Subject: [REPLACED by Cypher package]StreamCypher - A package for stream data cryptography
Posted by mdelfede on Wed, 29 Sep 2010 17:37:20 GMT
View Forum Message <> Reply to Message

[************* REPLACED BY Cypher package **********]
Please use newly added Cypher package, it does all in StreamCypher and more. StreamCypher will be removed when Cypher docs will be ready
[**]

I added StreamCypher package, along with its test app StreamCypherTest, which implement 2 classes for stream cryptography :

RC4   for RC4 cryptography (weak protection, small footprint)
Snow2   for Snow2 cryptography (strong protection, higher footprint)

The test app allows to encode/decode files using both classes.
Small documentation also included.

Ciao

Max

Subject: Re: StreamCypher - A package for stream data cryptography
Posted by Mindtraveller on Thu, 30 Sep 2010 08:03:47 GMT
View Forum Message <> Reply to Message

mdelfede, Koldo suggested merging our streamed cypher classes into one package: your and my AESEncoder/AESDecoder. I agree with him. What do you think?

Subject: Re: StreamCypher - A package for stream data cryptography
Posted by mdelfede on Thu, 30 Sep 2010 12:49:47 GMT
View Forum Message <> Reply to Message

Yep, that could be done.
We should indeed change some of both codes to have an uniform interface.
I have those :

```
	// constructors
	Snow2();
	Snow2(const String &key);
	Snow2(byte const *keyBuf, int keyLen);
	// key settings
	bool SetKey(const String &key);
	bool SetKey(byte const *keyBuf, int keyLen);

	// encode string
	String Encode(String const &s);

	// encode buffer, dest on different buffer
	void Encode(byte const *sBuf, byte *dBuf, dword bufLen);

	// encode buffer in place
	void Encode(byte *buf, dword bufLen);
```

(this for Snow2, exactly the same for RC4 encoding)

So I shall add streaming capabilities as your code, with operators << and >>; you should add the same as I have in my code above, so we'll have an uniform interface.
Is it feasible for you ?
For me, I could add dynamic streaming capabilities, as I've no need of a fixed block size.

Ah, BTW... AFAIK AES works on blocks of fixed size of 128 bits.... is the same for your encoder ?
I mean... you must work on multiple of 128 bytes ?

Ciao

Max

Subject: Re: StreamCypher - A package for stream data cryptography
Posted by Mindtraveller on Thu, 30 Sep 2010 20:01:15 GMT
View Forum Message <> Reply to Message

I 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.

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).

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

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.

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

Subject: Re: StreamCypher - A package for stream data cryptography
Posted by mdelfede on Thu, 30 Sep 2010 20:49:13 GMT
View Forum Message <> Reply to Message

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 ?

Max

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.

Subject: Re: StreamCypher - A package for stream data cryptography
Posted by mdelfede on Thu, 30 Sep 2010 22:58:39 GMT

View Forum Message <> Reply to Message

Mindtraveller wrote on Thu, 30 September 2010 23:44
....
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).

Encrypt() and Decrypt() are the same process in Snow/RC4, that's true, but they can't really mixed without resetting the machine state, so calling SetKey().
I'd like to keep both in a single class and throw an exception if mixing Encrypt/Decrypt calls without key resetting... but we can speak about it, of course.
That's just because of usage simplicity, you'd need just an Snow/AES class and call Encrypt/Decrypt, instead have both SnowEncrypt/SnowDecrypt classes and call, for example, Process() on both.

A "mixed" solution would be to have Snow class with both Encrypt()/Decrypt(), an AESEncrypt with a working Encrypt() and a Decrypt which just throws an exception and the way around for AESDecrypt class.

I think that the better solution depends on how is built the state machine for AES, which I don't know. If machines are the same, and just the encoding/decoding processes are different, I'd vote for a single class.... If also the state machines are different, I'd vote for 2 classes.

Quote:
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)

Agree, MD5 was just an example, your's is better

Quote:
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.

Well, Setkey() is maybe not good when streaming, but is indeed comfortable for block encoding/decoding with different keys.
Of course, SetKey() should reset all mechanics, including streaming state and buffers.
Anyways.... SetKey becomes useful also when re-using the class for encoding/decoding, which I find also comfortable behaviour.

Quote:
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.

We can't assume that user can accept a random number appended to its data. Just think to disk sector encoding, for example. Size of input data must be identical to size of output data.
In my Protect package it's foreseen to use random data as the 'nonce' vector, but it's separated from encoded block. There are cases in which you have to assume a constant or an user-assignable initialization vector kept outside of data stream. Also encoding a block in-place would be impossible if data sizes don't match.
So, IMHO, we should make it optional somehow, for example with a constructor parameter/SetKey parameter which can be Null by default meaning a random init vector, and provide a GetIV()/SetIv() to get or set it if needed.
The defaults could be different from AES and Snow/RC4, and streaming of 'nonce' vector could be made optional by constructor or member function also.

So, if you want to keep your AES compatible with former version, you could make it on by default in your class, and I could make it off in mines.

Quote:
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.

what about an eof() on class ?
Stream crypto is foreseen for variable/unknown stream sizes, so it should anyways have an IsEof() function for streaming.
And also for your class it would be good and allow to retrieve just the encoded data size letting your class to bother about it.

Quote:
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.

No, I think we have to keep just the S-Boxes in memory, which is far safer. At least, for Snow and RC4, but I guess for all cases should be the same. I don't want the key hanging around for more time than strictly needed

Quote:
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.

