Ongoing IREX

Ongoing IREX Evaluation
Concept, Evaluation Plan, and API Specification
Version 1.0

George W. Quinn
irex@nist.gov

Image Group
Information Access Division
Information Technology Laboratory

National Institute of
Standards and Technology
U.S. Department of Commerce

May 6, 2019

Status of this Document

This is the first public version of this document. Comments and questions should be submitted to irex@nist.gov. This document can be downloaded from http://www.nist.gov/itl/iad/ig/ongoing-irex.cfm.

Release Notes

The submission procedure, API, and implementation requirements for Ongoing IREX are similar to those of prior IREX evaluations.

Contents

1 Overview

1.1 Introduction

This document establishes a concept of operations (CONOPS) and application programming interface (API) for the Ongoing Iris Exchange (IREX) Evaluation. Ongoing IREX succeeds previous IREX evaluations.

IREX Program was initiated by NIST to support an expanded marketplace of iris-based applications. IREX I through IX provided quantitative support for iris recognition standardization, development, and deployment.

The latest information on IREX can be found on the IREX website (http://www.nist.gov/itl/iad/ig/irex.cfm).

1.2 Performance Metrics

Performance is assessed for one-to-one (a.k.a *verification mode*) matching. Below is a list of performance factors that might be posted to the Ongoing IREX homepage.

- **Matching Accuracy:** One-to-one matching is a binary classification problem. As such, accuracy is characterized by the trade-off between two types of classification error rates as a descrimination threshold is adjusted. This relationship is characterized by a DET (Detection Error Trade-off) curve [1]. DET curves have become a staple in biometric testing, superseding the analogous ROC (Receiver Operating Characteristic) curve. Compared to ROC curves, the logarithmic axes of DET curves provide a superior view of the differences between matchers in the critical high performance region. A comprehensive introduction to the fundamentals of assessing the accuracy of binary classification systems can be found in [2].
- **Timing Statistics:** Timing will be assessed as the elapsed real time (as opposed to CPU time) for core biometric operations (e.g. feature extraction, comparisons). Timing estimates will be computed on an unloaded machine running a single process. The machine's specifications are described in Section 1.4.
- **Accuracy-speed Trade-off:** Ongoing IREX might investigate whether a collective accuracy-speed trade-off exists for matchers submitted to Ongoing IREX. Accuracy-speed trade-off refers to the circumstance whereby more accurate matchers take longer to complete their operations.
- **Template Sizes:** The size of the proprietary templates generated by the implementation.
- **Runtime Memory Usage:** Ongoing IREX might monitor runtime memory usage during matching and/or feature extraction.

1.3 Iris Datasets

1.3.1 The OPS Dataset (TBD)

Ongoing IREX will begin by testing over a set of field collected iris images. The samples will be pulled from the same source as OPS II and III (used in IREX IV and IX respectively). The samples are collected from various locations over a period of years. Field collected samples tend to suffer more from quality related problems (e.g. motion and focus blur) than samples collected in more controlled laboratory settings.

The dataset is sequestered (i.e. not publicly available). The participants are not allowed to view any of the iris samples and will not be provided with a representative set of iris samples.

1.3.2 Ground Truth Integrity

A hazard with collecting operational data is that ground truth identity labels can be incorrectly assigned due to clerical error. NIST will attempt to correct ground truth errors in its test datasets whenever possible, and only when doing so will not unfairly bias results in favor of certain submissions over others.

1.4 Test Environment

Hardware specifications for some of NIST's test machines are:

- Dell M610 - Dual Intel Xeon X5680 3.3 GHz CPUs (6 cores each).
- Dell M910 - Dual Intel Xeon X7560 2.3 GHz CPUs (8 cores each).
- Dual Intel Xeon E5-2695 3.3 GHz CPUs (14 cores each; 56 logical CPUs total).

The test machines will have CentOS 7.2 installed, which runs Linux kernel 3.10.0 (http://www.centos.org/). An ISO image of the distribution can be downloaded from NIGOS (http://nigos.nist.gov:8080/evaluations/CentOS-7-x86_64-Everything-1511.iso).

1.5 Posting of Evaluation Results

NIST will post performance results for each successfully validated implementation to the Ongoing IREX homepage (http://www.nist.gov/itl/iad/ig/ongoing-irex.cfm) as soon as they become available. The participant will be notified once posted.

