ECE 646 – Fall 2015 Term Project

Overview, comparison of open crypto libraries for application development.

By Ravi Kota

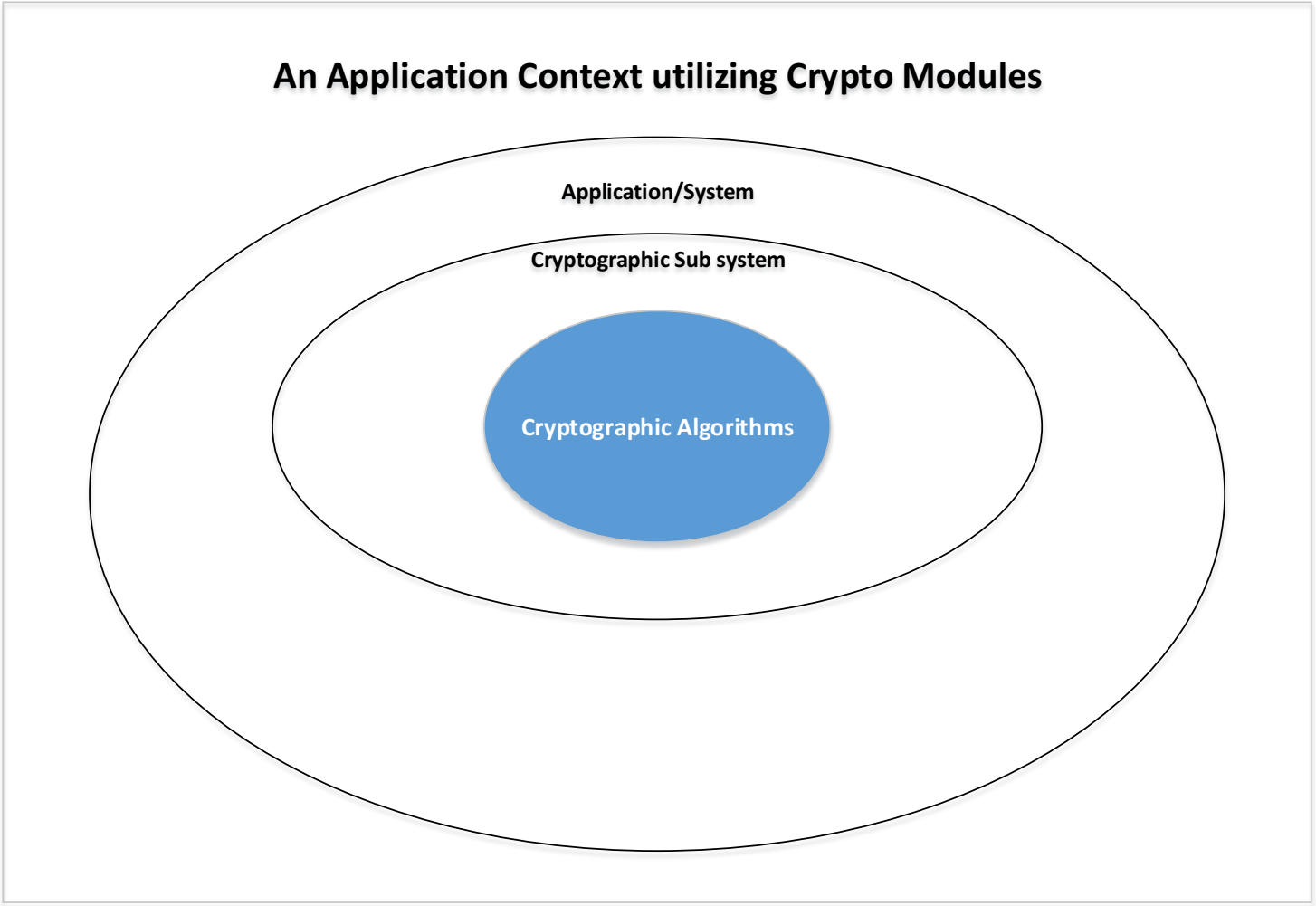
Goal

- How to determine which open source crypto library or libraries can be considered for a crypto application development?
- Reason: When enterprises build software applications, it is desirable to consider:
- Secure & correct implementation
- Compliant to standards
- Long term Supportability

Major Criteria

1. Identify common requirements of crypto applications. (List of Crypto primitives and other features).
2. Study crypto library APIs & analyse how to implement/support the common features of crypto applications.
3. Compile some performance data on open crypto libraries comparison from available papers.
4. Analyse the CVEs of the crypto libraries, how it impacts crypto applications.

