SCAPI Release Notes

April 8, 2014

# Introduction

SCAPI is an open-source Java library for implementing secure two-party and multiparty computation protocols (SCAPI stands for the "Secure Computation API"). It provides a reliable, efficient, and highly flexible cryptographic infrastructure. SCAPI is comprised of three layers:

1. Low-level cryptographic functions: these are functions that are basic building blocks for cryptographic constructions (e.g., pseudorandom functions, pseudorandom generators, discrete logarithm groups, hash functions and more belong to this layer).
2. Non-interactive mid-level cryptographic functions: these are non-interactive functions that can be applications within themselves in addition to being tools (e.g., encryption and signature schemes belong to this layer).
3. Interactive cryptographic protocols: these are interactive protocols involving two or more parties; typically, the protocols in this layer are popular building blocks like commitment schemes, zero knowledge proofs and oblivious transfer. This layer is yet to be released.

See <http://crypto.biu.ac.il/about-scapi.php> for more information about SCAPI.

# Version 2.3.0

## Release Date

July 31, 2014

## New Features and Functionality

1. An OpenSSL implementation of DSA signature scheme has been added.
2. An OpenSSL implementation of RSAPss signature scheme has been added.
3. An OpenSSL implementation of RSAOaep encryption scheme has been added.

## Bug Fixes, Changes and Added Features to Existing Packages (By Package)

**jni::MiraclJavaInterface**

1. **ECF2mPoint** – createF2mPointFromX and createRandomF2mPoint functions were not in use and have been deleted.
2. **ECFpPoint** – createFpPointFromX and createRandomFpPoint functions were not in use and have been deleted.

**jni::OpenSSLJavaInterface**

1. **DlogEC** and **DlogF2m**– In case of error, deletion of the allocated memory has been added.
2. **DlogFp** – The function encodeByteArrayToPoint has been added.
3. Seeding the PRNG was changed to use OpenSSL's RAND\_screen() and RAND\_poll() functions instead of hardcoded seed.

**jni::OtExtensionJavaInterface**

**OtExtension** – deleteSender and deleteReceiver functions have been added.

**Circuits::garbledCircuit**

The garbled circuit package has been widely changed.

The main change is that instead of having two circuit classes that differ only in the way of garbling (using seed or Encryption scheme) we now have one "basic" circuit and an "extended" circuit.

The basic circuit implements the basic circuit's functionalities (like garble, compute, translate) and the extended circuit adds some features like the ability to give some of the wires' keys, verify function, etc.

More details about that can be found at [GarbledCircuitManual.docx](https://github.com/cryptobiu/scapi/blob/master/doc/old/UserManual/GarbledCircuitManual.docx).

The specific changes in the code are:

1. The word "label" in the term of number of wire has been changed to "index". Respectively, "labels" has been changed to "indices".
2. **GarbledTabledHolder** changed to an interface. Two concrete classes have been added: **BasicGarbledTablesHolder** and **ExtendedGarbledTablesHolder**. One for each corresponding circuit class.

Each circuit's utility class added a check in the beginning of the function that the given holder is an instance of the expected concrete class.

1. **CircuitInput** classes have been changed to **GarblingParamters**. For example, FreeXORCircuitInput has been changed to FreeXORGarblingParameters.
2. Both constructors of **GarblingParameters** derived classes (FreeXORGarblingParameters, StandardGarblingParameters, MinimizeAESSetKeyGarblingParameters) have been merge to one constructor that gets a boolean indicates if the circuit should use row reduction or not.
3. **GarblingParameters** - isRowReductionWithFixedOutputKeys function has been deleted. We no longer need this Boolean since the option to have fixed output keys has been moved to the extended circuit.
4. **CircuitSeedCreationValues** has been deleted.
5. **CircuitTypeUtil** and all derived classes - garble function that accepts part of the wire's keys has been deleted. The option to give some of the keys moved to the extended circuit and deleted from the utility classes.
6. **CircuitTypeUtil** and all derived classes - garble function that accepts the seed returns CircuitCreationValues instead of CircuitSeedCreationValues. It also does not get the hash argument as before. This is because the computing of the hash function on the garbled circuit moved to the extended circuit.
7. **CircuitTypeUtil**::getHashedTables has been deleted for the same reason.
8. **FreeXORGarbledBooleanCircuitUtil** – The option to give in the garble function some of the keys has been removed. It includes deletion of the following functions: createNonOutputWireValues

generateBothStandardValues

generateBothXORNOTValues

generateBothXORValues

createGarbledTables

1. **FreeXORGarbledBooleanCircuitUtil** and **StandardGarbledBooleanCircuitUtil** – Computing the hash function on the garbled tables has been deleted from the garble function with the seed. It includes deletion of the createGarbledTables (with hash) function.
2. **FreeXORGarbledBooleanCircuitUtil** and **StandardGarbledBooleanCircuitUtil** – The implementation of getHashedTables(…) has been deleted. It includes the computeHashOnTables function.
3. **FreeXORRowReductionGarbledBooleanCircuitUtil** and **MinimizeAESSetKeyRowReductionGarbledBooleanCircuitUtil** and **StandardRowReductionGarbledBooleanCircuitUtil** - isRowReductionWithFixedOutputKeys member has been deleted. We no longer need this Boolean since the option to have fixed output keys has been moved to the extended circuit. For the same reason the Boolean parameter of the constructor has been removed and also the checkOutputGate() function.
4. **FreeXORRowReductionGarbledBooleanCircuitUtil** and **StandardRowReductionGarbledBooleanCircuitUtil** – The garble function that gets some of the keys has been removed.
5. **GarbledBooleanCircuit and GarbledBooleanCircuitImp**:

* Garble and verify functions that used to get a BooleanCircuit now do not take any arguments. The circuit saves the Boolean circuit given in the constructor and uses it instead.
* Garble function that gets partial keys has been deleted.
* Garble function that gets a seed has been added.
* SetInput functions do not take the partyNumber argument anymore. It set each given input using the wire's index.
* The internalVerify function has been added. This function behaves exactly as the other verify method except the last part. The other verify function verifies that the translation table matches the resulted output garbled values, while this function does not check it. Instead, it returns the resulted output garbled values.
* The verifiedTranslate function has been added. This function verifies that the given garbledOutput is valid according to the given all OutputWireValues. Then, translates it.
* getGarbledTables and setGarbledTables gets/sets an instance of GarbledTablesHolder instead of a byte[][].
* The getOutputWireIndices and getNumberOfParties functions have been added.

1. The **GarbledBooleanCircuitAbs** class has been added. This is an abstract class that holds all the common members and functionalities of circuits.
2. The **GarbledBooleanCircuitExtended** interface has been added. It includes the following functions:

* setInputKeys – Get the garbled values of the input wires and sets them.
* setOutputKeys – Get the garbled values of the output wires and sets them.
* getHashedCircuit - Compute the hash function on the garbled tables and translation table of the circuit.
* verifyHashedCircuit - Verifies that the given hashedCircuit is indeed the result of the given hash on the circuit's garbled tables and translation table.
* verify(allInputWireValues, allOutputWireValues) - This verify function is the same as the verify of the basic circuit but also check that the resulted output garbled values are equal to the given allOutputWireValues.
* verify(seed, allInputGarbledValues, allOutputGarbledValues, hash, hashedCircuit) - This verify function samples the keys using the given seed, then compute the hash function on the circuit's garbled tables and

translation table and checks that the result is equal to the given hashedCircuit.

1. The **GarbledBooleanCircuitExtendedImp** class has been added. This class implements the GarbledBooleanCircuitExtended interface.
2. **GarbledBooleanCircuitImp**:

* A default constructor (that gets only a Boolean circuit) has been added.
* The GarbledBooleanCircuitImp(CircuitInput input, byte[][] garbledTables, HashMap<Integer, Byte> translationTable) constructor has been deleted.
* The GarbledBooleanCircuitImp(GarblingParameters input, PseudorandomGenerator prg) constructor has been added.
* Some members and functions have been moved to the new abstract class.

1. **GarbledBooleanCircuitSeedGeneration** and **GarbledBooleanCircuitSeedGenerationImp** have been deleted.
2. The **GarbledTablesHolder** interface has been added. The previous class with the same name has been renamed to "**basicGarbledTablesHolder**".
3. The **IdentityGate** class has been added. It is used by the extended circuit.
4. **MinimizeAESSetKeyGarbledGate**, **MinimizeAESSetKeyRowReductionGate**, **StandardGarbledGate** and **StandardRowReductionGarbledGate** - The constructor that gets the keys of the wires has been deleted since it is no longer in use.
5. **StandardGarbledBooleanCircuitUtil** – The default encryption scheme has been changed from HashingMultiKeyEncryption to AESFixedKeyMultiKeyEncryption.
6. **StandardGarbledBooleanCircuitUtil** – The option to give in the garble function some of the keys has been removed.

**Comm:**

ListeningThread constructor accepts a Party instead of a port. Instead of construct a channel using the localhost we take the IP address from the given party.

**Exception:**

1. The following exception have not been used and thus deleted:
2. InitializationException
3. InputNotSetException
4. UnInitializedException

**interactiveMidProtocols**:

Added references to the paper and pseudocode document in all third layer's protocols.

**interactiveMidProtocols::commitmentScheme:**

1. Added the option to execute the commitment scheme offline. In order to do that, the following functions have been added:

Committer interface: generateCommitmentMsg and functions.

Verifier interface: verifyDecommitment.

1. Each concrete commitment scheme now implementing these functions. The code was changed accordingly.
2. **CmtCCommitmentMsg** and **CmtCDecommitmentMessage** are now implementing the Serializable interface, since there may be a situation that a message should be sent over a channel without knowing the specific type. To enable that the interfaces should be declared Serializable.
3. **CMTElGamalCommitterCore:**

The commit function had a call to encryptWithGivenRandomValue function of ElGamal encryption scheme. This function has been deleted since there is a new function that does the same thing – encrypt that accept a random value. Instead, there is a call to the new encrypt function.

1. **CmtElGamalHashCommitter and CmtElGamalHashReceiver**:

* Changed the default hash to use OpenSSl library instead of BC.
* Deleted the throwing of the following exceptions from the default constructor: IllegalArgumentException, SecurityLevelException, InvalidDlogGroupException.

1. **CmtPedersenReceiverCore :**

commitmentMap's value has been changed from GroupElement to CmtPedersenCommitmentMessage. This is since the message is needed in verifyDecommitment function, not just the group element.

1. **CmtPedersenHashCommitter, CmtPedersenHashReceiver**, **CmtSimpleHashCommitter:**

The default hash has been changed to use OpenSSL library instead of BC.

1. **CmtSimpleHashReceiver**:

* Changed the default hash to use OpenSSL library instead of BC.
* commitmentMap's value has been changed from byte[]to CmtSimpleHashCommitmentMessage. This is since the message is needed in verifyDecommitment function, not just the group element.