2 Participation Requirements

2.1 Participant Requirements

Participation is open to any commercial organization or academic institution that has the ability to implement an iris matching algorithm. There is no charge and participation is open worldwide.

A high-level description of the process is as follows:

1. Mail the completed Participation Agreement to NIST.
2. Format your matching software to adhere to the API specifications and runtime requirements described in the CONOPS document.
3. Submit your implementation to NIST, following the cryptographic protection procedures described in Section 2.2.
4. NIST will validate your submission to ensure its correct operation (Section 2.3).
5. NIST will assess the performance of your implementation and post the results to the Ongoing IREX Homepage.

Participants only need to complete the Participation Agreement the first time they submit an implementation to Ongoing IREX.

Participants are permitted to submit no more than two implementations over a 6 month period.

2.2 Submission Procedure

All software, data, and configuration files submitted to NIST must be signed and encrypted. Signing is performed to ensure authenticity of the submission (i.e. that it actually belongs to the participant). Encryption is performed to ensure privacy. The full process is described at http://biometrics.nist.gov/cs_links/iris/irex/NIST_biometrics_crypto2.pdf.

Implementations shall be submitted to NIST as encrypted *gpg* files. If the encrypted implementation is below 20MB, it can be emailed directly to NIST at irex@nist.gov. If the encrypted implementation is above 20MB, it can either be provided to NIST as a download from a webserver. NIST shall not be required to register or enroll in any kind of membership before downloading the implementation.

Note: NIST will not accept any implementations that are not signed and encrypted. NIST accepts no responsibility for anything that occurs as a result of receiving files that are not encrypted with the NIST public key.

2.3 Software Validation

Upon receipt, NIST will validate the implementation to ensure its correct operation. The validation process involves running the implementation over a small sample of test data. This test data will be provided to the participant, who must run the implementation in-house and provide NIST with the comparison results. NIST will then verify that the participant's in-house results are consistent with the output produced on the NIST blades. The validation data along with full instructions are posted on the Ongoing IREX homepage (http://www.nist.gov/itl/iad/ig/ongoing-irex.cfm).

2.4 Submission Requirements

2.4.1 Linking Requirements

Participants shall submit their implementations as pre-compiled and linkable libraries. Dynamic libraries are permitted, but static ones are preferred. Participants shall *not* provide any source code. Header files should not be necessary, but if provided, should not contain intellectual property of the company nor any material that is otherwise proprietary.

NIST will link the submitted library file(s) to our ISO 2011 C++ language test drivers. Participants are required to provide their libraries in a format that is linkable using g++ version 4.8.5. Thus, libraries must export their functions according to the C++11 naming convention. The functions that must be exported are defined in "irex.h" and described in Section 3. The software libraries must be 64-bit.

Participants may provide customized command-line linking parameters. A typical link line might be:

```
g++ -I. -Wall -m64 -o irex_main irex_main.c -L. -lirex_thebes_A_01 -lpthread
```

NIST will ignore requests to alter parameters by hand (e.g. modify specific lines in an XML configuration file). Any such adjustments must be submitted as new implementations.

Participants are strongly advised to verify library-level compatibility with g++ (on an equivalent platform) prior to submitting their software to NIST to avoid linkage problems (e.g. symbol name and calling convention mismatches, incorrect binary file formats, etc.). Intel ICC is not available. Access to GPUs is not permitted. Intel Integrated Performance Primitives (IPP) libraries are permitted if they are delivered as part of the developer-supplied library package. It is the provider's responsibility to establish proper licensing of all libraries.

Dependencies on external dynamic/shared libraries such as compiler-specific development environment libraries are discouraged. If absolutely necessary, external libraries must be provided to NIST after receiving prior approval from the test liaison. Image processing libraries such as libpng and NetPbm should not be required since NIST will handle image reading and decompression.

2.4.2 Naming Convention

All submitted libraries shall adhere to the naming convention described in Table 1. Additional dynamic or shared library files may be submitted that support this core library.

Table 1: Naming convention for an implementation library.

Form:	libIREX_provider_sequence.suffix			
Part:	libIREX	provider	sequence	suffix
Description:	First part of the name, fixed for all submissions	a single word name of the main provider. EXAMPLE: thebes	A two-digit decimal identifier starting at 00 and incrementing any time a new submission is sent to NIST	Either .so or .a
Example:	libIREX_thebes_01.a			

2.4.3 Installation Requirements

Installation Must be Simple Installation shall require the simple copying and/or decompression of files followed by a linking operation. There shall be no need for interaction with the participant provided everything goes smoothly. It shall not require an installation program.