Well, both Cypher and Crypto are equally good

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]
ok, that was just for planning the right interface, common to all if possible.

Subject: Re: StreamCypher - A package for stream data cryptography
Posted by dolik.rce on Fri, 01 Oct 2010 15:43:00 GMT
View Forum Message <> Reply to Message

Hi Max,

Today while trying to port Snow 2.0 to php (still no success) I found out about a newer version, Snow 3G. Out of curiosity I downloaded specs and sample implementation to play with it a bit and to see if it wouldn't be easier to port.

A really simple package I experimented with is attached in case you are interested It has no interface, since I believe you can fit it straight into the new interface that is being developed.

Best regards,
Honza

PS: The spec are located here: http://www.gsmworld.com/documents/snow_3g_spec.pdf

File Attachments
1) Snow3G.zip, downloaded 486 times

Subject: Re: StreamCypher - A package for stream data cryptography
Posted by mdelfede on Fri, 01 Oct 2010 15:50:51 GMT
View Forum Message <> Reply to Message

Hi Honza,

did you try with the reference implementation of Snow2.0 instead of the fast version ? It should be easier to port, and anyways we don't need a very fast encryption for PHP side, we just need to encrypt some small strings.

About the package, I'll implement first the interface, then add the already coded RC4 and Snow2, then we could add some other encryptors.

I've got to add some stuffs to Protect too.... the initialization vectors, for example, they're still missing.
Then I'd like to take your PHP and add a web auth module to Protect.... even with just plain RC4 if you couldn't port Snow to PHP. We can add the rest later

Ciao

Max

Subject: Re: StreamCypher - A package for stream data cryptography

Posted by dolik.rce on Fri, 01 Oct 2010 16:31:56 GMT
View Forum Message <> Reply to Message

mdelfede wrote on Fri, 01 October 2010 17:50Hi Honza,
did you try with the reference implementation of Snow2.0 instead of the fast version ? It should be easier to port, and anyways we don't need a very fast encryption for PHP side, we just need to encrypt some small strings.
Well, the trouble is not really the complexity. The biggest problem is the low level approach. Both the slow and the fast reference implementations use type dependent tricks, which makes it nightmare in php as it has only one integer type which has no idea about signedness and to make it even more fun it's range is platform dependent. I managed to rewrite it into working php code, it "just" doesn't work correctly I guess I missed some of the overflows. It might be actually easier to implement it from scratch looking only in specs... But I'll definitely keep trying, it drives me mad when I can't achieve something so simple (~400 lines of code)

Honza

Subject: Re: StreamCypher - A package for stream data cryptography
Posted by Mindtraveller on Sat, 02 Oct 2010 06:15:15 GMT
View Forum Message <> Reply to Message

Quote:If also the state machines are different, I'd vote for 2 classes. I don't really know exactly how it works in OpenSSL implementation. I just use it. And the ways, I use it for decryption/encryption in my classes, are different. Again: these are two different state machines in my case, that is why I'd prefer 2 classes instead of one.

Quote:No, I think we have to keep just the S-Boxes in memory, which is far safer OK. I have to add s-box support into AES classes too.

Quote:Well, both Cypher and Crypto are equally good Then I'd suggest Crypto as it sounds closer to Decrypt/Encrypt classes.

Please let me know when you make common interface for our classes.

Subject: Re: StreamCypher - A package for stream data cryptography
Posted by mdelfede on Sat, 02 Oct 2010 07:25:19 GMT
View Forum Message <> Reply to Message

Hi Pavel,

I started coding the interface... I had to name it Cypher because Crypto package is already on Uppsrc
I don't know its purpose, either... it's a quite small package.

As I see you use external openssl to make the encryption...

Here
 http://www.ultimatepp.org/forum/index.php?t=msg&&th= 5547&goto=28904#msg_28904

Zbych attached an AES256 module which could help to make the class self-contained without need to external apps. Thet (AFAIK) would allow you to do all without having to store the key.

I think I'll post the starting interface and my encryptors today or tomorrow, so you can review it.

Ciao

Max

Subject: Re: [REPLACED by Cypher package]StreamCypher - A package for stream data cryptography
Posted by mrk10000 on Wed, 27 Apr 2022 04:28:10 GMT
View Forum Message <> Reply to Message

Hi, excuse my silly question but i am not familiar with crypto i just want to use snow2 to encode and decode plain text on my app but i cannot find how to decrypt, just encrypt (i just cjecked both examples: streamcyper and cypher), thanks for your help

Raul

Subject: Re: [REPLACED by Cypher package]StreamCypher - A package for stream data cryptography
Posted by mdelfede on Wed, 04 May 2022 10:42:23 GMT
View Forum Message <> Reply to Message

They're simmetric encoders/decoders.
If you apply 2 times you get original text.

```
// key must be 16 or 32 bit... here a sample 16 bit key
key = "\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10";

// create encoder
Snow2 snow(key):

// encode text
String s "Some text to encode";
String encoded = snow(s);

// reset encoder/decoder
snow.SetKey(key);
```

```
// decode text
String decoded = snow(encoded);
```

Ciao

Massimo

Subject: Re: [REPLACED by Cypher package]StreamCypher - A package for stream data cryptography
Posted by mrk10000 on Wed, 11 May 2022 01:46:33 GMT
View Forum Message <> Reply to Message

Thanks a lot i will try today