1. **CmtPedersenTrapdoorCommitter:**

* The extended class has been changed – The class extends the CmtPedersenCommitter instead of CmtPedersenCommitterCore.
* The following functions have been deleted since they are implemented in the parent's class – sampleRandomCommitValue, generateCommitValue, generateBytesFromCommitValue.

1. **CmtPedersenTrapdoorReceiver:**

* The extended class has been changed – The class extends the CmtPedersenReceiver instead of CmtPedersenReceiverCore.
* The generateBytesFromCommitValue function has been deleted since it is implemented in the parent's class.

**interactiveMidProtocols::ot:**

1. **OTSemiHonestExtensionSender and OTSemiHonestExtensionReceiver:**

The native functions deleteSender and deleteReceiver has been added (accordingly). It is called from the finalize block.

**midLayer::asymmetricCrypto::digitalSignature:**

1. **CryptoPPRSAPss:**

A native function deleteRSA has been added. It is called from a finalize block.

**midLayer::asymmetricCrypto::encryption:**

1. **AsymAdditiveHomomorphicEnc**:

* An additional add function has been added. This function is identical to the existing one but get also a random value as a parameter instead of choosing it randomly.
* An additional multByConst function has been added. This function is identical to the existing one but get also a random value as a parameter instead of choosing it randomly.

1. **AsymmetricEnc**:

An additional encrypt function has been added. This function is identical to the existing one but get also a random value as a parameter instead of choosing it randomly.

All concrete encryptions schemes have been updated and added this function.

In most cases, the encrypt without the random value chooses a random BigInteger and call the other encrypt function with the chosen value.

1. **AsymMultiplicativeHomomorphicEnc:**

An additional multiply function has been added. This function is identical to the existing one but get also a random value as a parameter instead of choosing it randomly.

1. **CryptoPPRSAOaep:**

A native function deleteRSA has been added. It is called from a finalize block.

1. **DamgardJurikEnc:**

An additional reRandomize function has been added. This function is identical to the existing one but get also a random value as a parameter instead of choosing it randomly.

1. **ElGamalEnc:**

The function encryptWithGivenRandomValue has been deleted since the added encrypt function has the same perpuse. TheElGamalAbs class has been affected too since the encryptWithGivenRandomValue implementation has been changed to the new encrypt function.

1. **ScDamgardJurikEnc:**

* Fixed the s calculation in the encrypt function. s is calculated by

|p|/(|n|-1) + 1.

* Fixed the s calculations in other functions that accept the ciphertext. s is calculated by |c|/|n|.
* Fixed the message in the thrown exceptions.
* Added mod calculation on the result in the following functions: add,

multByConst, reRandomize.

**signature:**

Added OpenSSLDSASignature class.

**primitives::dlog:**

1. The option to determine which random object to use has been addded. This is done by adding constructors that accept an additional SecureRandom parameter to use in each concrete dlog class.
2. **ECF2mUtility** had the function checkCurveMembership that uses BouncyCastle functions. This was moved to **BcDlogECF2m** class since it is the only class that needs it.
3. ECF2mPointBc – When creating a point with given coordinates, there is a (optional) check that they are correct. The check is done via the curve class and not the util. (due to the previous noted change) .
4. **CryptoPpDlogZpSafePrime** and **OpenSSLDlogZpSafePrime** – the encodeByteArrayToGroupElement function is now not checking the created element. There is no need to check it.
5. **ECF2mPointMiracl** and **ECFpPointMiracl** – Two unused native functions have been deleted.
6. **MiraclDlogECFp**:: isMember – instead of using the utility class in order to check that the given point is on the curve, call a native function that do that.
7. **ECF2mPointOpenSSL** constructor that gets x and y coordinates– instead of checking the coordinated using the utility class, create the point without checking the coordinates and then check the created point using the isMember function of the DlogGroup.
8. **OpenSSLDlogECFp** ::encodeByteArrayToGroupElement – encode the given array using a native function instead of the utility class.
9. **OpenSSLDlogZpSafePrime** ::exponentiateWithPreComputedValues – the function implementation has been added. It calls the exponentiate function since in this dlog class is faster.

## Structural changes

* **Migration to git**
  + Migrated the repository to git, and moved the project to github in order to make it easier to contribute to scapi.
  + Took out the external libraries scapi is dependent on from the main codebase, and moved them to git submodules under the lib directory.
* **Linux and Mac OS X support**
  + Fixed the JNI interface code so it can support also unix based OSs.
  + Added a makefile that allows an automated compilation and installation of scapi.
* **Added shell scripts to compile and use scapi code**
  + The javac and java commands on scapi code must define the classpath and shared libraries correctly, and this results in complex commands. We added instead the **scapic** and **scapi** commands (currently only on unix-based operating systems) that allows compilation and execution of scapi code.
* **Directory structure**
  + Changed the main directory structure of scapi to better match open source “standards” and be more accessible.
  + **assets** – contains precompiled assets
  + **build** – used by the makefile for the build of scapi java and jni code
  + **doc –** houses the new sphinx based docs, but also the old docs and the javadoc docs.
  + **lib** – all the external libraries scapi is depended on
  + **scripts** – houses the scapi and scapic shell scripts.
  + **src –** the scapi source code:
    - **under src/java/ -** scapi source code
    - **under src/jni/** - scapi jni interfaces to external libraries source code

## New online documentation based on Sphinx

* Created sphinx based documentation and uploaded to readthedocs.org
* They can be found on scapi.readthedocs.org
* The intention is that the documents will keep evolving by the scapi development team and by scapi users.

## Other Notes

None.