No License Requirements or Usage Restrictions The implementation shall allow itself to be executed on any number of machines without the need for machine-specific license control procedures or activation. The implementation shall neither implement nor enforce any usage controls or restrictions

based on licenses, number of executions, presence of temporary files, etc. No activation dongles or other hardware shall be required.

Sufficient Documentation Must be Provided Documentation should be provided for all (non-zero) participant-defined error or warning return codes.

Disk-Space Limitations The implementation may use configuration files and supporting data files. The total size of all libraries and configuration and data files for any given submission shall not be more than a gigabyte.

2.4.4 Runtime Requirements

Single-threaded Requirement Implementations must run in single-threaded mode.

No writing to Standard Error or Standard Output The implementation will be tested in a non-interactive ”batch” mode without terminal support. Thus, the submitted library shall run quietly (i.e. it should not write messages to ”standard error” or ”standard output”. An implementation may write debugging messages to a log file. This log file must be declared in the documentation.

Exception Handling Should be Supported The implementation should support error/exception handling so that, in the case of an unexpected error, a return code is still provided to the calling application. The NIST test harness will gracefully terminate itself if it receives an unexpected return code, as it usually indicates improper operation of the implementation.

No External Communication Implementations running on NIST hosts shall not side-effect the runtime environment in any manner except through the allocation and release of memory. Implementations shall not write any data to an external resource (e.g. a server, connection, or other process). Implementations shall not attempt to read any resource other than those explicitely allowed in this document. If detected, NIST reserves the right to cease evaluation of the software, notify the participant, and document the activity in published reports.

Components Must be Stateless All implementation components shall be ”stateless” except as noted elsewhere in this document. This applies to iris detection, feature extraction and matching. Thus, all functions should give identical output, for a given input, independent of the runtime history. NIST will institute appropriate tests to detect stateful behavior. If detected, NIST reserves the right to cease evaluation of the software, notify the participant, and document the activity in published reports.

Minimum Speed Requirements The implementations shall perform operations within the time constraints specified by Table 2. These time limits apply to the function call invocations defined in Section 2 on a Dell M910 system described in Section 1.4. Since NIST cannot regulate the maximum runtime per operation, limitations are specified as 90th percentiles (i.e. 90% of all calls to the function shall complete in less time than the specified duration). The limitations assume each template is generated from a single iris image.

Table 2: Time limitations for specific operations.

Operation	Timing Restriction
Creation of a verification template from a single 640x480 pixel image	1,000 ms
Creation of an enrollment template from a single 640x480 pixel image	1,000 ms
Comparison between two templates generated from a single image each.	50 ms

3 API Specification

The design of this API reflects the following testing objectives:

- Support black-box testing.
- Support distributed processing.
- Support graceful and informative failure recovery.
- Support the ability to collect performance statistics (e.g. timing, template size).

Submitted library files must export and properly implement the functions defined in this section. Testing will proceed in two phases: (1) feature extraction / template generation followed by (2) template comparison. This basic program flow is detailed in Table 3.

Table 3: Program Flow

Stage	Function	Metrics
Feature Extraction	**convert_multiiris_to_enrollment_template()** Generates an enrollment template from one or more iris images from an individual. The implementation must be able to handle multiple calls to this function from multiple instances of the calling application.	Template size and generation time.
	convert_multiiris_to_verification_template() Generates a verification template from one or more iris images from an individual.	Template size and generation time.
Comparison	**match_templates()** Compares a verification template to an enrollment template.	Accuracy and comparison time.

3.1 Functions

Functions

- int32_t get_max_template_sizes (uint32_t &max_enrollment_template_size, uint32_t &max_↩ verification_template_size)
  Retrieves the maximum (per image) verification and enrollment template sizes.
- int32_t convert_multiiris_to_verification_template (const std::vector< iris_sample > &input_irides, uint32_t &template_size, uint8_t *verification_template)
  Generates a verification template from a vector of iris samples.
- int32_t convert_multiiris_to_enrollment_template (const std::vector< iris_sample > &input_irides, uint32_t &template_size, uint8_t *enrollment_template)
  Generates an enrollment template from a vector of iris samples.
- int32_t match_templates (const uint8_t *verification_template, const uint32_t verification_template↩ _size, const uint8_t *enrollment_template, const uint32_t enrollment_template_size, double &dissimilarity)
  Searches a verification template against an enrollment template and produces a dissimilarity score.