Crypto Application Context



Cryptographic Requirements

- Symmetric key ciphers
  - Modes of Operation
- Asymmetric key ciphers
  - Encryption/Decryption
  - Digital Signature
  - Key Exchange
- Hash functions
- Message Authentication Codes (MAC)
- Random Number generators
- Key Generation & Management

Other Requirements

- Attack/Security Level
- Memory/Speed requirements
- Support for standards
- Optimizations (low level optimization)
- Portability
- Special instructions available on specific processors
- Ease of installation
- Documentation support

Symmetric Ciphers

Algorithm	Key Length	References
TDEA	3 key TDES	800-131Ar1, 800-67, 800-197
AES	128, 192 & 256 bits	800-131Ar1
SKIPJACK	80 bit	Recently withdrawn
Stream Ciphers	?	

Block cipher mode of operation

- Electronic Codebook (ECB): Each block of plain text is encoded independently using the same key.
- Cipher Block Chaining (CBC): The input to the encryption algorithm is the XOR of the next-block of plaintext and the preceding block of ciphertext.
- Cipher Feedback (CFB): Input is processed s bits at a time. Preceding ciphertext is used as input to the encryption algorithm to produce pseudorandom output, which is XORed with plaintext to produce next unit of ciphertext.
- Output Feedback (OFB): Similar to CFB, except that the input to the encryption algorithm is the preceding encryption output, and full blocks are used.
- Counter (CTR): Each block of plaintext is XORed with an encrypted counter. The counter is incremented for each subsequent block.
- XTS-AES: Encryption for data stored in sector-based devices.

Asymmetric Ciphers

Algorithm	Application			Refs
	ENC/DEC	Digital Signature	Key Exchange	
RSA	YES	YES	YES	
Elliptic Curve	YES	YES	YES	
Diffie-Hellman	NO	NO	YES	
DSS	NO	YES	YES	

Key sizes:Asymmetric Ciphers

Function	Algorithm & Key Size
Digital Signature Generation & verification	DSA >= 112 bits of security strength DSA: len(p) >= 2048 AND len(q) >= 224
	RSA: len(n) >= 2048
	ECDSA: len(n) >= 224
Key Agreement using DH & MQV using finite fields, EC	DSA >= 112 bits of security strength DSA: len(p) >= 2048 AND len(q) >= 224
RSA based Key Agreement& Transport	Len(n) >= 2048

HASH Functions

NIST, ENISA Recommended Algorithms		References
SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 and SHA-512/256.	Based on modular arithmetic and logical binary operations	SP-800-107 rev-1
Whirpool	Uses AES style methods	EINSA-2014 report
SHA-3 Family (SHA3-224, SHA3-256, SHA3-384, SHA3-512)		

HASH Algorithm	Digest Length	Security Strengths (CR, First-Pre-Image, Second-Pre-Image)
SHA-1	160	(<80, 160,105-160)
SHA-224	224	(112, 224, 201-224)
SHA256	256	(128, 256,201-256)
SHA-384	384	(192, 384, 201-256)
SHA-512	512	(256, 512, 394-512)
SHA-512/224	512/224	(112, 224, 224)
SHA-512/256	256	(128, 256, 256)

Message Authentication Code

- The purpose of a MAC is to authenticate both the source of a message and its integrity without the use of any additional mechanisms.
- MACs Using Block Cipher Algorithms (SP800-38B)
- MACs Using Hash Functions (SP800-107)
- Universal Hash Functions: UMAC, GMAC,
- Approved algorithm for generating and verifying message/data authentication codes: **HMAC** (FIPS SUB 198-1)
- HMACs have two functionally distinct parameters, a message input and a secret key known only to the message originator and intended receiver(s).