# Version 2.2.0

## Release Date

April 8, 2014

## New Features and Functionality

**1. OpenSSL Wrapper**

SCAPI is now a wrapper for OpenSSL library. Wrappers for the following algorithm have been added:

Primitives' layer:

* Dlog Zp\*
* Dlog EC
* AES
* TripleDES
* Hmac
* Hash
* RC4
* RSA

Mid layer:

* CBC encryption (using AES or Triple DES)
* CTR encryption (using AES)

The wrapper classes can be found in the relevant packages. For example, a wrapper for OpenSSL's SHA-1 can be found in \JavaSrc\edu\biu\scapi\primitives\hash\openSSL package.

## Bug Fixes, Changes and Added Features to Existing Packages (By Package)

**Circuits::garbledCircuit**

MinimizeAESSetKeyRowReductionGarbledBooleanCircuitUtil and MinimizeAESSetKeyRowReductionGate have been changed from public to package private.

**Paddings**

A check has been added to the pad function in BitPadding and PKCS7Padding classes. We now check that the given padSize is positive. In any other case, an IllegalArgumentException is thrown.

**Primitives::dlog::miracle**

1. The unused ECF2mUtility class member has been deleted from ECF2mPointMiracl class.
2. The unused ECFpUtility class member has been deleted from ECFpPointMiracl class.

**Primitives::prf**

The unused key class member has been deleted from BcHMAC class.

**Primitives::trapdoorPermutation**

The RSAElement class has been changed from package private to public. This is because the OpenSSL implementation of RSA should use this class although it is not in the same package.

**C++Src::Jni**

1. AESPermutation:: optimizedCompute – Instead of copying each block and send the new block to the compute function, now we send the pointer to the next block.
2. CollisionResistantHash:: createHash – The JNI string is now deleted in the end of the function.
3. CollisionResistantHash::deleteHash – The implementation was added.

**C++Src::JniMiracl**

There were some prints that have been deleted.

## Other Notes

Notice that a new dll file for OpenSSL library is now provided.

# Version 2.1.0

## Release Date

March 6, 2014

## New Features and Functionality

**1. OTExtension**:

Semi Honest OTExtension protocol implementation has been added.

The semi-honest OT extension implemented is a SCAPI wrapper of the implementation by Michael Zohner from the paper:  
"G. Asharov, Y. Lindell, T. Schneier and M. Zohner. More Efficient Oblivious Transfer and Extensions for Faster Secure Computation. ACM CCS 2013". See <http://eprint.iacr.org/2013/552.pdf> for more details.  
Currently we support only 32-bit architecture; 64 bit will be supported in the future.

**2. Circuit**:

The ability of using Row Reduction technique in Garbled Circuit has been added.

## Bug Fixes, Changes and Added Features to Existing Packages (By Package)

**Circuits:** This package has major changes due to changes in the circuits' design.

The new concepts in the Garbled Circuit are:

* There are two main circuit **interfaces**:

GarbledBooleanCircuit is used for a circuit that is being garbled with a given or a sampled keys using a SecureRandom object.

GarbledBooleanCircuitSeedGeneration is used for a circuit that is being garbled with the given Prg and seed.

* For each interface, there is **one implementing class**. This class receives an **input object** that indicates the specific type of circuit to use (for example, FreeXOR) and all the underlying objects for this specific circuit type.
* During construction, the circuit uses the given input object to build a **utility class** that implements the functionalities regarding the specific circuit.
* When there is a call for one of the circuit functions, either the circuit implements it by itself (if this is a common functionality between all circuits' types) or delegates it to the underlying utility class.
* The garbling of the circuit is done by sampling two keys for each wire, then creating a garbled table for each gate.

The sampling is done in the garble function. There are three possibilities:

1. Create keys for all wires.
2. Accept keys for the input wires and sample the others.
3. Accept keys for the output wires and sample the others.

* In case the circuit should be sent to the other party, all that should be sent is the garbled circuit and the translation table.

The parties should decide ahead of time which underlying objects they both should use. Then, party one creates a circuit, samples keys and sends the garbled table and the translation table to party two. Party two creates an empty circuit with the same underlying objects as party one, receives the tables from p1 and sets them. As a result, p2 gets a circuit that is ready to be used.

* We added the row reduction circuit feature. There are utility classes that use this technique. In order to use row reduction, one should specify in the input object which KDF object to use.

There is one issue that should be taken into account when using the row reduction technique: There is a problem to sample keys from given output keys. That is, if the user wants to use the row reduction technique and to give the garble function the keys of the output wires, then he should mention it in the input object of the circuit. In addition, the user should add a gate in the end of the circuit that has the 01 truth table (XOR with zero).

To be more specific, the changes in the code are as follows:

**Circuits::circuit**

1. BooleanCircuit and Gates are no longer implementing the Serializable interface. It makes no sense that p1 sends the gates and the circuit, when all should be sent is the garbled table and the translation table; p2 can reconstruct the circuit from that. This way the message that is sent is much smaller.
2. Gate::verify(…) has been renamed to "equals" and overrides the naïve equals implementation.
3. The value data type of the wire has been changed from int to byte. It can get only 0 or 1 and byte is the smallest data type (in java) that can hold it.

**Circuits::encryption**

1. All encryption schemes are no longer implementing the Serializable interface. This should not be sent between parties. All parties should agree ahead of time on the concrete encryption scheme to use, and they all create a circuit with that object.

As a result, we deleted the readObject function in classes that override it.