3.1.1 get_max_template_sizes()

```
int32_t IREX::get_max_template_sizes (
              uint32_t & max_enrollment_template_size,
              uint32_t & max_verification_template_size )
```

Retrieves the maximum (per image) verification and enrollment template sizes.

These values will be used by the test harness to pre-allocate space for template data. For a vector of K iris samples, the test-harness will pre-allocate K times the provided value before calling convert_multiiris_to_verification_template() or convert_multiiris_to_enrollment_template().

Parameters

out	*max_enrollment_template_size*	The maximum (per image) size of an enrollment template in bytes.
out	*max_verification_template_size*	The maximum (per image) size of a search template in bytes.

Returns

Zero indicates success. Other values indicate a vendor-defined failure.

3.1.2 convert_multiiris_to_verification_template()

```
int32_t IREX::convert_multiiris_to_verification_template (
              const std::vector< iris_sample > & input_irides,
              uint32_t & template_size,
              uint8_t * verification_template )
```

Generates a verification template from a vector of iris samples.

The function must be able to handle multiple iris samples when provided (e.g. both left and right eyes from the same person). If the function returns a zero exit status, the template will be used for matching.If the function returns a value of 8, NIST will debug. Otherwise, a non-zero return value will indicate a failure to acquire and the template will *not* be used in subsequent search operations.

Parameters

in	*input_irides*	The iris samples from which to generate the template.
out	*template_size*	The size, in bytes, of the output template
out	*verification_template*	Template generated from the iris samples. The template's format is proprietary and NIST will not access any part of it. The memory for the template willbe pre-allocated by the NIST test harness. The implementation shall *not* allocate this memory.

Returns

Return Value	Meaning
0	Success.
2	Elective refusal to process the MULTIIRIS.
4	Involuntary failure to extract features.
6	Elective refusal to produce a (non-blank) template.
8	Cannot parse the input data.
Other	Vendor-defined failure.

If there are multiple iris samples, then a zero status should be returned as long as feature information could be extracted from at least one of the images.

3.1.3 convert_multiiris_to_enrollment_template()

```
int32_t IREX::convert_multiiris_to_enrollment_template (
              const std::vector< iris_sample > & input_irides,
              uint32_t & template_size,
              uint8_t * enrollment_template )
```

Generates an enrollment template from a vector of iris samples.

The function must be able to handle multiple iris samples when provided (e.g. both left and right eyes from the same person). If the function returns a zero exit status, the template will be used for matching.If the function returns a value of 8, NIST will debug. Otherwise, a non-zero return value will indicate a failure to acquire and the template will *not* be used in subsequent search operations.

Parameters

in	*input_irides*	The iris samples from which to generate the template.
out	*template_size*	The size, in bytes, of the output template.
out	*enrollment_template*	Template generated from the iris samples. The template's format is proprietary and NIST will not access any part of it. The memory for the template will be pre-allocated by the NIST test harness. The implementation shall *not* allocate this memory.

Returns

Return Value	Meaning
0	Success.
2	Elective refusal to process the MULTIIRIS.
4	Involuntary failure to extract features.
6	Elective refusal to produce a (non-blank) template.
8	Cannot parse the input data.
Other	Vendor-defined failure.

3.1.4 match_templates()

```
int32_t IREX::match_templates (
              const uint8_t * verification_template,
              const uint32_t verification_template_size,
```

```
const uint8_t * enrollment_template,
const uint32_t enrollment_template_size,
double & dissimilarity )
```

Searches a verification template against an enrollment template and produces a dissimilarity score.

Parameters

in	*verification_template*	A template generated by a call to convert_multiiris_to_verification_template().
in	*verification_template_size*	The size, in bytes, of the verification template.
in	*enrollment_template*	A template generated by a call to convert_multiiris_to_enrollment_template().
in	*enrollment_template_size*	The size, in bytes, of the enrollment template.
out	*dissimilarity*	A non-negative measure of the amount of dissimilarity between the templates.

Returns

Return Value	Meaning
0	Success.
2	One or more of the input templates were the result of a failed feature extraction.
Other	Vendor-defined failure.

3.2 Supporting Data Structures

This section describes the data structures used by the API.

3.2.1 point Struct Reference

A structure that specifies a coordinate in an image.

