# AN OPTIMAL BINARY MESSAGE PROTOCOL FOR STREAMING BITS

Author: Thotheolh (info@askg.info).

Date: 11th November 2014.

Change Log:

| Date | Comments |
|---|---|
| 11 Nov 2014 | Original Release |
| 30 Nov 2014 | Updated Theorem and Application |

## DISCLAIMER

The ideas within this paper are stated here independently with no awareness of any such protocols or ideas existing which may have already existed. The usage of the ideas here in a production environment would be at one's own risk.

## ABSTRACT

A stream-based cryptographic protocol (One Time Pad or Stream Cipher) uses a bitstream of internal states or a keystream to encrypt a bitstream of plaintext message. The security of the message depends on the size of the bitstream of the internal state or keystream used as every bit of plaintext would be encrypted with a corresponding bit from a keystream or internal state. In the event that the message is small in size (weak message), the keystream or internal state would also be weak. Cryptanalysis via the message length would be plausible and the application of bruteforce attacks would not be too far fetched for a weak message. The successful bruteforcing and revelation of possible message, keys or cipher states used to encrypt the message may allow other attacks to propagate and further reducing security of the message stream. In this paper, I would like to define the security of a stream-based encrypted message with the formula of $2^n$ where the $n$ represents the bit length of the message. Using the concept of how a block cipher encrypts data in blocks, messages can be made stronger to discourage bruteforce on a stream-based message.

## SECURITY DEFINITION

The definition of the formula of $2^n$ as the security measurement for a stream-based cryptographic message is used to determine the overall keystream or internal state bit size being used to encrypt a stream of message assuming that the message is sent over a communication medium in short burst (message session). A further extension of the formula under possible influence of a birthday attack would set the formula to an estimate of $2^{(n/2)}$.

An example of the application of this formula would be to predict the security measurement for a stream-based encrypted message of the text "hello" in an 8 bit ASCII format which would be a 40 bit message that would yield under $2^n$ formula a $2^{n=40}$ bit security and after the application of a possible birthday attack, it would only have about a $2^{n=20}$ bit security. NIST guidelines under the SP800-57[2] and SP800-131[1] have proposed guidelines for key length of at least 80 bits and the 40 bit security for

the "hello" message using 40 bits of keystream or internal state would not be adequate and under a birthday attack influence would reduce it to about a 20 bit keystream or internal state.

The proposed method to increase the security of a stream-based cryptographic protocol is to extend the amount of message bit length so that the message used would result in the use of keystream or internal state of an acceptable length to discourage the use of bruteforce attacks. A recommended basic *n* size of 96 bits would give be more than sufficient compared to an 80 bit security recommended by NIST and would theoretically be slightly slower than the use of an 80 bit key but still fast enough for implementations that require fast encryption with small key sizes. A recommended *n* size for standard messages that are not constrained or bound by the device or algorithm would be using a 128 bit *n* size and for higher levels of security, a *n* size of 256 bits or more would be recommended.

Messages of suitable *n* sizes would be pack into blocks and sent off in block streams but encrypted as bit streams which will be padded to fill the selected *n* size for each block. If the *n* size is selected with a strong size that discourages bruteforce attacks via traffic analysis of the message length, the keystream bits or state used would also be strong under the assumption that at least *n* size of keystream bits or state is used during cryptographic operations.

## THEOREM OF STREAM CRYPTOGRAPHY SECURITY

### Security Problem

The problem this paper covers relates to bruteforce attacks on a weak message length or a weak size key used in cryptography based on streaming messages. The main application for this paper would be the One Time Pad where it is assumed that the message length is equals to the length of OTP key used to encrypt the message. Possible application of such bruteforce attacks on a weak message length might also give insight into the particular state in the state mechanism of a cryptographic cipher which may yield clues to that may leak to attacks on the key, internal state or even the message security of a stream encrypted message. Most of the theories here would be heavily based on OTP encryption and mainly deal with OTP encryption.

