Subject: Re: StreamCypher - A package for stream data cryptography
Posted by mdelfede on Thu, 30 Sep 2010 20:45:44 GMT
View Forum Message <> Reply to Message

Mindtraveller wrote on Thu, 30 September 2010 22:01I fully support idea of making uniform interface.
We may even have one parent virtual class like CryptoEncoder and make our implementations as it's descendants. This could be useful for flexibility of usage in user code.

Yep, nice idea the base class. More than a virtual base class, I'd use a class with pure virtual members instead, bringing the interface and, maybe, the streaming-FIFO behaviour, see below.

BTW, as Dolik-Rce suggested in a PM, I shall also add a Decode() member, which is the same as Encode() in my case but not in general.
Maybe it would be better to name them Encrypt() and Decrypt() ?
Another addition would be some

String MD5Key(String const &);

also suggested by dolik-rce, which should avoid the hassle of fixed-length keys needed by Snow2 and (IIRC) by AES too, generating an hashed 32 bit key with any input string.

Quote:
Let's discuss interface in detail.

1. Constructors
Snow2() - OK
Snow2(const String &key) - OK
Snow2(byte const *keyBuf, int keyLen) - I believe first parameter should be const byte *keyBuf (in your case you define pointer which points to changeable data but cannot be changed itself).

Hmmmm, not right. The forms

byte const *ptr
const byte *ptr

Are equivalent. A const pointer to byte should be declared as

byte * const ptr

Anyways, I agree that first form is a bit more clear.

Quote:
2. Same for SetKey.
BTW, what do you do if user tries to encrypt with no key set?

Well, up to now, nothing.... I'm still undecided if throw an exception or generate a random key and go on.
I think former is better, as random key is almost useless if not for testing purposes.
I thought also on removing the constructor with empty params, but it can be useful on reusing the encoder : you can create an encoder without key and then use multiple SetKey().

BTW, I shall also add an initialization number, I guess that it's a must for stream cypher.... So another (qword) optional parameter to constructor and to SetKey.
The initialization vector should be useful also for block cypher, so it would be a good idea to add for both.

Quote:
3. Agreed with Encode interface with exception as in (1) with change (byte const *) to (const byte *).

4. My encoder works with data of any size. It just internally adds random data for stream to be of 128-bit (not byte!) aligned size.

Perfect ! But I just don't understand the need of the datasize parameter.... It isn't possible to use a FIFO-style queue, without a fixed buffer ? I mean... You drop data into the stream, they get encoded, blockwise in your case and stored in a growing buffer; extracting encoded data should just make the buffer shrink as needed.
I could do that easily for my stream encoders, so the interfaces would be identical.
What do you think about ?

Another stuff... I've seen in your docs that your encoder is 'not reusable', I mean you have to create a new one if encoding a new stream.
I think it would be better to add some reset mechanics inside the (newly added) SetKey() member, which would allow re-keying and resetting the encoder.

Adding a simple Reset() would be dangerous, as it would need to store the key or at least the S-Box, which could lead to easy keyfinding inside a running app,

Anyways, on week end I'll polish my interface and add the stream-FIFO stuff, so it'll be ready to be merged into a single cypher package.

About the name... what do you think about 'Cypher' ?

Last but not least... as we're making a general Cypher package, what about asymmetric-key encoding like RSA ?
That one would be a nice addition too, and we should fit it into the base interface.... But I don't know too much about RSA.
The problem is that you can both encode with pub key (true encoding) and encode with priv key

(digital signature)
and the way around, decode with priv key (true decoding) and decode with pub key (digital signature check).
So... if we keep the SetKey() as before, we can use the encoder just for a purpose at once (which IMHO is the most common usage...), otherwise we shall add a SetKey with both keys (IMHO overkilling, but not sure about).

Max
