ANSI C Cryptographic API Profile
for SHA-3 Candidate Algorithm Submissions

Revision 4: February 5, 2008

1. Overview

This document specifies the ANSI C interface profile for implementations of SHA-3 candidate algorithms. C implementations shall support the syntax and parameterization of the interface profile messages as described in this API. The API consists of a few data definitions and one function to compute hashes. The function specified in this API has return values listed that are largely used to supply error codes in the event of incomplete execution of the routine. The error values listed are not meant to be an exhaustive list. If additional error codes are useful for your implementation, please provide them.

2. Data Definitions

The following typedef is used to specify the arrays that will hold the data to be hashed and the resulting hash value.

```
typedef unsigned char BitSequence;
```

The byte length, n, of a BitSequence will be $n = \lceil bitlen/8 \rceil$, e.g., an 8-bit message will require 1 BitSequence element and a 13-bit message will require 2 BitSequence elements. BitSequence arrays will be indexed from 0 to n-1. Sequences of bits are enumerated from 0 to ($bitlen$-1). The i^{th} bit of the sequence will be stored in array element $\lfloor i/8 \rfloor$. Within a BitSequence array element, the bits are indexed 0 to 7 with bit 0 being the Most Significant Bit (MSB), i.e., the bit with the largest numerical value. Therefore, the i^{th} bit of the BitSequence will be found in the i % 8 bit position of the $\lfloor i/8 \rfloor$ bitSequence element.

The following typedef is used to provide the data length of the message to be hashed. It should be set to the largest integral data type that the target platform and compiler can understand. Preferably this will be an unsigned 64-bit integer. If the target platform and compiler cannot handle a 64-bit data type, use a 32-bit unsigned data type instead.

```
typedef unsigned long long DataLength;    // a typical 64-bit value
```

The following enumeration is to provide return values for the API Hash function. Additional return values may be added. These values shall be documented.

```
typedef enum HashReturn { SUCCESS = 0, FAIL = 1, BAD_HASHLEN = 2 };
```

3. Hash

The ANSI C programming interface uses a function called ***`Hash()`*** to process data using the candidate algorithm and to return the resulting hash value. The ***`Hash()`*** function is called with

a pointer to the *data* to be processed, the length of the data to be processed (*databitlen*), a pointer to the storage for the resulting hash value (*hashval*), and a length of the desired hash value (*hashbitlen*).

- **Hash()**

```
HashReturn Hash(const BitSequence *data, DataLength databitlen,
                BitSequence *hashval, int hashbitlen);
```

Hash the supplied data and provide the resulting hash value. Set return code as appropriate.

Parameters:
data: the data to be hashed
databitlen: the length, in bits, of the data to be hashed
hashval: the resulting hash value of the provided data
hashbitlen: the length in bits of the desired hash value

Returns:

```
SUCCESS - on success
FAIL – arbitrary failure
BAD_HASHLEN – unknown hashlen requested
...
```