Subject: Re: StreamCypher - A package for stream data cryptography
Posted by Mindtraveller on Thu, 30 Sep 2010 21:44:04 GMT
View Forum Message <> Reply to Message

Quote: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. Agreed. That was exactly what I meant.

Quote: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() ? Encrypt/Decrypt fits better than Encode/Decode.
But I must disagree about Decode. There is a problem if user calls Encrypt and then Decrypt. I have two classes: one for decryption and other for encryption. They have different internal mechanisms and I don't really know why they should be merged into one class. The general problem is that encryption and decryption are really different processes in AES and they cannot be called one by one (especially if we want streaming).
My proposal is to have Encryptor class and Decryptor class (they might be the same in your case).

Quote:String MD5Key(String const &); Latest investigations revealed potential weaknesses in MD5 hashing. That is why I used SHA256 instead of MD5 for internal key generation. So I agree with you in general (yes, we must generate hash from user password as internal crypto-key), but i suggest using SHA256 instead of MD5. Function can be found in AESStream.cpp @ 42:
String AESHashedString(const String &s)

Quote:Hmmmm, not right. The forms
byte const *ptr
const byte *ptr
Are equivalent You are right. I mixed it up with (byte * const ptr). Neverthless I would suggest using (const byte *) notation as more readable in general.

Quote: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(). I suppose it could make more problems than solve them. SetKey() makes a danger of changing the key while en/decryption streaming is in progress. That is why I denied any possible empty constructors. My thought was to make my classes stable by design. It could be good to hear suggestions from AESStream users, is SetKey() really good here.
Finally, if we agree with SetKey(), I will add it for the sake of uniformness. But I'd warn us against creating any potential problems.

Quote:BTW, I shall also add an initialization number, I guess that it's a must for stream cypher If you mean initial vector for streamed encryption, I must say that only good way to make it is to use OpenSSL's random number generation. There was a number of investigations about initial vector. In short, the conclusion was: NEVER use constant initial vector. The best initial vector is random vector. That is why I generate random initial vector in constructor without any doubts.

Quote:But I just don't understand the need of the datasize parameter. Data size is written into the header of my protected container. So decryptor "knows" the overall size of data even if user doesn't (that's good feature). Besides, streamed decryption becomes more robust as it doesn't decrypt more data than needed.

Quote: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 'Not reusable' is a drawback of design I've chosen (see above). It is discussable of course.
Talking about Reset(), yet we have to keep the key in memory if we support streaming. So we have to think how to crypt the key in memory to avoid plain keyfinding. Any ideas are welcome here.

Quote:About the name... what do you think about 'Cypher' ? It doesn't make big difference for me. Maybe, if we use Encryptor/Decryptor name, we could call package Crypto. If you don't like it, please fill free to use Cypher or anything else.

Quote:Last but not least... as we're making a general Cypher package, what about asymmetric-key encoding like RSA ? I would not assume highly different crypto scheme to use the same interface. I propose making what we planned before, catch any bugs there. And only after that to think about RSA. I used it about 10 years ago and as I remember it should use different scheme with a different interface.

Quote:Thinking about it... I'll drop a Cypher package into Bazaar on week end, with the basic interface, if we agree about it on these days.
Then I can add my couple of stream cyphers, and you your AES one.
Next we can update the CypherTest package too, allowing to test all the encoder supplied, and we can update the docs.
Finally, I'll remove the StreamCypher package and you your AesEncoder.
What do you think about ? OK.