Crypto++: Past Validations and Future Directions

Jeffrey Walton
Security Consultant
Introductions
About Me

- Jeffrey Walton
  - Security Architect
  - Mobile Security Engineer
  - Library Maintainer

- Verticals
  - US DoD
  - US Federal
  - US Financial
About Crypto++

- C++ class library
- Originally written by Wei Dai
- Released in June 1995
- General purpose crypto library
- Given to community 2015
- https://www.cryptopp.com
About Crypto++ (continued)

- C++03 through C++17
  - Heavy use of templates
  - F-Bound quantification
  - Static polymorphism
  - Acquired taste
FIPS Validations
Validations

- **Crypto++ 5.0.4**
  - October 2002
  - Certificate 343

- **Crypto++ 5.2.3**
  - September 2005
  - Certificate 562

- **Crypto++ 5.3.0**
  - September 2006
  - Certificate 819

- **Historical Validation list**
  - Sometime around 2015 or 2016
Sponsor and Lab

■ Benefactor
  ‣ Groove Networks
  ‣ Productivity & Collaboration
  ‣ Purchased by Microsoft

■ Testing Lab
  ‣ Cygnacom
FIPS Module
Windows DLL

- NIST approved algorithms
  - RNG, AES, SHA, MAC, RSA, DH
- One set of `#define`'s
  - DLL_EXPORT (et al)
- Artifacts
  - cryptopp.lib
  - cryptopp.dll
Windows Lib

- Non-FIPS routines
  - Non-approved algorithms
  - Encoders, Decoders
  - Sources, Filters, Sinks
- Another set of #define's
  - DLL_IMPORT (et al)
- Artifact
  - cryptlib.lib
Windows App

- Multiple libraries
  - Needed both cryptopp.lib and cryptlib.lib
  - cryptopp.lib import library
  - cryptlib.lib static archive

- Lots of confusion
  - Nightmare for users
  - Users think it’s a regular "Crypto++ DLL"
  - Three wiki pages covering them
App View

Application

Lib: cryptlib.lib

Import: cryptopp.lib
DLL: cryptopp.dll
Explicit Instantiations

■ Pure C++ library
  ▸ Everything in a header
  ▸ Instantiated at call site

■ FIPS DLL
  ▸ Header (*.h) + source (*.cpp)
  ▸ Source explicitly instantiates object
  ▸ \texttt{template class X \langle Y\rangle}
Future Directions
Improve Design (?)

- Add C interface
  - Avoid C++ ABI instability
- Flatten API
  - FIPS_rng_generate_block
  - FIPS_aes_gcm_128
  - Init-Update-Final pattern
Improve Design (?) (continued)

- C++ classes to wrap flattened API
  - OK to provide to users ...
  - ... but we don’t export the classes

- Add Engine-like interface
  - Already have object registry
  - Add provider strings
  - Additional namespace?
Another Validation

- Probably not
  - Need sponsor or benefactor
  - Need time and energy

- Already probed DHS and NSF
  - Looking for projects and grants
Native Services

- Use native services
  - Crypto++ wrapper for OS or external library
  - We already do this on a limited basis
    - RNGs, Pipes, Sockets, Timers
- Apple
  - CommonCrypto
- Linux/Mozilla
  - Network Services (NSS)
- Windows
  - CAPI or Bcrypt
OpenSSL Integration

- Utilize FIPS Object Module
  - Use it like external service
  - `libcryptopp.a + fipscanister.o`
  - Use `fipsld++` to build `fips_premain.c` and link against `fipscanister.o`
  - `fipsld++` available from OpenSSL wiki

- Platform not validated?
  - Crypto++ becomes an OpenSSL customer
  - $5K to $10K private label validation
Init struct/OpenSSL

```c
#include <sstream>
#include <cstddef>
#include <cstdlib>
#include <cstring>
#include "usr/local/ssl/fips-2.0/include/openssl/opensslconf.h"
#include "usr/local/ssl/fips-2.0/include/openssl/e_os2.h"
#include "usr/local/ssl/fips-2.0/include/openssl/fipsym.h"
#include "usr/local/ssl/fips-2.0/include/openssl/evp.h"
#include "usr/local/ssl/fips-2.0/include/openssl/fips.h"

struct OsslFipsInit
{
    OsslFipsInit()
    {
        const char password[] = "etaonrishdlcupfm";
        if (FIPS_module_mode_set(1, password) != 1)
        {
            throw std::runtime_error("FIPS_module_mode_set failed.");
        }
        std::cout << "FIPS_module_mode_set succeeded." << std::endl;
    }
};
```
PIMPL class/OpenSSL

```cpp
//////////////////////////////////////////////////////////////////////////// The OpenSSL implementation //////////////////////////////////////////////////////////////////////////

using EVP_CIPHER_CTX_ptr = std::unique_ptr<EVP_CIPHER_CTX, decltype(&::EVP_CIPHER_CTX_free)>;

struct OsslAesImpl
{
  OsslAesImpl() : m_ctx(NULLPTR, ::EVP_CIPHER_CTX_free) {}

  void Init(const byte* key, size_t ksize, const byte* iv, size_t vsize, CipherDir direction)
  {
    m_ctx = EVP_CIPHER_CTX_ptr(EVP_CIPHER_CTX_new(), ::EVP_CIPHER_CTX_free);
    if (direction == ENCRYPTION)
    {
      int rc = M_EVP_EncryptInit_ex(m_ctx.get(), EVP_aes_256_cbc(), NULL, key, iv);
      if (rc != 1)
        throw std::runtime_error("EVP_EncryptInit_ex failed");
    }
    else
    {
      int rc = M_EVP_DecryptInit_ex(m_ctx.get(), EVP_aes_256_cbc(), NULL, key, iv);
      if (rc != 1)
        throw std::runtime_error("EVP_DecryptInit_ex failed");
    }
  }
```
Command Line/FOM linking

```
skylake:cryptopp-openssl$ FIPSLD_CC=g++ /usr/local/ssl/fips-2.0/bin/fipsldd++ -g3 -01 -fno-pic openssl-init.cxx openssl-aes.cxx test.cxx /usr/local/ssl/fips-2.0/lib/fipsscanister.o ./libcryptopp.a -o test.exe
skylake:cryptopp-openssl$ ./test.exe
FIPS_module_mode_set succeeded.
skylake:cryptopp-openssl$
```
Questions?

- Hopefully useful answers

- Jeffrey Walton
  - noloader@gmail.com
  - jeffrey@deltoid.com

- Credits
  - Wei Dai, who gave the world Crypto++
  - Cryptographers and researchers who make it happen