MACs

Crypto Primitive	NIST, ENISA Recommended Algorithms	References
Message Authentication Code: generation & verification	HMAC (>112), CMAC(with AES,3 key TDEA), GMAC (AES)	800-131Ar1

Random Number Generators

RNGs	Description
HASH_DRBG	uses an approved hash function during the instantiate, reseed and generate functions.
HMAC_DRBG	HMAC_DRBG uses multiple occurrences of an approved keyed hash function, which is based on an approved hash function.
CTR_DRBG	DBRG based on block cipher uses an approved block cipher. CTR_DBRG uses a block cipher (TDEA or AES) in CTR mode.
Dual_EC_DRBG	Dual Elliptic Curve random number generator was recently withdrawn.
NRBG Generators	No Approved methods

RBG & Key Agreement

Crypto Primitive	NIST, ENISA Recommended Algorithms	Refs
Random bit Generators	HASH_DRBG, HMAC_DRBG CTR_DRBG	SP 800-90A
Key agreement using Diffie-Hellman (DH) and Menezes-Qu-Vanstone (MQV) schemes using finite fields	>=112 bits of security strength; len(p) >= 2048 and len(q) >= 224	SP 800-56A

Random Number Generators

RNGs	Description
HASH_DRBG	uses an approved hash function during the instantiate, reseed and generate functions.
HMAC_DRBG	HMAC_DRBG uses multiple occurrences of an approved keyed hash function, which is based on an approved hash function.
CTR_DRBG	DBRG based on block cipher uses an approved block cipher. CTR_DBRG uses a block cipher (TDEA or AES) in CTR mode.
Dual_EC_DRBG	Dual Elliptic Curve random number generator was recently withdrawn.
NRBG Generators	No Approved methods

Study crypto library APIs

Following c/c++, java libraries are studied for the above requirements

C++ based Libraries

1. Openssl libraries
2. nss (Network Security Services)
3. Crypto++ 5.6.2
4. Botan
5. AWS s2n library for TLS encryption

Java Based Libraries

6. GNU Crypto project
7. BouncyCastle

Others

9. PyCrypto - Python Cryptography Toolkit
10. Perl Crypto

Openssl:libcrypto

- The OpenSSL **crypto** library implements a wide range of cryptographic algorithms discussed in the previous section.
- Sub libraries of Libcrypto
- evp: evp - high-level cryptographic functions. The EVP library provides a high-level interface to cryptographic functions.
- bn: This library performs arithmetic operations on integers of arbitrary size. It was written for use in public key cryptography, such as RSA and Diffie-Hellman.
- SHA: SHA3 also included
- buffer: The buffer library handles simple character arrays. Buffers are used for various purposes in the library, most notably memory BIOs.
- Others: IO, Error handling, Data encoding, lhash etc.
- FIPS 140-2 Validated.

Openssl:libcrypto

Crypto Primitive	Algorithms supported	API
Symmetric Ciphers	DES, TDEA, AES, other stream ciphers	EVP_Encrypt, EVP_Decrypt family of APIs
Asymmetric Key Ciphers	RSA – encrypt /decrypt/sign/verify	RSA_Family APIs
	DH - Key Agreement	DH_ Failimy APIs
	EC - operations on elliptic curves over finite fields.	EC_ family APIs
	DSA – Digital Signature	DSA_ Family APIs

NSS libraries

- Libnss library implements a wide range of cryptographic algorithms discussed in the previous section
- NSPR is a platform abstraction library that provides a cross-platform API to common OS services
- The NSS cryptographic module has two modes of operation: the non-FIPS (default) mode and FIPS mode.
- NSS – FIPS mode: APIs are very clear well documented.
- A core element of NSS is FreeBL, a base library providing hash functions, big number calculations, and cryptographic algorithms.
- Softoken is an NSS module that exposes most FreeBL functionality as a PKCS#11 module.

NSS libraries

Crypto Primitive	Algorithms supported
Symmetric Encryption	DES, TDES, AES(wECB,CBC modes) RC2, RC4
HASH Functions	SHA-1/SHA-256/SHA-384/SHA-512
Digital Signature	RSA, DSA, ECDSA
Key Agreement	RSA, Diffie-Hellman, ECDSA
DRBG	HASH_DRBG
References	http://www-archive.mozilla.org/projects/security/pki/nss/nss-3.11/nss-3.11-algorithms.html http://hg.mozilla.org/projects/nss/file/tip/lib

Crypto++

- Crypto++ implements a wide range of cryptographic algorithms discussed in the previous section
- The FIPS validated library is only available on Windows as a DLL.
- Simpler to understand the APIs: Namespaces,
- Good Documentation: main web page

Crypto++

Primitive	Supported Algorithms
Symmetric Ciphers	3DES, AES, Twofish, Serpent, CAST-256 etc
Asymmetric Ciphers	RSA, DSA, ElGamal, Nyberg-Rueppel etc
HASH Function	SHA-1, SHA-2, Whirlpool etc
MAC	HMAC, GMAC, CMAC
Key Agreement	Diffie-Hellman (DH), Unified Diffie-Hellman (DH2), Menezes-Qu-Vanstone (MQV), LUCDIF, XTR-DH

Botan

- Botan C++ library implements a wide range of cryptographic algorithms discussed in the previous section
- Simpler to understand the APIs
- Good Documentation
- FIPS140-2 Compliant ?