Before beginning to explain the mechanism of the theories below, here are a list of symbols that will be used through the paper.

Cardinality: | x |
Concatenation: x | y | z
Equivalence: ≡
Non-equivalence: ≠
If-Else Logic with multiple 'AND' conditions:

```
if ( (x) && (y) && (z) ) :
    something
else:
    otherthing
```

For the entire operation, we will first set a *Bounds* or threshold as to the lowest limit a message *m* should have it's length acceptable. Next, we will Set the length for the secret key *S* and also ensuring that *m* and *S* are equivalent in terms of length. Finally, proving that both *m* and *S* that are equivalent to each other in terms of the threshold *Bounds* being setup at the beginning.

**Definition of Theorem**
In this segment, we will define the generic structure for the usage of the theory.

*Setting Bounds:*

$Bounds = b$
$x : \{1,0\}^{b}$

Here, we set a *Bounds* of length *b* and define that *x* with 1 and 0 as elements and length of *b* whereby *b* is the minimal *Bounds* for which *x* is allowed.

*Checking S secret key Against Bounds:*

$S : \{1,0\}^{*}$
$Bounds \geq |\, S \,| \rightarrow$ Acceptable
$Bounds < |\, S \,| \rightarrow$ Unacceptable

Here, the secret key *S* is defined with 1 and 0 as elements of arbitrary length as long as the length of *S* meets *Bounds* requirement to be acceptable.

*Checking m Message Against Bounds:*

$m : \{1,0\}^{*}$
$Bounds \geq |\, m \,| \rightarrow$ Acceptable
$Bounds < |\, m \,| \rightarrow$ Unacceptable

Here, the secret key *m* is defined with 1 and 0 as elements of arbitrary length as long as the length of *m* meets *Bounds* requirement to be acceptable.

*Proving Strength of Proper M:*

if ( ( $|\, m \,| \rightarrow$ Acceptable ) && ( $|\, S \,| \rightarrow$ Acceptable ) && ( $|\, m \,| \equiv |\, S \,|$ ) ):
    Finally Acceptable
else:
    Finally Unacceptable

The length of *m* and *S* must be acceptable and *m* and *S* must have equivalent length to be finally acceptable since the primary assumptions includes the message *m* must have secret key *S* of same length for message and key.

**Sample Proof of Theorem's Proper Use**

*Setting Bounds*

$Bounds = b$
$b = 80$
$x : \{1,0\}^b$

Minimal *Bounds* of 80 bits are selected as per NIST specifications on key length requirements.

*Checking S Key Against Bounds*

$S : \{1,0\}^{256}$
$Bounds \geq |\, S \,| \rightarrow$ Acceptable

A 256 bit secret key is used.

*Checking M Against Bounds*

$m : \{1,0\}^{256}$
$Bounds \geq |\, m \,| \rightarrow$ Acceptable

A message of 256 bit length is used.

*Proving Strength of Proper M*

$(\, |\, m \,| \rightarrow$ Acceptable $)$ && $(\, |\, S \,| \rightarrow$ Acceptable $)$ && $(\, |\, m \,| \equiv |\, S \,|\, )$:
Finally Acceptable

The message length is acceptable, secret key length is acceptable and message and secret key length are the same which results in being finally accepted as a proper use of the theory.

**Sample Proof of Theorem's Improper Use**

*Setting Bounds*

$Bounds = b$
$b = 80$
$x : \{1,0\}^b$

Minimal *Bounds* of 80 bits are selected as per NIST specifications on key length requirements.

*Checking S Key Against Bounds*

$S : \{1,0\}^{64}$
Bounds < | m | → Unacceptable

A 64 bit secret key is used.

*Checking M Against Bounds*

$m : \{1,0\}^{40}$
Bounds < | m | → Unacceptable

A message of 40 bit length is used.