1. AES128MultiKeyEncryption: the unused SecureRandom object has been deleted.
2. AES128MultiKeyEncryption: a constructor that accepts an AES object has been added.
3. AESFixedKeyMultiKeyEncryption: the "throws" declarations have been deleted from the constructor.
4. HashingMultiKeyEncryption: CryptographicHash and SecureRandom arguments have been added to the constructor.
5. The visibility of all MultiKeyEncryptionScheme's functions has been changed from package private to public.
6. MultiSecretKey: renamed numberOfKeys() to getNumberOfKeys().

**Circuits::garbledCircuit**

The changes were systemic, all classes have been changed.

As mentioned above, there are some hierarchies in the circuit's files. The main hierarchies are:

* Circuit's interfaces and concrete classes

1. GarbledBooleanSubCircuit has been deleted.
2. GarbledBooleanCircuit is the main circuit interface. Contains all circuit's functionalities:

* Garble
* Set Inputs
* Compute
* Verify
* Translate

1. The GarbledBooleanCircuitSeedGeneration interface has been added, for circuits that should generate keys using Prg and seed.
2. The GarbledBooleanCircuitImp and the GarbledBooleanCircuitSeedGenerationImp concrete classes have been added.

* Utility classes

1. The CircuitTypeUtil interface has been added. It contains all functionalities that differ in each concrete circuit:

* createGates
* garble
* getHashedTables

1. The FreeXORGarbledBooleanCircuitUtil class has been added. The implementation is based on the FreeXORGarbledBooleanSubCircuit class that has been deleted.
2. The FreeXORRowReductionGarbledBooleanCircuitUtil class has been added. It implements the FreeXOR circuit using the row reduction technique.
3. The MinimizeAESSetKeyGarbledBooleanCircuitUtil class has been added. The implementation is based on the MinimizeAESSetKeyGarbledBooleanSubCircuit class that has been deleted.
4. The MinimizeAESSetKeyRowReductionGarbledBooleanCircuitUtil class has been added. It implements the MinimizeAESSetKey circuit using the row reduction technique.
5. The StandardGarbledBooleanCircuitUtil class has been added. The implementation is based on the StandardGarbledBooleanSubCircuitclass that has been deleted.
6. The StandardRowReductionGarbledBooleanCircuitUtil class has been added. It implements the standard circuit using the row reduction technique.

* Input classes

1. The CircuitInput interface has been added. It contains the following two functions:

* createCircuitUtil() returns the utility class related to the concrete circuit.
* getUngarbledCircuit() returns the BooleanCircuit that the garbled circuit garbles.

1. The following concrete input classes have been added:

* FreeXORCircuitInput
* MinimizeAESSetKeyCircuitInput
* StandardCircuitInput
* Gates classes

1. The implementation of Serializable in GarbledGate has been deleted. Gates should not be sent but should be created in both parties.
2. The throwing of KeyNotSetException and TweakNotSetException in GarbledGate::compute() and GarbledGate::verify() has been deleted.
3. The AbstractGarbledGate has been deleted. As a result, all concrete gates contain the fieldsthat used to be in the abstract class – inputWireLabels, outputWireLabels, gateNumber.
4. The visibility of gates has been changed from public to package private since they are built by the circuit utility and the user should not be aware of them.
5. FreeXORNOTGate: extends FreeXORGate instead of AbstractGarbledGate.
6. StandardGarbledGate: added a garbledTablesHolder field. This is a pointer to all garbled tables of the circuit. The current gate should add its garbled table to the right location according to the gate number.
7. StandardGarbledGate: the reference to garbled circuit has been deleted. There is no need in this since we have the reference to the tables directly.
8. StandardGarbledGate: the constructor's arguments list has been changed – instead of getting the circuit and extract from it the encryption scheme and the garbled tables array, now it gets the encryption scheme and the garbled tables array as arguments. In addition, there is no longer use in signalBits since they can be recovered from the wires' keys.
9. StandardGarbledGate: the function createGarbledTables has been added. This function is called from the constructor and from the circuit in case there is a need to create the garbled tables after the gate has already been created.

This function accepts both keys of all wires and calculates the garbled tables of this gate.

1. MinimizeAESSetKeyGarbledGate: extends StandardGarbledGate instead of AbstractGarbledGate. The above issues regarding StandardGarbledGate are true also here.

* GarbledWire

1. The Serializable implementation has been deleted.
2. The label class member has been deleted. The label of each wire is kept in the circuit instead of the wire.
3. The signalBit data type has been changed from int to byte.

**InteractiveMidProtocols::ot**

1. The class OTOnByteArrayROutput now implements also OTBatchROutput.
2. The OTRExtensionInput and OTSExtensionInput classes have been deleted. Equivalent files have been added in the new OTExtension implementation.
3. otBatch package is moved from interactiveMidProtocols package to the interactiveMidProtocols::ot package. All references have been updated.
4. The interface OTBatchSOutput has been added. This interface will be implemented by the OT Batch protocol which has an output for the sender.
5. The return value of OTBatchSender::transfer(…) function has been changed from void to OTBatchSOutput. This is because there are cases that the sender has an output (for example, in correlated OTExtension). Implementation that has no output will return null.
6. According to the above new return value, OTSemiHonestDDHBatchSenderAbs returns null in the transfer function because it has no output.

**InteractiveMidProtocols::ot::OTBatch**

This package used to be under interactiveMidProtocols. We decided that from now on, since it is a kind of OT protocol, it is more convenient to put it under the OT package.

**Primitives::Dlog**

The creation of a file called "EcFpPointMiracleResults.csv" in the constructor of MiraclAdapterDlogEC has been deleted. This file is not in use anymore and should not be created.

