted Computer System Evaluation Criteria (TCSEC), which is a US Government Department of Defense (DoD) standard that sets basic requirements for assessing the effectiveness of computer security controls built into a computer system, the Director of the Central Intelligence Directive 6/3 protection levels 3, 4, and 5 (PL3, PL4, PL5), and the standards of Multiple Independent Levels of Security (MILS).

b. Audit Capabilities

Auditing ensures the appropriate artifacts for continued verification and validation of trusted access. All administrative actions (modification via the Admin dashboard) and authentication actions are captured and stored in a data repository. Account-specific authentication records can be audited by a user with Administrator privileges, via an administrator dashboard.
6. Data Protection

a. Network Transmissions

All network transmissions are required by the BOPS standard to run over Transport Layer Security (TLS) or its predecessor Secure Sockets Layer (SSL), with server and client-side X.509 certificates enforcing two-way authentication. Boriken servers (typically Apache Tomcat) are configured to use TLSv1.2.

Before a client and server can begin to exchange information protected by TLS, they must securely exchange keys and/or agree upon encryption keys and a cipher to use when encrypting data. These options are configurable and can also be negotiated on a per-session basis in line with client-side capabilities. For instance, if a client machine/application does not support TLSv1.2, then it may fallback to lower version.

Exact TLS configuration depends upon the IBOPS-compliant server configuration. Naturally, strong key generation and exchange algorithms are encouraged - similarly with encryption algorithms, in recognition of possible constraints such as compatibility or performance issues.

b. Data Protection Strategy

The general IBOPS standard philosophy is to prevent the centralized storage of sensitive data in one place. IBOPS does not create large server-side repositories of sensitive client data such as biometric images, for example.

Also as a general rule, encrypted data and the key to decrypt that data should not be in the same place. For example, in an IBOPS system, sensitive data such as the user’s website credentials are stored encrypted on the mobile device, with the associated key stored in the IBOPS server repository, attached to the user’s account. This helps prevent an attacker who has gained access to the mobile from reading the credentials, and vice-versa.

c. Client-side data

Sensitive data stored on the client side includes:

- the user’s biometric vector (NOT the full biometric)
• the encrypted p12 file containing Client Certificate and private key
• (optional or implementation specific) encrypted\(^8\) user authentication data for third party identity systems

On iOS, all of this data is stored in the iOS KeyChain. This is an encrypted container that holds passwords for multiple applications and secure services.

On Android the p12 is stored in the app shared preferences\(^9\):
/data/data/com.your.package/shared_prefs/com.your.package_preferences.xml

The biometric vector is stored internal to the client side application.

It is important to note that except for a handful of unique situations, a full biometric data set cannot be derived from a biometric vector, and that the possession of a biometric vector alone is not sufficient in order to obtain authenticated access to the IBOPS server. Identity Assertion under the IBOPS standard relies on several stages of authentication, including biometric vector matching, liveness checking, and the submission of the Client Certificate. Should the biometric vector be replaced/replayed, a Client Certificate must also be present and liveness must succeed in order to impersonate a user. In addition, IBOPS has replay protection mechanisms built in (see the section titled [Replay Prevention](#)).

d. Server-side data
   i. IBOPS Server Data

Sensitive data that may be stored within the Boriken server includes:

• Device identifiers
• Geospatial locations
• User identifiers such as email, phone number
• User roles
• Client Certificate password data
• Other possible attributes sent to the BOPS server during Genesis and/or session creation.

---

\(^7\) The sensitivity of the biometric vector is debatable, as a full biometric can usually not be extracted from the vector and in the event the vector is stolen we help address replay via the intrusion detection capabilities plus presence of the Client Certificate.

\(^8\) The user authentication data is stored in the same encrypted format that was returned from the Boriken server during genesis/enrollment. The data gets encrypted by the server using the Elliptic Curve Integrated Encryption Scheme (ECIES#) algorithm. The key used for this encryption (571 ECC) was generated and is stored on the server.

\(^9\) [http://developer.android.com/reference/android/content/SharedPreferences.html](http://developer.android.com/reference/android/content/SharedPreferences.html)
Depending on the IBOPS product or solution, session creation may add sensitive data. For example, the withdrawal of money from an ATM will add the account and amount as sensitive data.

Every session that is started and completed for and on behalf of a user or devices is also stored as an audit/history. This allows administrators or other privileged users to see usage patterns and other critical audit data.