*Proving Strength of Proper M*

( | *m* | → Unacceptable ) && ( | *S* | → Unacceptable ) && ( | *m* | ≠ | *S* | ) :
Finally Unacceptable

The message is unacceptable from every aspect. The message length (40 bits → bruteforce the mesage) is shorter than the *Bounds*, the secret key is only 64 bits (easily bruteforced) and also shorter than the *Bounds* and finally the key and message lengths are not the same.

**MESSAGE PPROTOCOL**

The proposed message protocol would utilize the *n* size formula in a practical sense to allow the implementation of padding of random bits to hide the true size of the message. A defined number of bits would be placed in front of the protocol to define the actual message size followed by the actual message bits and then followed by a random string of filler padding bits. A 128 bit message would therefore not be able to fit a 128 bit *n* size block due to the header that contains the message size bit indicator. The bit indicator size varies between different *n* size message blocks and will be recommended later in this section.

The ASCII skeleton diagram of the protocol is shown below in an unencrypted block.

Message with *n*=128: [<---8 bits message size---><---message content---><---padding bits--->]

A program parser parsing such a message protocol would have to discern the message size of *n* to know how much of header bits it would read in to find out the actual location and actual message positions within the block. For a 128 bit size of *n*, an 8 bit message header that indicates the message content size would be used and the message content size would be in integer format and then converted to bitstring appended as the header of the message. An example is a message of 29 bits which the header will represent as 00011101 in binary. Padding bits maybe randomly generated and appended behind the message content or can be a predetermined value as long as the header bits are valid. A randomly generated padding bitstring may provide an element of surprise and randomness for attackers.

## RECOMMENDATION OF N SIZES

### Application of Theorem's Proof on Recommended *n* sizes

We will put the theory above to proof the message protocol with it's *n* size meeting *Bounds*.

Message's *n* size block, *Block*, defined with size of 128 bit security provided.
$Block : \{1,0\}^{n=128}$

Message *m*.
$m : \{1,0\}^{*}$

Header *h*.
Since $n$=128, 8 bit header is used.
$h : \{1,0\}^{8}$

Padding *p*.
Size of *p* depends on the subtraction of *n* from size of *m* and size of *h*.
$p : \{1,0\}^{n - |m| - |h|}$

Setting *Bounds* with minimal threshold of 80 bit security.
$Bounds = b$
$b = 80$
$x : \{1,0\}^{b}$

Checking *S* Key Against *Bounds*.

$S : \{1,0\}^{Block}$
$Bounds \geq |S| \rightarrow$ Acceptable

A 128 bit secret key is used due to *Block* being 128 bits.

Composing *M* block message (the entire message with header, message content and padding).

$M = h \mid m \mid p$

Checking *M* Against *Bounds*.

$M : \{1,0\}^{Block}$
$Bounds \geq |M| \rightarrow$ Acceptable

A message of 128 bit length is used due to *Block* being 128 bits.

Proving Strength of Proper *M*.

( | *M* | → Acceptable ) && ( | *S* | → Acceptable ) && ( | M | ≡ | *S* | ):
Finally Acceptable

From the above proves, it is evident that the *n* size must be sufficient to stop a bruteforce attack on the message or keystream. The lowest recommended bounds of *n* size would be 96 bits (not considering birthday attack) which will give about $2^{(n/2)} = 2^{48}$ message or keystream size. While a fully birthday attack resistant *n* size would be 256 bits which will give about $2^{(n/2)} = 2^{128}$ key size.

**Recommendations of Header Sizes**

The header size is calculated by the $2^{|h|} = n$ formula to derive the optimal header bit sizes to use as a binary indicator of the amount of message bits | *m* | in the message *m*.