Public Attributes

- uint16_t x
  X-coordinate (0 == leftmost)
- uint16_t y
  Y-coordinate (0 == topmost)

Detailed Description A structure that specifies a coordinate in an image.

3.2.2 iris_sample Struct Reference

Defines a structure that holds a single iris with corresponding attributes.

Public Attributes

- eye_label eye
  The eye label for this iris sample (0 == undefined, 1 == right eye, 2 == left eye).
- uint16_t image_width
  Image width in pixels.

- uint16_t image_height
  Image height in pixels.
- uint16_t wavelength
  The wavelength with which the iris sample was acquired.
- uint8_t bit_depth
  Bit depth, 8 or 24 for RGB visible spectrum images.
- uint8_t * data
  Pointer to image raster data (RGBRGBRGB... for 24-bit images).
- iris_boundary * boundary
  Always set to NULL for now.

Detailed Description Defines a structure that holds a single iris with corresponding attributes.

Member Data Documentation

image_width `uint16_t image_width`

Image width in pixels.

image_height `uint16_t image_height`

Image height in pixels.

wavelength `uint16_t wavelength`

The wavelength with which the iris sample was acquired.

bit_depth `uint8_t bit_depth`

Bit depth, 8 or 24 for RGB visible spectrum images.

data `uint8_t* data`

Pointer to image raster data (RGBRGBRGB... for 24-bit images).

boundary `iris_boundary* boundary`

Always set to NULL for now.

3.2.3 iris_boundary Struct Reference

A structure that holds manual segmentation information for an iris sample.

Public Attributes

- point center
  Coordinate representing manual estimate of iris center.
- std::vector< point > pupil_boundary
  Vector of points outlining pupil boundary.
- std::vector< point > limbic_boundary
  Vector of points outlining limbic (iris-sclera) boundary.
- std::vector< point > upper_eyelid_boundary
  Vector of points outlining upper eyelid boundary.
- std::vector< point > lower_eyelid_boundary
  Vector of points outlining lower eyelid boundary.

Detailed Description A structure that holds manual segmentation information for an iris sample.

Member Data Documentation

center `point center`

Coordinate representing manual estimate of iris center.

pupil_boundary `std::vector<point> pupil_boundary`

Vector of points outlining pupil boundary.

limbic_boundary `std::vector<point> limbic_boundary`

Vector of points outlining limbic (iris-sclera) boundary.

upper_eyelid_boundary `std::vector<point> upper_eyelid_boundary`

Vector of points outlining upper eyelid boundary.

lower_eyelid_boundary `std::vector<point> lower_eyelid_boundary`

Vector of points outlining lower eyelid boundary.

References

[1] A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki. The DET curve in assessment of detection task performance. In *Proc. Eurospeech*, pages 1895–1898, 1997. 1

[2] Charles E. Metz. Basic principles of ROC analysis. In *Seminars in Nuclear Medicine*, pages 8–283, 1978. 1

Application to Participate in Ongoing IREX

1. Who Should participate

1.1. Organizations ("Organizations") that develop iris matching software are eligible to participate.

1.2. Anonymous participation will not be permitted. This means that signatories to this document, *Application and Agreement to Participate in Ongoing IREX*, acknowledge that they understand that the results of the test will be published with attribution to their Organization.

2. How to Participate

2.1. In order to participate, an Organization must complete this form and mail it to the location designated in Section 7.

2.1.1. The Responsible Party is an individual with the authority to commit the Organization to the terms in this Agreement.

2.1.2. The Point of Contact is an individual within the Organization with detailed knowledge of the submitted software implementation.

2.1.3. The Responsible Party and Point of Contact may be the same person.

2.2. Upon receipt of the signed agreement, NIST will classify the Organization as a "Participant" in Ongoing IREX. Applicants need only send a completed application for their first submission. Participants need not send a new participation form for subsequent submissions.

2.3. Participant shall provide submissions ("Submissions") as specified in Section 2 of the Ongoing IREX Concept, Evaluation Plan, and API Specification. A Submission shall include all library files, configuration files, documentation, and all other files required by NIST and the Participant to validate and execute the tests specified in the Test Plan.

2.4. The Submission need not be used in a production system or be commercially available. However, the Submission must, at a minimum, be a stable implementation capable of conforming to the Test Plan that NIST has published for Ongoing IREX.

