# Proposal for cryptographic functions inside C++ standard library

Markus Mayer

January 30, 2014

| | |
|---|---|
| **Document number:** | Nnnnn=yy-nnnn |
| **Date:** | 2014-01-29 |
| **Project:** | Programming Language C++, Library Working Group |
| **Reply-to:** | Markus Mayer <lotharlutz at gmx dot de> |

## Contents

## 1 Introduction

## 2 Motivation and Scope

## 3 Impact On the Standard

## 4 Design Decisions

### 4.1 Representation of bytes and byte ranges

As most cryptographical functions work on single bytes or an ranges of bytes, it's an essential question of how to represent these types within the API. For the rest of this document a byte is defined by 8 bits. This proposal doesn't not apply to architectures lacking an 8 bit type.

#### Single byte

A single byte is represented as an `unsigned char`. This decision was made to avoid misinterpretation as a Character.

#### Byte range

Fixed sized byte ranges are represented as `std::array<unsigned char, SIZE>`. This allows to specify ranges lager then the available int types and avoids a misinterpretation as an integral number (including endianess).

Variable length ranges are represented either as a pair of iterators, or as a pointer (`void*`) to and the size (`std::size_t`) of a memory region. The iterator variant allows a flexible way to specify ranges without exposing to much constrains (e.g. non continuous storage in memory). The second variant avoids unnecessary casting if you want to pass an type greater than `unsigned char` (e.g. `fn(&range, sizeof(range))`).

### 4.2 Hash functions

A hash function is any algorithm that maps data of arbitrary length to data of a fixed length. Each hash function implementation must fulfill the following synopsis which is based on boost::crc. OpenSSL and libgcrypt uses an equal interface (but in C). Crypto++'s interface is considered to complex (inheritance depth > 6).

```
class hash_function
{
public:
        typedef std::array<unsigned char, ALGORITHM_DEFINED> result_type;

        void process_bytes(const void* buffer, std::size_t byte_count);
```

```
    template<class IterType>
    void process_bytes(IterType bytes_begin, IterType bytes_end);

    void operator()(const void* buffer, std::size_t byte_count);

    template<class IterType>
    void operator()(IterType bytes_begin, IterType bytes_end);

    void reset();

    result_type digest() const;

};
```

**Alternatives**

- `result_type` could also be called `value_type`. `value_type` is used by boost::crc, but `result_type` was selected to correspond with std::function.
- `process_byte[s](...)` could also be called `update(...)` (OpenSSL, Crypto++) or `write, putc` (libgcrypt).
- `reset()` is called `restart()` by Crypto++
- Alternative names for `digest()` are `hash_value()` or `checksum()`.

## 5 Technical Specifications

## 6 Acknowledgments