

CompactSHA – Secure Hash Standard

SHA-256 C++ class library suitable for embedded systems

Reference Manual

Revision 1.1

25th August 2009

COPYRIGHT © 2005 - 2009 UBISYS TECHNOLOGIES GMBH

ubisys

1 Introduction

The Secure Hash Algorithms (SHA) are a collection of different cryptographic hash functions designed to generate a secure unique hash key for any dataset or message. The algorithms were designed by the National Security Agency (NSA) and published by the U. S. National Institute of Standards and Technology (NIST) as a U. S. Federal Information Processing Standard [1], [2].

SHA algorithms are used in many applications to verify data integrity, especially in applications requiring a cryptographically strong digest, i.e. one that cannot easily be duplicated.

The SHA family of algorithms knows three different algorithms: SHA-0, SHA-1 and SHA-2, which differ in details of the algorithm and digest size.

CompactSHA implements the SHA-256 algorithm, which is part of the SHA-2 family.

1.1 SHA-256

SHA-256 uses a digest size of 256 bits. The algorithm calculates the hash value by segmenting the data into blocks of 512 bits and processes the data iteratively with six logical functions and 64 constants. The start hash value is represented by the first 32 bits of the decimal places of the square root of the first eight prime numbers (2 to 19).

Every block needs 64 rounds of the SHA-2 family compression function.

1.2 Security

Different analysis has shown that attacks on the SHA-0 and SHA-1 algorithms are possible and thus, possibly defeating a cryptographic signature based on their usage.

The U. S. National Institute of Standards and Technology (NIST) advices to use the SHA-2 algorithms for new applications.

For SHA-2, an attack is not yet known. Currently, as of 2009, the SHA-2 algorithms are considered to be secure.

2 Implementation

The CompactSHA implementation is optimized for 32 bit embedded systems, with little memory and computing power.

The algorithm is encapsulated in the class `CSecureHashAlgorithm256`.

Two modes of application are supported: a one-step hash operation on a given buffer and an interface consisting of three functions to allow hash calculations on incoming data streams, without the necessity to keep the whole data in one buffer.

In both cases, the calculated hash value is available in the member variable `m_abDigest`.

2.1 Usage example: single buffer

For the one-step calculation, simply call the `Calculate()` function and pass the address and size of the buffer:

```
CSecureHashAlgorithm256 sha;
unsigned char abBuffer[1024];
// Implement: Fill buffer ...
sha.Calculate(abBuffer, sizeof(abBuffer));
// Read the hash from the member m_abDigest
```

2.2 Usage example: stream processing

To calculate a hash value of an incoming stream, i.e. only small chunks of data are available at a given time, use the three-method interface, consisting of `Start()`, `Update()` and `Finish()`.

```
CSecureHashAlgorithm256 sha;
// Initialize
Start();
while (...)
{
    // Implement: get buffer ...
    Update(pBuffer, cbBuffer);
}
Finish()
// Read the hash value from m_abDigest
```

3 Class Reference

A single class, CSecureHashAlgorithm256 is defined.

3.1 CSecureHashAlgorithm256

3.1.1 Declaration

```
class CSecureHashAlgorithm256;
```

3.1.2 Attributes

```
// Contains the calculated digest  
unsigned char m_abDigest[32];
```

3.1.3 Member functions

```
// Initializes the SHA algorithm. Must be called once before calling  
// Update()  
void Start();  
  
// Must be called after all data has been hashed. The m_abDigest  
// member is meaningful only after a call to Finish()  
void Update(const unsigned char *pbInput, unsigned int nLength)  
  
// Finalize computation of the hash value. After calling this function, the  
// hash value should be read from the member array m_abDigest  
void Finish();  
  
// one-step interface:  
// Performs the sequence of Start(), Update(pData, nLength) and Finish  
// in a single call.  
void Calculate(const void *pData, unsigned int nLength);
```

4 References

- [1] FIPS PUB 180-1: Federal Information Processing Standards Publication 180-1: "SECURE HASH STANDARD", 17th April 1995
- [2] FIPS PUB 180-2: Federal Information Processing Standards Publication 180-2: "SECURE HASH STANDARD", 1st August 2002

5 Revision History

Revision	Date	Changes
1.0	15 th June 2005	Initial Version
1.1	25 th August 2009	Add notes on security status of SHA-0 and SHA-1.

6 License

CompactSHA ("the software") remains the sole property of ubisys technologies GmbH, Düsseldorf, Germany ("ubisys"). A limited license is granted to the licensee including the right to distribute the software in binary form as part of licensee's products. The source code must not be disclosed to third parties.

The software is provided "as is", without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose and non-infringement.

In no event shall ubisys be liable for any claim, damage or other liability, whether in an action of contract, tort or otherwise, arising from, out of or in connection with the software or the use or other dealings in the software.

7 Contact

UBISYS TECHNOLOGIES GMBH
HARDWARE AND SOFTWARE DESIGN
ENGINEERING AND CONSULTING
AM WEHRHAHN 45
40211 DÜSSELDORF
GERMANY
T: +49. 211. 54 21 55 00
F: +49. 211. 54 21 55 99
www.ubisys.de
info@ubisys.de