**C++Src::JniOtExtension**

The OT Extension was not published in the previous version and this directory that contains the code of the OTExtension dll was not in use.

1. The option to specify the number of treads to use in the execution has been added.
2. The option to specify which DlogGroup to use in the execution has been added.
3. The option to specify which OT Extension version to use in the execution (General, Correlated or Random) has been added.

## Other Notes

None.

# Version 2.0.0

## Release Date

November 14, 2013

## New Features and Functionality

1. **RandomOracle**:
2. Added RandomOracle interface in primitives' package.
3. Added two random oracle concrete implementations:

* HKDFBasedRO
* HashBasedRO

1. **InteractiveMidProtocols layer**: This is SCAPI's third layer which contains cryptographic protocols which are popular building blocks. This layer contains implementations of the following protocols:
2. *Sigma protocols:*

* Schnorrs Sigma-Protocol for DLOG (SigmaDlog)
* Sigma-Protocol for Diffie-Hellman Tuples (SigmaDH)
* Sigma-Protocol for Extended Diffie-Hellman Tuples (SigmaDHExtended)
* Sigma Protocol for Pedersen Commitment Knowledge (SigmaPedersenCmtKnowledge)
* Sigma-Protocol that a Pedersen-Committed Value is x (SigmaPedersenCommittedValue)
* Sigma Protocol for El Gamal Commitment Knowledge (SigmaElGamalCmtKnowledge)
* Sigma Protocol that an ElGamal-Committed Value is x (SigmaElGamalCommittedValue)
* Sigma Protocol of ElGamal Private Key (SigmaElGamalPrivateKey)
* Sigma Protocol that ElGamal-Encrypted Value is x (SigmaElGamalEncryptedValue)
* Sigma Protocol that Cramer-Shoup-Encrypted Value is x (SigmaCramerShoupEncryptedValue)
* Sigma Protocols that a Damgard-Jurik Encrypted Value is 0 (SigmaDamgardJurikEncryptedZero)
* Sigma Protocols that a Damgard-Jurik Encrypted Value is x (SigmaDamgardJurikEncryptedValue)
* Sigma Protocols that 3 Damgard-Jurik Ciphertexts are a Product (SigmaDamgardJurikProduct)

1. *Sigma protocol transformations:*

* Sigma Protocol AND of multiple statements (SigmaAND)
* Sigma Protocol OR of 2 statements (SigmaORTwo)
* Sigma Protocol OR of multiple statements (SigmaORMultiple)

1. *Zero Knowledge (ZK), automatic transformations from Sigma protocols to zero-knowledge protocols:*

* Zero-Knowledge from any Sigma Protocol (ZKFromSigma)
* Zero-Knowledge Proof of Knowledge from any Sigma Protocol (ZKPOKFromSigmaCmtPedersen). This protocol can use any perfectly hiding trapdoor (equivocal) commitment scheme. Currently this is not up to the user and we always use PedersenTrapdoor commitment scheme.
* ZKPOK from any Sigma-Protocol ROM Fiat-Shamir (ZKPOKFiatShamirFromSigma)

1. *Oblivious Transfer (OT), includes preprocessing in constructor which gives optimization for many executions, when existing:*

* Semi-Honest OT (OTSemiHonestDDH)
* Private OT Naor-Pinkas (OTPrivacyOnlyDDH)
* Oblivious Transfer with One-Sided Simulation (OTOneSidedSimulationDDH)
* Oblivious Transfer with Full Simulation (OTFullSimulationDDH)
* Oblivious Transfer with Full Simulation ROM (OTFullSimulationROMDDH)
* Oblivious Transfer with UC Security DDH (OTUCDDH)

1. *OT Batch, optimized OT for many executions, where such optimizations exist beyonda single preprocessing:*

* Semi-Honest Batch OT (OTBatchSemiHonest)

1. *Commitment*

* Pedersen Commitment (CmtPedersen)
* Pedersen-Hash Commitment (CmtPedersenHash)
* Pedersen Commitment with Trapdoor for Equivocation (CmtPedersenTrapdoor)
* ElGamal Commitment (CmtElGamal)
* ElGamal-Hash Commitment (CmtElGamalHash)
* Hash-Based Basic Commitment (CmtSimpleHash)
* Equivocal Commitments (CmtEquivocal)

1. *Coin Tossing*

* Coin-Tossing of a Single Bit Blum (CTBlum)
* Coin-Tossing of a String (CTString)
* Semi-Simulatable Coin-Tossing of a String (CTSemiSimulatable)

## Bug Fixes, Changes and Added Features to Existing Packages (By Package)

**Properties files:**

1. Added properties files for RandomOracle:

* RandomOracle.properties
* RandomOracleDefault.properties

1. Added SCAPI's defaults properties file, SCAPIDefaultConfig.properties.

**C++ Source::JNI**

1. Added the function deleteAES() in AES implementation using Crypto++ library.

**C++ Source::JNIMiracl**

1. Added AES implementation using Miracl library.
2. Added deleteMip() function that releasing the allocated memory.
3. Added a check after creating Miracl points that the point is valid. If not, release the memory and return null.

**C++ Source::JNINtl**

1. Added support for polynomial fields used by SigmaORMultiple protocol.

**Exceptions package:**

1. Added exceptions :

* CommitValueException.
* InvalidDlogGroupException.

**Generals package:**

1. Added SCAPIDefaultConfiguration.

**Encryption package:**

1. Added SecurityLevelException throws declaration in CramerShoup and ElGamal encryptions constructors in case that the given Dlog or the Hash function do not meet the required Security Level.
2. Added encryptWithGivenRandomValues() function in ElGamal encryption .
3. Added ElGamalKem implementation.
4. Added TPBlumGoldWasserEnc interface.