Below is a list of recommended header size in bits:
n <= 128 will use 7 bits header.
*n* <= 256 will use 8 bits header.
*n* <= 512 will use 9 bits header.
*n* <= 1024 will use 10 bits header.
*n* <= 2048 will use 11 bits header.
*n* <= 4096 will use 12 bits header.
*n* <= 8192 will use 13 bits header.
*n* <= 16284 will use 14 bits header.

It is also recommended to use uniform *n* sizes throughout a cryptographic communication session using stream-based cryptography to repel possible attempts at traffic analysis of unequal message sizes.

**APPLICATION OF THEOREM AND BENEFITS**

The proposed protocol can be beneficial if applied to the field of stream-based cryptography especially in the field of OTP. Below are some proposed application of the above protocol and theory with it's benefits.

**Tinfoil Chat**

The Tinfoil Chat project (TFC) [3] is a secure communication project that uses custom built hardware from cheap off-the-shelf parts from electronics store to demonstrate the application of data diodes in hardware to limit the damages of malwares in a secure communication system from high level threats. It uses OTP as one of it's ciphering scheme to encrypt messages which was the original reason I created this paper to help TFC implement a traffic analysis via bruteforce on *m* with security of $2^n$ while using OTP as the ciphering mechanism.

**Mobile Field Devices**

The wide spread growth of portable smart devices allows rapid communication in real time. Some application of high security portable messaging devices might benefit from short messages (chat messages) secured using OTP with pre-generated or on-the-fly hardware generated random keystream. The benefits of such scheme would be to send short messages on the field securely using OTP that has a huge margin in security as long as the randomness of the bits and the security of $2^n$ while using OTP as the ciphering mechanism are met. If the design of the TFC could be shrunk down to microchip size and embedded in portable secure communication devices, the usage of low bandwidth, low power, low complexity of cryptographic engine and the low processing requirement for sending OTP encrypted messages would be ideal in scenarios where high security is required on the move.

**OTP Key Management via Blacklists**

With the use a fix length keystream / message, you would effectively be able to predict how much keystream bits you need and pre-generate them and distribute them or generate them on the fly. You would be able to detect repeating fix length keystreams that breaks the security of OTP (no repeating keys allowed) by blacklisting them more effectively.

A scheme would be to keep a key hash of spent keys and check the active keystreams for offending key hashes before using them to encrypt messages. This would make key management becomes more visible and manage-able with the use of fix length keystream / message.

**Message Obfuscation**

The additional benefits of splitting messages into multiple blocks would also allow the control of message being sent out to defeat traffic analysis and an attacker would not end up with the precise size of a message. An example would be to use a predefine *n* size of 128 but when sending a *n* size of 129 bits message which uses a 129 bit keystream or state and that would mean the use of 2 message blocks with message bits that cannot fit into the first message block spilling into the second message block with the rest of the message bits in the second block filled with padding bits. The total bits used would be a size of 256 bits which would give a higher *n* size making the cryptanalysis via bruteforce much more difficult and the actual size of the message hidden from plain sight.

**CONCLUSION**

With the use of wrapping messages into fixed size blocks, traffic analysis and bruteforce attacks against vulnerable message with weak *n* sizes becomes much harder as the overall *n* size of the block makes up adequately following the $2^n$ and $2^{(n/2)}$ formula to calculate the weak message sizes of a stream-based message sent out in sessions.

**REFERENCES**

[1] Elaine Baker., Allen Roginsky. Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key
Lengths, 2011, National Institute of Standards and Technology, Gaithersburg MD, 20899, http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdf, (retrieved November 2014).

[2] Elaine Barker., William Barker., William Burr., William Polk., Miles Smid. Recommendation for Key Management – Part 1:
General (Revision 3), 2012, National Institute of Standards and Technology, Gaithersburg MD, 20899, http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf, (retrieved November 2014).

[3] Markus Otella. Tinfoil Chat Messaging platform immune against mass surveillance, 2014, University of Helsinki, Finland, https://www.cs.helsinki.fi/u/oottela/tfc.pdf, (retrieved November 2014).