Native or External?

Lessons learned implementing cryptography for VisualWorks

Martin Kobetic
Cincom Smalltalk Engineering
ESUG 2011

Let's implement SSL!

DES, MD5, SHA, RSA, DSA, RC4, X.509, ASN.1, DER, ...

* SSL 3.0, RSA, DES, RC4, basic X.509 (RSA only)
* DH, DSA, + SSL integration, AES
* X.509 on ASN.1, faster RSA (CRT),...
* new ASN.1 (read/write), more X.509
* protocols/S (HTTPS, SMTPS,....)

What is Cryptography

hashes
MD5, SHA1, SHA256, ...

secret key (symmetric) ciphers
AES, DES, RC4,...

public key algorithms
* signing (RSA, DSA, ECDSA)
* encryption (RSA)
* key agreement (DH, ECDH)

Hashes - Native

```
(MD5 hash: 'Hello') asHexString.

buffer := ByteArray new: 16384.
hash := SHA new.
file := (ObjectMemory imageFilename
            withEncoding: #binary) readStream.
[     [ file atEnd ] whileFalse: [ | read |
            read :=
                  file nextAvailable: buffer size
                        into: buffer
                        startingAt: 1.
            hash updateWith: buffer from: 1 to: read ].
] ensure: [ file close ].
hash digest asHexString.
```

Hashes - External

```
buffer := ByteArray new: 16384.
hash := Hash new algorithm: 'SHA1'; yourself.
file := (ObjectMemory imageFilename
            withEncoding: #binary) readStream.
[     [ file atEnd ] whileFalse: [ | read |
            read :=
                  file nextAvailable: buffer size
                        into: buffer
                        startingAt: 1.
            hash update: read from: buffer ].
      hash finish asHexString
] ensure: [ file close. hash release ].
```

Hashes - Xtreams

```
((ObjectMemory imageFilename reading
        hashing: 'SHA1'
)       -= 0;
        close;
        digest
) asHexString
```

Ciphers - Native

```
message := 'Hello World!' asByteArrayEncoding: #ascii.
key := 'Open Sesame!!!!!' asByteArrayEncoding: #ascii.

((ARC4 key: key) encrypt: message) asHexString.

cipher := AES key: key.

cipher := CipherBlockChaining on: cipher.
iv := ByteArray new: 16 withAll: 1.
cipher setIV: iv.

cipher := BlockPadding on: cipher.
(cipher encrypt: message) asHexString
```

Ciphers - Speed

```
megs := 100.
buffer := ByteArray new: 1000.
time := [
	megs * 1000 timesRepeat: [
		1 to: buffer size do: [ :i |
			buffer at: i put: (buffer at: i) ]]
	] timeToRun.
megs asFloat / time asSeconds.

time := [ self readWriteMegs: megs ] timeToRun.
megs asFloat / time asSeconds
```

Ciphers - External

```
buffer := ByteArray new: message size + iv size.
cipher := Cipher new.
padding := iv size - (message size \\ iv size).
padding := ByteArray new: padding withAll: padding.
[       | count |
        cipher algorithm: 'AES' mode: 'CBC'
                key: key iv: iv encrypt: true.
        count := cipher update: message size
                        from: message into: buffer.
        result := buffer copyFrom: 1 to: count.
        count := cipher update: padding size
                        from: padding into: buffer.
        result := result, (buffer copyFrom: 1 to: count).
        count := cipher finishInto: buffer.
        result, (buffer copyFrom: 1 to: count)
] ensure: [ cipher release ]
```

Ciphers - Xtreams

```
(((   ByteArray new writing
          encrypting: 'AES' mode: 'CBC' key: key iv: iv
)     hashing: 'SHA1'
)     write: message;
      write: padding;
      close;
      terminal
) asHexString
```

Public Key - Native

```
keys := RSAKeyGenerator keySize: 1024.
keys publicKey.

rsa := RSA new privateKey: keys privateKey.
rsa useMD5.
sig := rsa sign: message.
```

Public Key - External

```
key := PrivateKey RSALength: 1024.
[       | digest |
        digest := (message reading hashing: 'SHA1')
                                -= 0; close; digest.
        key sign: digest
                hash: 'SHA1'
                padding: 'PKCS1'
] ensure: [ key release ]
```

Native - Pros

* understanding and know-how
* ease of use and deployment
* automatically cross platform
* easy integration
* debugging

Native - Cons

* maintenance/evolution
* security issues
* speed
* certification
* hardware integration
* export restrictions

External - Pros

* development cost (maybe)
* evolves for free (hopefully)
* speed
* certification (possibly)
* hardware integration (possibly)

External - Cons

* platform coverage
* integration issues
* support
* FFI issues
* brittleness

Summary

* seamless use
* seamless deployment
* platform coverage
* capability coverage
* extensibility
* other constraints
    * certification/approved implementations
    * hardware support
    * performance