**Keys package:**

1. Added getP() and getQ() functions in DamgardJurikPrivateKey interface.
2. Changed the ScDamgardJurikPrivateKey constructor from

public ScDamgardJurikPrivateKey(BigInteger t, BigInteger d) to

public ScDamgardJurikPrivateKey(RSAModulus rsaMod).

1. Added implementation of getP() and getQ() function in ScDamgardJurikPrivateKey class.

**Ciphertext package:**

1. Added ElGamalCiphertextSendableData class.
2. Changed ElGamalOnByteArraySendableData and ElGamalOnGrElSendableData to implement ElGamalCiphertextSendableData instead of AsymmetricCiphertextSendableData.
3. Added equals implementation in ElGamalOnByteArrayCiphertext and ElGamalOnGroupElementCiphertext.

**Primitives package:**

1. Added CryptoPpAES::deleteAES native function that is called from the finalize block.
2. Added AES implementation using Miracl library.
3. Deprecated generateElement(BigInteger x, BigInteger y) function in elliptic curves classes and interface, use generateElement(boolean bCheckMembership, BigInteger...values) instead.
4. Deprecated generateElement (Boolean bCheckMembership, BigInteger x) in DlogZp classes and interface, use generateElement(boolean bCheckMembership, BigInteger...values) instead.
5. Deleted implementation of generateElement(boolean bCheckMembership, BigInteger...values) in DlogGroupEC. The implementation moved to each concrete class.
6. Elliptic curves' exponentiate function: In case the given exponent is negative, compute the function with its value modulus the order of the group.
7. Changed constructors of ECF2mPointBc and ECFpPointBc. Added parameter bCheckMembership indicates whether or not check if the given values are valid. Updated calls to this constructor.

**SecurityLevel package:**

1. Added the following security levels:
   * CommitSecLevel
   * SecureCommit
   * EquivocalCmt
   * ExtractableCmt
   * NonMalleableCmt
   * FullyTrapdoorCmt
   * PerfectlyBindingCmt
   * PerfectlyHidingCmt
   * StatisticallyHidingCmt
   * Covert
   * Malicious
   * OneSidedSimulation
   * PrivacyOnly
   * SemiHonest
   * StandAlone
   * UC

**Factories package:**

1. Fixed bug in universalHashFactory:: getObject(String algName, String provider) function.
2. Added RandomOracleFactory.

## Other Notes

None.

# Version 1.0.2.2

## Release Date

May 29, 2013

## New Features and Functionality

None.

## Bug Fixes, Changes and Added Features (By Package)

**Properties files:**

1. Deleted properties files of Mid-layer components.
2. Padding properties file: Change ScapiPKCS7 to ScapiPKCS7Padding.

**C++ source::JNI:**

1. Fixed validity check of Rabin element. Instead of checking that the element is in the right range and calculating if it has a square root, compute the Jacobi function.
2. Implement the compute function of RabinPermutation using Crypto++ math function instead of using RabinFunction::ApplyFunction() of Crypto++.
3. Implement the invert function of RabinPermutation using Crypto++ math function instead of using RabinFunction::CalculateInverse() of Crypto++.
4. Added to the creation of random Rabin element the check that the chosen element r meets the requirement GCD(r, N) = 1.

**Circuits’ package:** This package has major changes that include the following:

1. Added serialization capabilities to all elements in circuits.
2. Added support of sub circuits which resulted in the addition of classes. Now a sub circuit is the base class and the circuit inherits from the sub circuit. Most of the functionality has moved to the sub circuit; the translation table and all related functionality remained in the circuit classes.
3. Changed the garbled tables to hold byte array and not ByteArraySymCiphertext for efficiency reasons.
4. Moved the garbled tables from the gates to the garbled circuit and adding reference to the garbled circuit in the gates. This way this table can be sent separately and not the entire garbled circuit.

To be more specific, the changes in the code are:

1. Gate’s constructors turn to public instead of package private.
2. Gate::Verify function turns to public instead of package private.
3. Wire::getValue function turns to public instead of package private.
4. Changed the input in BooleanCircuit to hold input for each party separately instead of holding the inputs for all parties together. This change caused the following changes:
5. Deleted the constructor

BooleanCircuit(Gate[] gates, int[] outputWireLabels, Map<Integer, Wire> setInputWires)

1. Changed the input in BooleanCircuit. Now we get the input wire labels of each party instead of get the input of all parties.
2. Changed

setInputs(Map<Integer, Wire> presetInputWires)to

setInputs(Map<Integer, Wire> presetInputWires,int partyNumber)

and **void** setInputs(File inputWires) to **public** setInputs(File inputWires, int partyNumber).

Added parameter partyNumber and changed the implementation of the function to meet the new idea of holding the input for each party.

1. Instead of throwing RuntimeException in Compute() function in case that the input not set yet, throw NotAllInputsSetException in case that not all the parties’ inputs were set.
2. Changed int[] getInputWireLabels() to ArrayList<Integer> getInputWireLabels(int partyNumber).

Return the inputWires for the requested party only.

1. Added functions int getNumberOfInputs(int partyNumber) and int getNumberOfParties()
2. In MultiKeyEncryptionScheme interface the following functions have been changed:

• ByteArraySymCiphertext encrypt(ByteArrayPlaintext plaintext) to byte[] encrypt(byte[] plaintext).