Botan

Primitive	Algorithms supported
Symmetric Ciphers	DES, 3DES, AES and alsa20/XSalsa20, ChaCha20, and RC4 stream ciphers
Asymmetric Cipher	RSA (ENC/DEC), RSA,DSA, ECDSA - Signatures
Key Agreement	Diffie-Hellman, ECDH
HASH Functions	SHA-2, SHA-3 & Others

Java Based Crypto Libraries

GNU Crypto Library

- Versatile, high-quality, and provably correct implementations of cryptographic primitives
- Implemented most of the approved algorithms

Bouncy Castle

- Light weight cryptp APIs
- JCA compliant
- Easy installation
- Adequate documentation
- Free

Crypto using Python & Perl

PyCrypto

- The Python cryptography toolkit is intended to provide a reliable and stable base for writing Python programs that require cryptographic functions.
- Provides simple, consistent interface for similar classes of algorithms.

PerlCrypto

- There are many crypto packages are available (search in http://search.cpan.org/)
- CryptX - Crypto toolkit is a self-contained perl crypto package. License is free.
- NET::SSLeay module contains perl bindings to openssl.

Performance

Average speed measured for each library for AES, 3DES are shown below Library. The buffer size of the data used in this measurement rangers from 16 bytes to 1 MB.

Cipher/Library	openSSL	Botan	Crypto++	nsslibs
Rijndael AES	44,153 KB/sec	21,807 KB/sec	27,155 KB/sec	N/A
3DES	12,070 KB/Sec	6,698 KB/Sec	6,744 KB/Sec	N/A

Common Vulnerabilities Exposures

Library	CVEs
Openssl	19 CVEs published in 2014, and about 10 during 2015
NSS	6 CVEs published on NSS libraries in 2015
BOTAN	7 CVEs in 2015, 1 in 2014
Crypto++	1 CVE in 2015. CVE-2015-2141
AWS s2n	1 in 2015
Bouncy Castle	1 CVE in 2015. CVE-2015-7940

Conclusion

OpenSSL, NSS libraries have been in use since long time and under-go constantly scrutiny by the researchers and users. They are FIPS 140-2 complaint and implements NIST approved algorithms. Despite the CVEs, these libraries are well maintained and supported. Most the CVEs are due to software implementation and might affect a narrow scenario in some cases.

All the new applications should use open source libraries that are compliant to NIST recommendations.

End

Q & A

Older Slides/backup slides

- Older Slides/backup slides to follow

Comments

- The growth of elliptic curve use has bumped up against the fact of continued progress in the research on quantum computing, which has made it clear that elliptic curve cryptography is not the long term solution many once hoped it would be
- https://www.nsa.gov/ia/programs/suiteb_cryptography/

NSS API structure

FIPS Mode	Default Mode
General-purpose functions Encryption functions Decryption functions Message digesting functions Signature and MAC generation functions Signature and MAC verification functions Dual-function cryptographic functions Key management functions Random number generation functions Slot and token management functions Session management functions Object management functions Parallel function management functions	base, certdb, certhigh, ckfw, crmf, cryptohi, dbm, dev, freebl, jar, libpkix, nss, pk11wrap, pkcs12, pkcs7, pki, smime, softoken, sqlite, ssl, sysinit, util, zlib

Useful Links

- http://csrc.nist.gov/groups/ST/toolkit/examples.html
- http://csrc.nist.gov/groups/STM/cavp/index.html
- http://www.keylength.com./
- https://en.wikipedia.org/wiki/Comparison_of_TLS_implementations
- http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar1.pdf

Key Generation

Key Generation(NIST SP 800-57 revison-3)

- Two types of automated key-establishment schemes are defined: key transport and key agreement. (NIST SP 800-56A, SP 800-56B)
- Key Agreement: The key-agreement schemes employ public-key techniques utilizing Discrete Logarithm Cryptography (DLC). The security of these DLC-based key-agreement schemes depends upon the intractability of the discrete logarithm problem.
- Key Transport: Key transport is the distribution of a key (and other keying material) from one entity (the sender) to another entity (the receiver). The keying material is encrypted by the sending entity and decrypted by the receiving entity/entities.