2.5. The Submission must be encrypted before transmission to NIST. Instructions for submitting can be found at http://biometrics.nist.gov/cs_links/iris/irex/NIST_biometrics_crypto2.pdf. Generic encryption instructions can be found in the Image Group's *Encrypting Software for Transmission to NIST* document available at http://www.nist.gov/itl/iad/ig/encrypt.cfm. A box for the Participant's public key fingerprint is included on the Agreement. Submissions that are not signed with the public key fingerprint listed on the Agreement will not be accepted.

2.6. Submissions must be compliant with the Test Plan, NIST test hardware, and NIST test software.

3. Points of Contact

3.1. The Ongoing IREX Liaison is the U.S. Government point of contact for Ongoing IREX.

3.2. All questions should be directed to the irex@nist.gov, which will be received by the Ongoing IREX Liaison and other IREX personnel.

4. Release of Results

4.1. After successful completion of testing, NIST will publish the results along with the Organization's name on the Ongoing IREX website.

4.2. Participant will be notified of the results via the Responsible Party and the Point of Contact provided on the Agreement.

4.3. After the release of the results, Participant may use the results for their own purposes. Such results shall be accompanied by the following phtase: "Results shown from NIST do not constitute an endorsement of any particular system, product, service, or company by the U.S. Government." Such results shall also be accompanied by the Internet address (URL) of the Ongoing IREX website (http://www.nist.gov/itl/iad/ig/ongoing-irex.cfm).

5. Additional Information

5.1. Any data obtained during Ongoing IREX, as well as any documentation required by the U.S. Government from the Participant (except the Submission), becomes the property of the U.S. Government. Participant will not acquire a proprietary interest in the data and/or submitted documentation. The data and documentation will be treated as sensitive information and only be used for the purposes of NIST tests.

5.2. Participant agrees that they will not file any Ongoing IREX related claim against IREX sponsors, supporters, staff, contractors, or agency of the U.S. Government, or otherwise seek compensation for any equipment, materials, supplies, information, travel, labor and/or other Participant-provided services.

5.3. The U.S. Government is not bound or obligated to follow any recommendations that may be submitted by the Participant. The U.S. Government, or any individual agency, is not bound, nor is it obligated, in any way to give any special consideration to Participant on future contracts.

5.4. By signing this Agreement, Participant acknowledges that they understand any test details and/or modifications that are provided in the Ongoing IREX website supersede the information on this Agreement.

5.5. Participant may withdraw from Ongoing IREX at any time before their Submission is received by NIST, without their participation and withdrawal being documented on the Ongoing IREX website.

5.6. NIST will use the Participant's Submission only for NIST tests, and in the event errors are subsequently found, to re-run prior tests and resolve those errors.

5.7. NIST agrees not to use the Participant's Submission for the purposes other than indicated above, without the express permission by the Participant.

6. Reminders

6.1. NIST requests that applicants send an email to irex@nist.gov after they have sent their applications. NIST will respond with a confirmation message upon receipt of the application.

6.2. See http://www.nist.gov/itl/iad/ig/irexix.cfm for the latest updates and information on Ongoing IREX.

7. Application Submission

7.1. Please mail the completed and signed Agreement to:

```
Ongoing IREX Test Liason (A214)
100 Bureau Drive
A214/Tech225/Stop 8940
NIST
Gaithersburg, MD 20899-8940
USA
```

Organization Name

Responsible Party

Full Name		
Address (Line 1)		
Address (Line 2)		
Address (Line 3)		
Phone Number	Fax Number	E-mail Address

Point of Contact Check if same as Responsible Party above: ☐

Full Name		
Address (Line 1)		
Address (Line 2)		
Address (Line 3)		
Phone Number	Fax Number	E-mail Address

Participant **must** complete the box below per the instructions for transmission of encrypted content to NIST, as defined at http://biometrics.nist.gov/cs_links/iris/irex/NIST_biometrics_crypto2.pdf. If preferred, Participant may fax their public key fingerprint to the Ongoing IREX Liaison at (301) 975-5287.

Public Key Fingerprint

Participant
NIST `A75C EECD EF65 3197 7E66 A960 67D0 4015 407A D929`

With my signature, I hereby request consideration as a Participant in the Ongoing IREX Evaluation, and I am authorizing my Organization to participate in Ongoing IREX according to the rules and limitations listed in this document.

With my signature, I also state that I have the authority to accept the terms stated in this Agreement.

Signature of Responsible Party Date