• ByteArrayPlaintext decrypt(ByteArraySymCiphertext ciphertext) to byte[] decrypt(byte[] ciphertext).

1. Added MultiKeyEncryptionScheme::getCipherSize()
2. AES128MultiKeyEncryption and AESFixedKeyMultiKeyEncryption that implements MultiKeyEncryptionScheme interface changed according to the changes in the interface that described before (12, 13).
3. The constructor of HashingMultiKeyEncryption changed to create CryptoPPSHA1.
4. Moved exceptions from circuits::encryption package to exceptions package. The moved exception are:
   * CiphertextTooLongException
   * InvalidKeySizeException
   * KeyNotSetException
   * PlaintextTooLongException
   * TweakNotSetException
   * InputNotSetException
5. Deleted AbstractGarbledBooleanCircuit. Functionality moved to file AbstractGarbledBooleanSubCircuit.
6. Added classes:
   * AbstractGarbledBooleanSubCircuit
   * CircuitUtil
   * FreeXORGarbledBooleanSubCircuit
   * FreeXORGateSlim
   * FreeXORNOTGate
   * GarbledBooleanSubCircuit
   * MinimizeAESSetKeyGarbledBooleanSubCircuit
   * StandardGarbledBooleanSubCircuit
7. Update all the current circuits implementations to the new interface.

**Comm package:**

1. Added utility class LoadParties. This class is not a must in Scapi but is definitely helpful to read a list of parties from a file.
2. Deprecated the constructor public Party(String name, InetAddress ipAdress, Key key, int port, Role role)
3. Added the constructor public Party(InetAddress ipAdress, int port)
4. Added communication example package.

**Exception package:**

1. Added exceptions:
   * CannotBeGarbledException
   * CheatAttemptException
   * CircuitFileFormatException
   * InvalidChannelException
   * InvalidInputException
   * NotAllInputsSetException
   * NoSuchPartyException
2. Deprecated InvalidChannel.

**Encryption:**

1. Deprecated functions generateCiphertext, generatePublicKey, generatePrivateKey, generateKeyPair. Instead, added functions reconstructCiphertext, reconstructPublicKey, resonstructPrivateKey.

In general the reason for deprecating generate[Something](Serializable) and adding instead reconstruct[Something](Serializable) is that the name generate... suggests ”from sratch” whereas reconstruct... suggests that the object already existed and we are only reproducing it.

1. Added default constructor to DJKeyGenParameterSpec.
2. Added ScElGamalKEM implementation.
3. Added TPBlumGoldWasserEnc interface.
4. Added constructor public ScDSAPublicKey(ScDSAPublicKeySendableData data, DlogGroup dlog).
5. Added toString() implementation in ScCramerShoup keys.
6. Added toString() implementation in all ciphertext and plaintext classes.
7. Added ElGamalKEMCiphertext.
8. Deleted constructor public ScEncryptThenMac(String encName, String macName). This is because we currently do not support factories to the mid-layer. In future implementation, we may add this feature and this constructor will be back.

**Primitives’ layer:**

1. Added thorough explanation in Javadoc about encode/decode byte[] to GroupElement, mapGroupElementToByteArray and when to use what.
2. Deprecated function generateElement. Instead, added function reconstructElement.
3. Removed outdated file PseudorandomGeneratorAbs.
4. Fixed RabinPermutation::isElement.
5. Fixed RabinPermutation::compute, checkValidity, generation of random element. Most changes in C++ code, see “C++ source::JNI” above.
6. Deprecated function generateTPElement. Instead, added function reconstructTPElement.

**Factories:** For the moment canceled the mid-Layer’s factories feature.

## Other Notes

None.

# Version 1.0.2.1

## Release Date

January 20, 2013

## Additional Features and Functionality

None.

## Bug Fixes and Changes

1. *HKDF:* Fixed a bug that a key derived from the same entropy source was different in subsequent calls to this function.
2. *ScPrgFromPrf:* This class doesn’t have and/or use a KDF anymore to generate the key, instead it delegates the generation to the underlying PRF, which is the right thing to do.
3. *BcPRG:* In generateKey check that the key size is greater than 0 and a multiple of 8.
4. *Miracl Dlog:*
5. Added function endExponentiateWithPreComputedValues which cleans up any resources used by exponentiateWithPreComputedValues for the requested base.
6. Overrode the exponentiateWithPreComputedValues function, such that if the curve being used is a Koblitz curve then the regular exponentiation function is called, else the parent’s implementation is called. (This is because this optimization is actually slower on Koblitz curves.)
7. Fixed bug found in exponentiateWithPrecomputedValues.
8. Fixed bug in encoding and decoding of binary strings to clog group elements.
9. *Communication layer:* The communication layer has been changed. The way to work is to use the CommunicationSetup class in order to obtain communication channels of type PlainTcpChannel. Then, for each such channel, you can decorate it as an AuthenticatedChannel and/or as an EncryptedChannel in order to add automatic authentication and/or encryption when sending messages over the channel. If you want a “secure channel” that is encrypted and authenticated, the best way to work is to decorate it as an EncryptedChannel with an auth-encryption scheme (i.e., an encryption scheme with security level AuthenticatedEnc, like ScEncryptThenMac). Examples on how to do this appear under the Examples Tab on the SCAPI webpage.

## Other Notes

SCAPI has been tested only on Windows 7 installations. However, using BouncyCastle as the provider, we know of successful installations on Linux.

# Version 1.0

## Release Date

November 6, 2012

## Additional Features and Functionality

None.

## Bug Fixes and Changes

None.

## Other Notes

This is the first beta version of SCAPI.